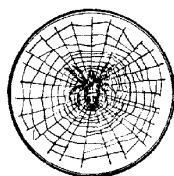


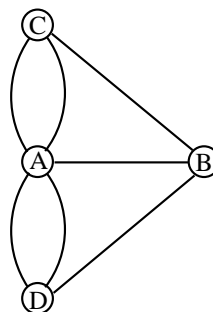
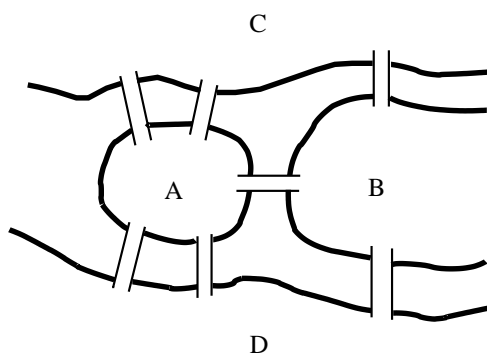
ŽILINSKÁ UNIVERZITA  
FAKULTA RIADENIA A INFORMATIKY

---



Stanislav Palúch

ALGORITMICKÁ  
TEÓRIA GRAFOV



---

VYDALA ŽILINSKÁ UNIVERZITA V ŽILINE, 2008

Tlačová predloha týchto textov bola vytvorená v typografickom systéme L<sup>A</sup>T<sub>E</sub>X pod operačným systémom Linux. Obrázky boli nakreslené programom xfig takisto pod operačným systémom Linux.

Recenzenti: Prof. RNDr. Ján Plesník, Dr.Sc.  
Doc. RNDr. Ferdinand Gliviak, CSc.

---

© S. Palúch, 2008  
ISBN-80-XXXX-XXX-X

# Úvod

Teória grafov ako samostatná matematická disciplína vznikla v prvej polovici dvadsiateho storočia – teda pomerne neskoro – napriek tomu, že niektoré prvky myslenia používaného súčasnou teóriou grafov sa sporadicky objavovali už dávno predtým. Za pioniersku prácu teórie grafov sa považuje práca o probléme siedmich mostov mesta Kaliningrad (pozri časť 6.5.1), ktorú r. 1736 vyriešil vynikajúci švajčiarsky matematik Leonhard Euler. Od toho istého autora pochádza Eulerova polyedrická formula (8.1), ktorá hovorí o vzťahu počtu stien, vrcholov a hrán konvexného telesa. R. 1847 Kirchoff navrhol riešenie zložitého elektrického obvodu s využitím jeho podschémy, ktorú v dnešnej grafárskej terminológii nazývame kostrou grafu (viď. časť 4.5.3). Írsky matematik R. W. Hamilton r. 1859 študoval problémy „cestovania“ po vrcholoch a hranách pravidelného dvanásťstenu. Jednou z úloh, ktoré formuloval, bola aj úloha nájdenia okružnej cesty, ktorá každý vrchol dvanásťstenu obsahuje práve raz. Táto úloha sa stala predchodcom známeho problému obchodného cestujúceho (pozri časť 6.3). Roku 1874 Cayley pri štúdiu štruktúrálnej chemických vzorcov používal grafické zobrazovanie (pozri časť 1.6.2) a v tejto súvislosti Sylvester r. 1878 prvýkrát použil termín graf v dnešnom zmysle teórie grafov.

Definitívny vznik modernej teórie grafov sa viaže na rok 1936, kedy maďarský matematik D. König publikoval prvú monografiu z teórie grafov. Odvtedy sa teória grafov rozvíja v dvoch smeroch – teoretickom a algoritmickom. Teoretický smer viacej študuje rôzne vlastnosti grafov, algoritmický smer sa viac zaoberá hľadaním optimálnych algoritmov na riešenie rôznych, najčastejšie extrémálnych úloh v grafoch. Veľmi dobrou a stále aktuálnou knihou s algoritmickým

prístupom k teórii grafov je [2] z roku 1975, slovenská kniha [13] z roku 1983 a tiež [9] z roku 1999, ktorá navyiac uvádza aj množstvo praktických aplikácií.

V roku 1965 si Edmonds ako prvý uvedomil, že existujú dobré – polynomiálne algoritmy a algoritmy ostatné – nepolynomiálne. Aj problémy teórie grafov sa dajú rozdeliť na dobré – také, pre ktoré existuje polynomiálny algoritmus riešenia a ťažké – také, pre ktoré polynomiálny algoritmus nemáme a veríme, že ani neexistuje. A máme aj problémy, ktoré zatiaľ nevieme zaradiť. Začína sa rozvíjať teória zložitosti. Dnes takmer nie je možné publikovať problém diskkrétnej matematiky bez rozboru jeho zložitosti.

V posledných troch desaťročiach exponenciálne rastie počet úspešných aplikácií teórie grafov. Prispel k tomu aj úžasný rozvoj výpočtovej techniky, ktorý nemá obdobu v žiadnej inej oblasti ľudskej činnosti. Rýchlosť i rozsah pamäte počítačov sa za ten čas zväčšili niekoľkotisíckrát. Cena klesla tak, že počítač je dostupný aj priemerne zarábajúcemu slovenskému občanovi. Teória grafov dostala vo výpočtovej technike silný prostriedok na realizáciu svojich algoritmov.

Bez výpočtovej techniky nemal algoritmus veľký praktický význam. Úlohu malého rozmeru praktik vyriešil preskúmaním všetkých možností, úloha veľkého rozmeru by si pri algoritmickej výpočte vyžadovala pre ručný výpočet neúmerne veľa výpočtov, a preto takúto úlohu riešil praktik intuitívne. V tom čase bol algoritmus súčasťou teórie hovoriacou: „Na to, aby si získal žiadané riešenie, treba urobiť postupne tieto a tieto kroky. Ak budeš dostatočne dlho žiť, dopočítaš sa k riešeniu.“

Dnes je algoritmus teórie grafov návodom, podľa ktorého možno napísať program pre riešenie daného problému. Operačná rýchlosť i rozsah RAM pamäte dnešných počítačov stačí na vyriešenie grafových problémov súvisiacich i s veľmi rozsiahlymi praktickými úlohami.

Ukazuje sa, že teória grafov je veľmi dobrým nástrojom na tvorbu matematických modelov pre najrôznejšie problémy, od modelovania komunikačných sietí, cez modelovanie sociálnych vzťahov, compatibility chemikálií v skladoch, stavov diskrétného systému až po modely pomáhajúce skúmať nukleové kyseliny RNA a DNA. Preto sa dostala do učebných plánov inžinierskeho štúdia na väčšine svetových univerzít.

Pre inžinierske smery je podstatná algoritmickej teória grafov, z ktorej sa tu budem snažiť prezentovať časti s najväčším uplatnením v praxi. Terminológia

---

teórie grafov – slovenská či anglická – je značne nejednotná. Za základ tu prezentovanej som použil Plesníkovu terminológiu z knihy [13]. Pre jednoduchosť presne definujem a ďalej používam iba dve grafové štruktúry – graf a digraf. Ak sa niekedy dostanem aj k zložitejším, vystačím s ich intuitívnym poňatím.

Do tejto publikácie som vyberal hlavne také vety, ktoré formulujú význačné vlastnosti grafov dôležité pre vysvetlenie či konštrukciu algoritmov. Pre mnohé vety uvádzam aj dôkaz, hlavne vtedy, keď je konštruktívny alebo vtedy, keď je založený na často využívanom princípe. Dôkazy viet ukončujem značkou ■.

Pri prezentácii algoritmov som vynaložil veľké úsilie na to, aby k ich realizácii na počítači bol už len krôčik. Pre jednoznačné označenie miesta v texte, kde končí popis algoritmu, používam značku ♣. Postup podľa niekoľkých prvých algoritmov prezentujem v tabuľkách, v ktorých vidieť, ako sa v čase výpočet vyvíjal. Pri ďalších algoritmoch už predpokladám, že čitateľ bude schopný pokračovať v tabuľkových výpočtoch sám.

Snažil som sa uviesť jednotlivé algoritmy v čo najjednoduchšej forme. Algoritmy na hľadanie najkratšej cesty v grafe podávam tak, aby bolo dobre vidieť, čo majú spoločné a čo rozdielne. Algoritmus na hľadanie cesty maximálnej kapacity a algoritmus na hľadanie záporného cyklu v grafe sú pôvodné (aspoň som sa s podobnou formuláciou doteraz nestretol). Pri väčšine algoritmov sa snažím o jednoduchý rozbor zložitosti v súlade so súčasným trendom v teórii grafov. Odporúčam študentom, najmä tým s informatickým zameraním, aby si väčšinu algoritmov naprogramovali sami. Niet lepšieho spôsobu na porozumenie ich činnosti.

Väčšinu kapitol dopĺňam časťou o praktických aplikáciách. Mnohé z nich sú originálne dopravné a spojárske aplikácie, niektoré z nich vznikli počas mojej výskumnej činnosti na Výskumnom ústave dopravnom v Žiline. Iné mám z literatúry, najmä z [4] a [9]. Záujemcov o ďalšie aplikácie teórie grafov odkazujem hlavne na knihu [9].

Náplň tohto učebného textu, terminológia, spôsob prezentácie algoritmov a mnohé ďalšie podrobnosti sa vyvíjali v dlhých diskusiách s RNDr. Štefanom Peškom, CSc., ktorému som vďačný za mnohé cenné nápady a pripomienky. Moja mimoriadna vďaka patrí RNDr. Milošovi Franekovi a recenzentom prof. RNDr. Jánovi Plesníkovi, doc. Gliviakovi, CSc. a vedeckému redaktorovi prof.

Petrovi Cenkovi, CSc., ktorí celý text pozorne prečítali a opravili množstvo formálnych i faktických chýb, niektoré z nich boli veľmi nepríjemné.

Teória grafov je užitočná. Ale nielen to. Teória grafov je aj krásna. Kto však chce vidieť krásu krajiny, musí sa namáhať a vyliezť na vysoký vrch. Podobne je to aj s každou matematickou disciplínou – kto chce vidieť jej krásu, musí sa ponamáhať. Výsledok však stojí za to. Zázitok z poznania a zázitok z netušených vlastných schopností sa nedá precítiť inak.

Prajem vám všetkým, aby ste pri štúdiu tejto knihy zažili veľa pekného.

V Žiline, 19. februára 2008

*Stanislav Palúch*  
Autor.

# Obsah

Úvod	3
<b>1 Základné pojmy teórie grafov</b>	<b>13</b>
1.1 Binárne relácie	13
1.2 Úvodné poznámky k terminológii	17
1.3 Grafy a digrafy	18
1.4 Rovnosť a izomorfizmus grafov	31
1.5 Reprezentácia grafov a digrafov	32
1.6 Aplikácie	41
1.6.1 Modelovanie reálnej dopravnej siete	41
1.6.2 Chemické grafy	42
1.6.3 Intelektuálne vlastníctvo počítačového čipu	43
1.7 Cvičenia	43
<b>2 Algoritmy a ich zložitosť</b>	<b>45</b>
2.1 Algoritmy	45
2.2 Úloha lineárneho programovania	48
2.3 Polynomiálne transformácie	50
2.4 NP-ťažké úlohy	53
2.5 Aproximácia	55
2.6 Heuristiky	56
2.7 Cvičenia	58
<b>3 Cesty v grafoch</b>	<b>59</b>

3.1	Sledy, ťahy a cesty v grafoch a digrafoch . . . . .	59
3.2	Súvislosť grafov . . . . .	63
3.3	Typy súvislosti digrafov . . . . .	64
3.4	Tarryho prieskum grafov . . . . .	65
3.5	Najkratšia cesta . . . . .	71
3.6	Výpočet matice vzdialeností . . . . .	83
3.7	Hľadanie cyklov zápornej ceny v digrafe . . . . .	90
3.8	Hľadanie cyklov zápornej ceny v grafe . . . . .	93
3.9	Cesta maximálnej spoľahlivosti . . . . .	94
3.10	Aplikácie . . . . .	95
3.10.1	Misionári a kanibali . . . . .	95
3.10.2	Jazdec na šachovnici . . . . .	96
3.10.3	Odmeriavanie vody . . . . .	98
3.10.4	Zalamovanie odstavca v typografii . . . . .	99
3.10.5	Elektronický cestovný poriadok . . . . .	101
3.11	Cvičenia . . . . .	103
<b>4</b>	<b>Acyklické grafy, stromy a kostry</b>	<b>105</b>
4.1	Stromy a ich vlastnosti . . . . .	105
4.2	Prehľadávanie grafu do hĺbky a do šírky . . . . .	110
4.3	Najlacnejšia a najdrahšia kostra . . . . .	113
4.4	Cesta maximálnej priepustnosti . . . . .	119
4.5	Aplikácie . . . . .	122
4.5.1	Štruktúra adresárov na disku . . . . .	122
4.5.2	Prefixové kódovanie . . . . .	123
4.5.3	Riešenie zložitých elektrických obvodov . . . . .	125
4.5.4	Úloha o elektrifikácii . . . . .	127
4.5.5	Plánovanie prepravy nadrozmerných nákladov . . . . .	128
4.6	Cvičenia . . . . .	129
<b>5</b>	<b>Acyklické digrafy</b>	<b>131</b>
5.1	Vlastnosti acyklických digrafov . . . . .	131
5.2	Extremálne cesty v acyklických digrafoch . . . . .	138



---

5.3	Metódy časového plánovania . . . . .	141
5.4	Klasická interpretácia metódy CPM . . . . .	152
5.5	Aplikácie . . . . .	156
5.5.1	Prioritný strom a halda . . . . .	156
5.6	Cvičenia . . . . .	159
<b>6</b>	<b>Pochôdzky v grafoch</b>	<b>161</b>
6.1	Eulerovské ťahy . . . . .	161
6.2	Úloha čínskeho poštára . . . . .	167
6.3	Úloha obchodného cestujúceho – TSP . . . . .	171
6.4	Ďalšie heuristiky pre TSP . . . . .	176
6.5	Aplikácie . . . . .	178
6.5.1	Sedem mostov mesta Königsberg . . . . .	179
6.5.2	Plánovanie športových stretnutí . . . . .	180
6.5.3	Riadenie súradnicového zapisovača . . . . .	181
6.5.4	DeBruijnovské postupnosti . . . . .	181
6.5.5	Miešačka farieb . . . . .	183
6.5.6	Optimálne poradie fáz v svetelne riadenej križovatke . . . . .	184
6.5.7	Grayov kód – Gray Code . . . . .	185
6.5.8	Vrtanie otvorov na plošných spojoch . . . . .	186
6.5.9	Švajčiarsky systém šachového turnaja . . . . .	187
6.6	Cvičenia . . . . .	188
<b>7</b>	<b>Toky v sieťach</b>	<b>189</b>
7.1	Siete a toky v sieťach . . . . .	189
7.2	Rezervná a zväčšujúca polocesta . . . . .	191
7.3	Rezové množiny . . . . .	193
7.4	Fordov–Fulkersonov algoritmus . . . . .	198
7.5	Siete s viacerými zdrojmi a ústiami . . . . .	201
7.6	Dolné medze pre tok . . . . .	202
7.7	Najlacnejší tok danej veľkosti . . . . .	205
7.8	Aplikácie . . . . .	211
7.8.1	Priraďovacia úloha . . . . .	211
7.8.2	Dopravná úloha . . . . .	213

7.8.3	Dopravná úloha s medziskladmi . . . . .	214
7.8.4	Optimalizácia turnusov v autobusovej doprave . . . . .	216
7.9	Cvičenia . . . . .	218
<b>8</b>	<b>Farbenie grafov</b>	<b>219</b>
8.1	Rovinné grafy . . . . .	220
8.2	Chromatické číslo a $k$ -zafarbiteľnosť . . . . .	225
8.3	Heuristiky pre farbenie grafu . . . . .	229
8.4	Exaktný algoritmus na farbenie grafov . . . . .	230
8.5	Aplikácie . . . . .	236
8.5.1	Priradenie registrov počítača . . . . .	236
8.5.2	Priradenie rádiových frekvencií . . . . .	236
8.5.3	Problém nákupných tašiek . . . . .	237
8.5.4	Rozvrhovanie voliteľných predmetov . . . . .	238
8.5.5	Fázovanie svetelne riadenej križovatky . . . . .	238
8.5.6	Minimalizácia počtu autobusových stanovišť . . . . .	241
8.6	Cvičenia . . . . .	243
<b>9</b>	<b>Niektoré ďalšie ťažké úlohy</b>	<b>245</b>
9.1	Centrá a mediány . . . . .	245
9.2	Kliky a maximálne nezávislé množiny . . . . .	250
9.3	Dominujúce množiny . . . . .	254
9.4	Aplikácie . . . . .	256
9.4.1	Znovu fázovanie svetelne riadenej križovatky . . . . .	256
9.4.2	Úlohy o koalíciách . . . . .	256
9.4.3	Ešte raz o havarijných strediskách . . . . .	257
9.4.4	Dámy na šachovnici . . . . .	258
9.4.5	Ústredne v komunikačnej sieti . . . . .	258
9.5	Cvičenia . . . . .	259
<b>A</b>	<b>Anglicko – slovenský slovníček</b>	<b>261</b>
	<b>Register</b>	<b>267</b>

*OBSAH*

11

---

**Literatúra**

**273**



# Kapitola 1

## Základné pojmy teórie grafov

### 1.1 Binárne relácie

Pre štúdium grafov sú mimoriadne dôležité vzťahy, do ktorých vstupujú dvojice objektov, nazývané tiež binárnymi reláciami. Preto v tejto časti zhrnieme niekoľko základných definícií a faktov o binárnych reláciách.

**Definícia 1.1.** Usporiadaná dvojica  $(u, v)$  prvkov  $u, v$  z množiny  $V$  je taká dvojica, pri ktorej je určené, ktorý z prvkov  $u, v$  je na prvom a ktorý na druhom mieste. Usporiadaná  $n$ -tica prvkov je taká  $n$ -tica prvkov  $(a_1, a_2, \dots, a_n)$ , pri ktorej je určené poradie prvkov.<sup>1</sup>

**Definícia 1.2.** Karteziánsky súčin  $A \times B$  dvoch množín  $A, B$  je množina všetkých usporiadaných dvojíc tvaru  $(a, b)$ , kde  $a \in A, b \in B$ . Karteziánsky súčin  $A_1 \times A_2 \times \dots \times A_n$  množín  $A_1, A_2, \dots, A_n$  je množina všetkých usporiadaných  $n$ -tíc  $(a_1, a_2, \dots, a_n)$ , kde  $a_i \in A_i$  pre  $i = 1, 2, \dots, n$ .

---

<sup>1</sup>Striktná teória množín definuje usporiadanú dvojicu ako množinu nasledovne:  $(u, v) = \{u, \{u, v\}\}$ . Čitateľ sa možno začuduje, prečo nedefinujeme usporiadanú dvojicu ako zobrazenie  $d$  množiny  $\{1, 2\}$  do množiny  $V$  – potom by  $d(1)$  bol prvý a  $d(2)$  druhý prvok usporiadanej dvojice  $d$ . Ako však uvidíme, zobrazenie sa definuje ako binárna relácia, a tá už k svojej definícii potrebuje mať definíciu usporiadanej dvojice. Bol by to postup definícií dokola. Pre naše účely však úplne stačí uvedené intuitívne chápanie usporiadanej  $n$ -tice.

Píšeme  $A^2 = A \times A$ . Podobne  $A^n$  je definované vzťahom  $A^n = \underbrace{A \times A \times \cdots \times A}_{n\text{-krát}}$ .

V matematike veľmi často potrebujeme vyjadriť skutočnosť, že dva prvky skúmanej množiny  $V$  sú v nejakom vzťahu – relácii. Ak skúmame množinu všetkých priamok v rovine, môžeme skúmať, či dve priamky  $p, q$  sú rovnobežné alebo nie. V kladnom prípade píšeme  $p \parallel q$ . Iný vzťah – kolmost dvoch priamok vyjadríme ako  $p \perp q$ . Pri skúmaní množiny prirodzených čísel  $\mathbb{N}$  môžeme študovať deliteľnosť čísel – ak je číslo  $n$  deliteľné číslom  $m$ , píšeme  $m|n$ .

Vzťah  $u \rho v$  je úplne a jednoznačne charakterizovaný množinou všetkých usporiadaných dvojíc  $(u, v)$ , pre ktoré platí  $u \rho v$ . To nás oprávňuje zaviesť nasledujúcu definíciu.

**Definícia 1.3.** Binárna relácia  $\rho$  na množine  $V$  je ľubovoľná podmnožina karteziánskeho súčinu  $V \times V$ .

Ak je  $\rho$  binárnou operáciou na  $V$  a  $(u, v) \in \rho$ , hovoríme, že **prvok  $u$  je v relácii  $\rho$  s prvkom  $v$**  a píšeme  $u \rho v$ .

**Príklad 1.1.** Príklady binárnych relácií na množine všetkých prirodzených čísel  $\mathbb{N}$ .

$$P = \{(n, k.n) \mid k, n \in \mathbb{N}\}, \quad Q = \{(m, n) \mid m, n \in \mathbb{N}, (\exists k \in \mathbb{N}) (m+k = n)\}$$

Potom  $a P b$  práve vtedy, keď  $a|b$ ,  $a Q b$  práve vtedy, keď  $a < b$ .

**Definícia 1.4.** Hovoríme, že binárna relácia  $\rho$  na množine  $V$  je  
**reflexívna na  $V$** , ak pre každé  $v \in V$  platí  $v \rho v$ ,  
**antireflexívna na  $V$** , ak pre žiadne  $v \in V$  neplatí  $v \rho v$ ,  
**symetrická na  $V$** , ak pre každé  $u, v \in V$  z platnosti  $u \rho v$  vyplýva platnosť  $v \rho u$ ,  
**tranzitívna na  $V$** , ak pre každé  $u, v, w \in V$  z platnosti  $u \rho v$  a  $v \rho w$  vyplýva  $u \rho w$ ,  
**antisymetrická na  $V$** , ak pre každé  $u, v \in V$  z platnosti  $u \rho v$  a  $v \rho u$  vyplýva  $u = v$ .

**Príklad 1.2.** Predstave o binárnych reláciách môže pomôcť tabuľka binárnej relácie  $\rho$ . Je to tabuľka, ktorá na mieste  $(a_i, b_j)$  má znak  $\bullet$  ak  $a_i \rho b_j$ , inak je toto miesto prázdne. Uvedieme tabuľky pre niekoľko binárnych relácií na množine  $V = \{1, 2, 3, 4, 5, 6\}$ .

	1	2	3	4	5	6
1	•					
2		•				
3			•			
4				•		
5					•	
6						•

a)

	1	2	3	4	5	6
1	•					
2	•	•				
3	•	•	•			
4	•	•	•	•		
5	•	•	•	•	•	
6	•	•	•	•	•	•

b)

	1	2	3	4	5	6
1			•		•	•
2		•	•		•	
3	•	•	•			
4						
5	•	•			•	
6	•					•

c)

Tabuľka a) je tabuľkou binárnej relácie  $u = v$ , tabuľka b) prislúcha relácii  $u \leq v$ , tabuľka c) je tabuľkou symetrickej binárnej relácie  $\rho$ , ktorá nie je reflexívna. Vidíme, že tabuľka reflexívnej relácie má prvkami • obsadenú celú hlavnú diagonálu, tabuľka antireflexívnej relácia má celú hlavnú diagonálu voľnú. Tabuľka symetrickej relácie je symetrická podľa svojej hlavnej diagonály, tabuľka antisymetrickej relácie nemá obsadené prvkami • žiadne dve rôzne políčka symetrické podľa hlavnej diagonály. Tranzitivitu binárnej relácie už tak ľahko na prvý pohľad z jej tabuľky nevidno.

**Definícia 1.5.** **Ekvivalencia** na množine  $V$  (alebo **relácia ekvivalencie** na množine  $V$ ) je binárna relácia na  $V$ , ktorá je reflexívna, symetrická a tranzitívna.

**Definícia 1.6.** **Rozkladom množiny**  $V$  nazveme systém neprázdnych podmnožín  $A_i \subseteq V$ ,  $i \in I$  taký, že  $A_i \cap A_j = \emptyset$  pre  $i, j \in I$ ,  $i \neq j$  a  $\bigcup_{i \in I} A_i = V$ . Množiny  $A_i$ ,  $i \in I$  nazveme **triedami rozkladu**.

**Definícia 1.7.** Nech  $P$  je relácia ekvivalencie na množine  $V$ , nech  $a \in V$ . **Triedou ekvivalencie**  $P$  k prvku  $a$  nazveme množinu

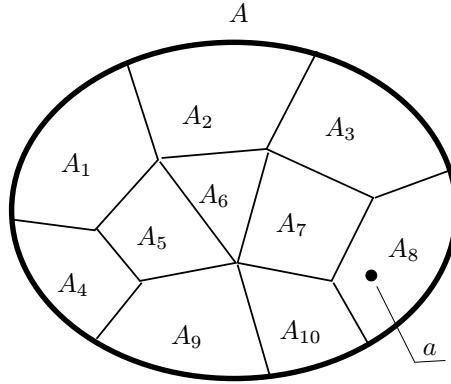
$$[a] = \{v \mid v \in V, v P a\}. \quad (1.1)$$

Prvok  $a$  nazveme reprezentantom triedy  $[a]$ .

**Veta 1.1.** Nech  $P$  je ekvivalencia na množine  $V$ . Potom všetky triedy ekvivalencie  $P$  tvoria rozklad množiny  $V$ .

**Veta 1.2.** Nech  $\mathcal{P} = \{A_i \mid i \in I\}$  je rozklad množiny  $V$ . Potom existuje práve jedna relácia ekvivalencie  $P$  taká, že triedy ekvivalencie  $P$  sú práve všetky triedy rozkladu  $\mathcal{P}$ .

**Príklad 1.3.** Rovnosť = prvkov akejkoľvek množiny je reflexívna, symetrická a tranzitívna relácia. Každá trieda ekvivalencie obsahuje práve jeden prvok.

Obr. 1.1: Rozklad množiny  $A$ .

Každá trieda rozkladu môže byť reprezentovaná svojím ľubovoľným prvkom  $a$ .

Vezmime za základnú množinu množinu  $\mathbb{Z}$  celých čísel a definujme binárnu reláciu  $P$  predpisom:

$$m P n \quad \text{práve vtedy, keď} \quad |m| = |n|. \quad (1.2)$$

Triedy ekvivalencie  $P$  sú  $[0] = \{0\}$ ,  $[1] = \{-1, 1\}$ ,  $[2] = \{-2, 2\}$ ,  $[3] = \{-3, 3\}$ ,  $\dots$

**Príklad 1.4.** Nech  $V$  je množina, nech  $H = V \times V$  je množina všetkých usporiadaných dvojíc prvkov z  $V$ . Na množine  $H$  zavedieme reláciu ekvivalencie  $\equiv$  predpisom:

$$(u, v) \equiv (x, y) \quad \text{práve vtedy, keď} \quad (u = x \text{ a } v = y) \text{ alebo } (u = y \text{ a } v = x). \quad (1.3)$$

Pomocou rovností usporiadaných dvojíc možno vzťah (1.3) preformulovať nasledovne

$$(u, v) \equiv (x, y) \quad \text{práve vtedy, keď} \quad (u, v) = (x, y) \text{ alebo } (u, v) = (y, x). \quad (1.4)$$

Relácia  $\equiv$  je ekvivalencia na množine  $H$ , jej triedy sú tvaru  $[(v, v)] = \{(v, v)\}$  alebo  $[(u, v)] = \{(u, v), (v, u)\}$  pre  $u \neq v$ . Triedy ekvivalencie  $\equiv$  nazveme **neusporiadanými dvojicami** prvkov z množiny  $V$ .

Triedu ekvivalencie typu  $[(u, v)]$ , kde  $u \neq v$ , možno stotožniť s dvojprvkovou množinou  $\{u, v\}$ . Triedu ekvivalencie typu  $[(u, u)]$ , možno stotožniť s jedno-prvkovou množinou  $\{u, u\} = \{u\}$ . Preto budeme písať  $\{u, v\}$  namiesto  $[(u, v)]$ . Množinu všetkých neusporiadaných dvojíc prvkov z  $V$  budeme značiť  $V \circ V$ .



**Definícia 1.8.** Hovoríme, že relácia  $\preceq$  je **usporiadanie** na množine  $V$ , ak  $\preceq$  je reflexívna, tranzitívna a antisymetrická relácia. Dvojicu  $(V, \preceq)$ , kde  $\preceq$  je usporiadanie na  $V$ , voláme **čiasťočne usporiadaná množina**.

Nech  $\preceq$  je usporiadanie na množine  $V$ , nech pre prvky  $u \in V$ ,  $v \in V$  neplatí ani  $u \preceq v$ , ani  $v \preceq u$ . Potom hovoríme, že prvky  $u, v$  sú **neporovnateľné**.

Ak pre  $u \in V$ ,  $v \in V$  platí  $u \preceq v$  alebo  $v \preceq u$ , hovoríme, že prvky  $u, v$  sú **porovnateľné**.

**Lineárne usporiadanie** je také usporiadanie  $\preceq$  na  $V$ , že pre každú dvojicu prvkov  $u, v \in V$  je  $u \preceq v$  alebo  $v \preceq u$ . Dvojicu  $(V, \preceq)$ , kde  $P$  je lineárne usporiadanie na  $V$ , voláme **lineárne usporiadaná množina** alebo **reťazec**.

**Príklad 1.5.** Relácia inklúzie  $\subseteq$  je usporiadaním na množine všetkých podmnožín nejakej základnej množiny  $A$ . Ak má množina  $A$  aspoň dva rôzne prvky, potom existujú dve neprázdne disjunktné podmnožiny množiny  $A$ , ktoré sú neporovnateľné v usporiadaní  $\subseteq$ . Usporiadanie  $\subseteq$  preto nie je lineárnym usporiadaním.

**Príklad 1.6.** Relácia  $n \leq k$  je lineárnym usporiadaním na množine všetkých celých čísel  $\mathbb{Z}$ .

## 1.2 Úvodné poznámky k terminológii teórie grafov

Cieľom tejto kapitoly je definovať základné pojmy používané v teórii grafov. Terminológia teórie grafov však nie je ustálená ani v anglickej ani v slovenskej literatúre. Tak napríklad niektoré anglické zdroje používajú termín *graph* pre najvšeobecnejšiu grafovú štruktúru (u nás to bude pseudomigraf), potom pre štruktúru, ktorá je pre nás grafom majú termín *simple graph*. Pre iné zdroje je *graph* definovaný rovnako, ako v definícii 1.9.

V slovenčine sa pre orientovanú hranu používajú aj termíny oblúk, či šíp. Podobne rovinný graf – planárny graf, úplný graf – kompletný graf, atď.

Pre túto publikáciu som zvolil terminológiu, ktorá je kompromisom medzi Plesníkovou terminológiou z knihy [13] a tradičnou terminológiou používanou v inžinierskych predmetoch na Fakulte riadenia a informatiky Žilinskej univerzity.

Upozorňujem čitateľa, že pri štúdiu akejkoľvek literatúry používajúce teóriu grafov je nutné ozrejmiť si v akom zmysle táto literatúra používa grafovú

terminológiu. Naopak, pri písaní vlastnej práce je vhodné uviesť buď definície základných používaných pojmov, alebo radšej uznávaný zdroj, z ktorého terminologický systém práca používa.

### 1.3 Grafy a digrafy

**Definícia 1.9. Grafom** nazveme usporiadanú dvojicu  $G = (V, H)$ , kde  $V$  je neprázdna konečná množina a  $H$  je množina neusporiadaných dvojíc typu  $\{u, v\}$  takých, že  $u \in V$ ,  $v \in V$  a  $u \neq v$ , t. j.

$$H \subseteq \{\{u, v\} \mid u \neq v, u, v \in V\} \subset V \circ V. \quad (1.5)$$

Prvky množiny  $V$  nazývame **vrcholmi** a prvky množiny  $H$  **hranami** grafu  $G$ .

**Definícia 1.10. Digrafom** nazveme usporiadanú dvojicu  $\vec{G} = (V, H)$ , kde  $V$  je neprázdna konečná množina a  $H$  je množina usporiadaných dvojíc typu  $(u, v)$  takých, že  $u \in V$ ,  $v \in V$  a  $u \neq v$ , t. j.

$$H \subseteq \{(u, v) \mid u \neq v, u, v \in V\} \subset V \times V. \quad (1.6)$$

Prvky množiny  $V$  nazývame **vrcholmi** a prvky množiny  $H$  **orientovanými hranami** digrafu  $\vec{G}$ .

V slovenskej literatúre sa pre orientovanú hranu používa tiež termín **šíp** (napríklad v literatúre [13]) alebo tiež **oblúk** (z anglického termínu **arc**), Plesník v [13] používa pre neorientovanú hranu termín **rebro**.

Ak nebude hroziť nebezpečenstvo nedorozumenia, budeme v prípade digrafof namiesto termínu „orientovaná hrana digrafu“ používať skrátený termín „hrana digrafu“.

Majme graf  $G = (V, H)$ , resp. digraf  $\vec{G} = (V, H)$ . Množina  $V$  sa volá **vrcholová množina** grafu  $G$ , resp. digrafu  $\vec{G}$ , množina  $H$  sa volá **hranová množina** grafu  $G$ , resp. digrafu  $\vec{G}$ . V literatúre o teórii grafov sa používa aj zápis  $V_G$ ,  $H_G$  pre vrcholovú a hranovú množinu grafu  $G$ , čím sa špecifikuje, že  $G = (V_G, H_G)$ .

Graf a digraf sú najjednoduchšie grafové štruktúry, v ktorých nie sú dovolené hrany typu  $\{v, v\}$ , resp.  $(v, v)$ , nazývané tiež **slučky**. V grafe, resp. digrafe môže pre každú dvojicu  $u, v \in V$  existovať najviac jedna hrana typu  $\{u, v\}$ , resp.  $(u, v)$ .

Poznamenajme, že v grafe je hrana  $\{u, v\}$  totožná s hranou  $\{v, u\}$ , kým v digrafe  $(u, v)$  a  $(v, u)$  sú rôzne orientované hrany.

Na digraf  $\vec{G}(V, H)$  sa možno pozerat' i nasledujúcim spôsobom:

Keďže  $H \subseteq V \times V$ , digraf je vlastne množina s antireflexívnou binárnou reláciou. Trošku zložitejšia je takáto interpretácia grafu  $G = (V, H)$ . Množina  $H$  neusporiadaných dvojíc z  $V$  tu jednoznačne korešponduje so symetrickou antireflexívnou binárnou reláciou  $\rho$  na množine  $V$ , pre ktorú platí  $u \rho v$  práve vtedy, keď  $\{u, v\} \in H$ .

**Definícia 1.11. Diagram grafu.** Graf často reprezentujeme graficky a príslušný obrázok voláme diagram grafu. **Diagram grafu**  $G = (V, H)$  v nejakom priestore  $\mathcal{P}$  je množina  $B$  bodov a množina  $S$  súvislých čiar v priestore  $\mathcal{P}$  takých, že

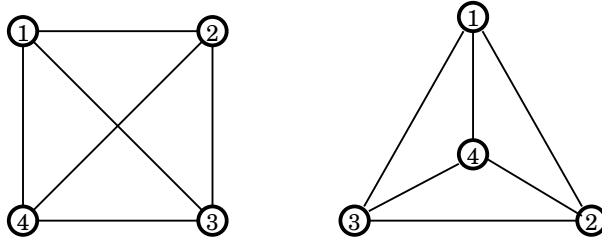
- Každému vrcholu  $v \in V$  zodpovedá práve jeden bod  $b_v \in B$  a každému bodu  $b \in B$  zodpovedá práve jeden vrchol  $v \in V$  (t. j.  $b = b_v$ ), pričom pre  $u, v \in V$ ,  $u \neq v$  je  $b_u \neq b_v$ .
- Každdej hrane  $h \in H$  zodpovedá práve jedna čiara  $s_h \in S$  a každej čiare  $s \in S$  zodpovedá práve jedna hrana  $h \in H$  (t. j.  $s = s_h$ ), pričom pre  $h, k \in H$ ,  $h \neq k$  je  $s_h \neq s_k$ .
- Ak  $h = \{u, v\} \in H$ , potom čiara  $s_h$  má koncové body  $b_u, b_v$ . Okrem koncových bodov žiadna čiara neobsahuje žiaden bod typu  $b_w \in B$ .
- Navyiac sa často žiada, aby bol diagram nakreslený tak, že žiadna čiara samu seba nepretína a dve čiary majú najviac jeden priesečník.

Veľmi často sa za priestor  $\mathcal{P}$  berie rovina. Skúmajú sa však aj diagramy grafov na guľovej ploche, anuloide, Möbiovej ploche či v trojrozmernom Euklidovskom priestore.

Podobne ako diagram grafu možno definovať diagram digrafu, ak namiesto čiar použijeme orientované čiary, resp. šípky.

**Definícia 1.12.** Diagram grafu, resp. digrafu v rovine nazveme **rovinný**, ak sa jeho hrany nepretínajú nikde inde okrem vrcholov. Graf  $G = (V, H)$ , resp. digraf  $\vec{G} = (V, H)$  nazveme **rovinný**, ak k nemu existuje rovinný diagram.

*Poznámka.* V niektorej slovenskej literatúre sa namiesto termínu rovinný graf používa termín **planárny graf**, čo je pravdepodobne ovplyvnené anglickým **planar graph**.



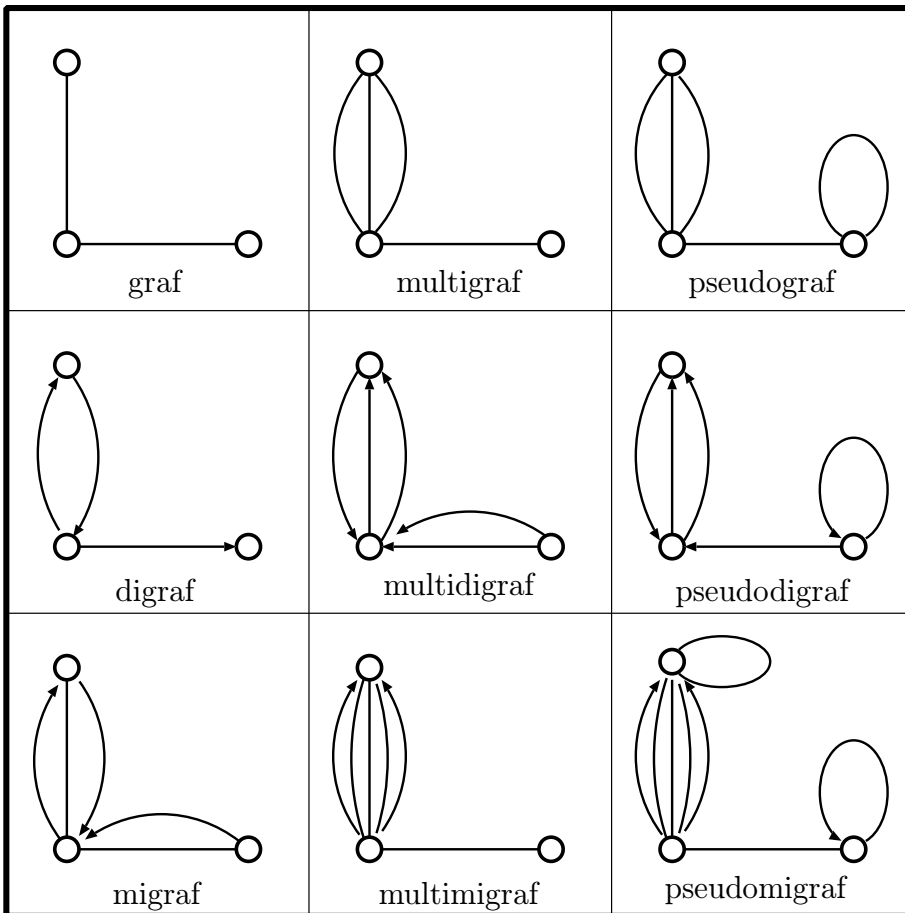
Obr. 1.2: Dva diagramy toho istého grafu  $G = (V, H)$ ,  
kde  $V = \{1, 2, 3, 4\}$ ,  $H = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$ .

V teórii grafov treba striktno rozlišovať medzi grafom a jeho diagramom. Graf je dvojica množín vrcholov a hrán, diagram grafu je „obrázok“, ktorý istým spôsobom korešponduje s príslušným grafom. Ak je daný diagram grafu, existuje k nemu jediný príslušný graf. K jednému grafu však možno nakresliť ľubovoľné množstvo rôznych diagramov, o ktorých na prvý pohľad ani nemusí byť zrejmé, že sú diagramami toho istého grafu. Diagram obsahuje oveľa viac informácií ako príslušný graf - sú to napr. súradnice jednotlivých bodov, tvar spojnic atď. Na obrázku 1.2 sú dva veľmi odlišné diagramy toho istého grafu. Pritom ľavý diagram nie je rovinný, ale pravý diagram rovinný je. Pretože existuje ku grafu  $G$  rovinný diagram, je príslušný graf rovinný.

Teória grafov však študuje i zložitejšie útvary ako je graf a digraf. V týchto štruktúrach sú dovolené viacnásobné hrany, slučky a dokonca i oba typy hrán naraz. Pre tieto štruktúry už definície grafu a digrafu nepostačujú, a preto sa používajú zložitejšie matematické modely. Ukážky ich diagramov sú na obrázku 1.3. Keď sa s nimi stretne, budeme ich chápať intuitívne. Presnú definíciu väčšiny týchto pojmov nájde čitateľ na tejto strane pod čiarou.<sup>2</sup>

<sup>2</sup>**Pseudomigraf** je usporiadaná trojica  $G = (V, H, \phi)$ , kde  $V$  je neprázdna konečná množina vrcholov,  $H$  je konečná množina hrán,  $V \cap H = \emptyset$  a  $\phi$  je zobrazenie  $\phi : H \rightarrow V \circ V \cup V \times V$ . V poslednom vzťahu  $V \circ V$  znamená množinu všetkých neusporiadaných dvojíc prvkov z  $V$  a  $V \times V$  je obvyklý karteziansky súčin - t. j. množina všetkých usporiadaných dvojíc prvkov z  $V$ .

Všetky ostatné štruktúry sú špeciálnymi prípadmi pseudomigrafu. Ak hranová množina  $H$  pseudomigrafu  $G$  neobsahuje slučky, hovoríme, že  $G$  je **multimigraf**. Ak  $\phi : H \rightarrow V \circ V$ , ide o **pseudograf**, ak sú navyše zakázané slučky máme **multigraf**. Podobne v orientovanom prípade, ak  $\phi : H \rightarrow V \times V$ , ide o **pseudodigraf**, ak hranová množina  $H$  neobsahuje slučky, máme **multidigraf**.



Obr. 1.3: Diagramy všeobecných grafových štruktúr.  
 (Obrázok je prevzatý z Plesníkovej knihy [13] s láskavým zvoľením autora.)

**Definícia 1.13.** Hovoríme, že graf  $G' = (V', H')$  je **podgrafom grafu**  $G = (V, H)$ , ak platí  $V' \subseteq V$  a  $H' \subseteq H$ . V tomto prípade budeme písať  $G' \subseteq G$ . Digraf  $\vec{G}' = (V', H')$  je podgrafom digrafu  $\vec{G} = (V, H)$ , ak  $V' \subseteq V$  a  $H' \subseteq H$ .

**Definícia 1.14.** Hovoríme, že graf  $G' = (V', H')$  je **faktorovým podgrafom** grafu  $G = (V, H)$ , ak platí  $V' = V$  a  $H' \subseteq H$ . Analogicky definujeme **faktorový podgraf digrafu**  $\vec{G}$ .

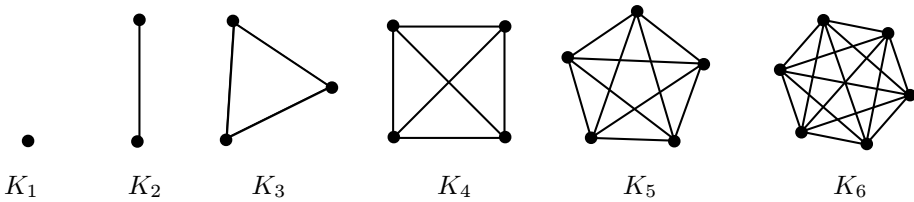
Nech  $G = (V, H)$  je graf. Ak pre štruktúru  $G' = (V', H')$  platí  $V' \subseteq V$ ,  $H' \subseteq H$ , ešte nemusí byť  $G'$  podgrafom grafu  $G$ .

**Príklad 1.7.**  $G = (\{1, 2, 3\}, \{\{1, 2\}, \{1, 3\}\})$ ,  $G' = (\{1, 2\}, \{\{1, 3\}\})$ .  $G'$  totiž nie je vôbec graf, lebo hrana  $\{1, 3\}$  nie je dvojicou prvkov z množiny  $\{1, 2\}$ .

**Definícia 1.15.** Graf  $G = (V, H)$  nazveme **úplným**, ak množina  $H$  obsahuje všetky možné dvojice typu  $\{u, v\}$ , kde  $u, v \in V$  a  $u \neq v$ . Úplný graf o  $n$  vrcholoch budeme značiť  $K_n$ .

Podobne digraf  $\vec{G} = (V, H)$  nazveme **úplným**, ak množina  $H$  obsahuje všetky možné dvojice typu  $(u, v)$ , kde  $u, v \in V$  a  $u \neq v$ .

*Poznámka.* Niektorá literatúra používa namiesto termínu **úplný graf** termín **kompletný graf**.



Obr. 1.4: Diagramy úplných grafov  $K_1$  až  $K_6$ .

**Definícia 1.16.** **Maximálny podgraf**  $G'$  grafu  $G$  s **nejakou vlastnosťou**  $\mathcal{V}$  je taký podgraf grafu  $G$ , ktorý má vlastnosť  $\mathcal{V}$ , a pritom neexistuje podgraf  $G''$  grafu  $G$  s vlastnosťou  $\mathcal{V}$  taký, že  $G' \subseteq G''$  a  $G' \neq G''$ .

**Minimálny podgraf**  $G'$  grafu  $G$  s **vlastnosťou**  $\mathcal{V}$  je taký podgraf grafu  $G$ , ktorý má vlastnosť  $\mathcal{V}$ , a pritom neexistuje podgraf  $G''$  grafu  $G$  s vlastnosťou  $\mathcal{V}$  taký, že  $G'' \subseteq G'$  a  $G'' \neq G'$ .

**Definícia 1.17.** Nech  $G = (V, H)$  je graf (digraf),  $V' \subseteq V$ . Hovoríme, že  $G'$  je **podgraf grafu (digrafu)  $G$  indukovaný množinou vrcholov  $V'$** , ak  $G'$  je maximálny podgraf grafu  $G$  s množinou vrcholov  $V'$ .

Nech  $H' \subseteq H$ . Hovoríme, že  $G'$  je **podgraf grafu (digrafu)  $G$  indukovaný množinou hrán  $H'$** , ak  $G'$  je minimálny podgraf grafu  $G$  s množinou hrán  $H'$ .

**Definícia 1.18.** Nech  $G = (V, H)$  je graf, resp. digraf,  $v \in V$ ,  $h \in H$ . Vrchol  $v$  je **incidentný s hranou  $h$** , ak je  $v$  jedným z vrcholov hrany  $h$ . Hrany  $h, k \in H$ ,  $h \neq k$  sú **prilahlé** alebo **susedné**, ak majú spoločný jeden vrchol. Vrcholy  $u, v$ , sú **prilahlé** alebo **susedné**, ak  $\{u, v\} \in H$ , t. j. ak  $\{u, v\}$  je hranou, resp. ak  $(u, v) \in H$  alebo  $(v, u) \in H$ .

Symbolom  $H(v)$  budeme označovať množinu všetkých hrán grafu  $G$  incidentných s vrcholom  $v$ , symbolom  $V(v)$  budeme označovať množinu všetkých vrcholov prilahlých k vrcholu  $v$ .

Nech  $\vec{G} = (V, H)$  je digraf,  $u \in V$ ,  $v \in V$ ,  $h \in H$ . Hovoríme, že orientovaná hrana  $h$  **vychádza z vrchola  $u$** , alebo že **vrchol  $u$  je začiatočný vrchol orientovanej hrany  $h$** , ak  $h = (u, x)$  pre niektoré  $x \in V$ . Hovoríme, že orientovaná hrana  $h$  **vchádza do vrchola  $v$** , alebo že **vrchol  $v$  je koncový vrchol orientovanej hrany  $h$** , ak  $h = (y, v)$  pre niektoré  $y \in V$ . Orientovaná hrana  $h$  je **incidentná** s vrcholom  $v$ , ak hrana  $h$  vchádza do vrchola  $v$  alebo vychádza z vrchola  $v$ .

Symbolom  $H^+(v)$  budeme označovať množinu všetkých orientovaných hrán digrafu  $\vec{G}$  vychádzajúcich z vrchola  $v$ , symbolom  $H^-(v)$  budeme označovať množinu všetkých orientovaných hrán digrafu  $\vec{G}$  vchádzajúcich do vrchola  $v$ ,  $H(v) = H^+(v) \cup H^-(v)$  je množina všetkých incidentných hrán s vrcholom  $v$ . Symbolom  $V^+(v)$  budeme označovať množinu koncových vrcholov všetkých hrán z  $H^+(v)$ , symbolom  $V^-(v)$  množinu začiatočných vrcholov všetkých hrán z  $H^-(v)$ . Platí  $V(v) = V^+(v) \cup V^-(v)$ .

Pomocou pojmov z predchádzajúcej definície môžeme špecifikovať podgrafy grafu indukované množinou vrcholov, resp. hrán aj nasledujúco.

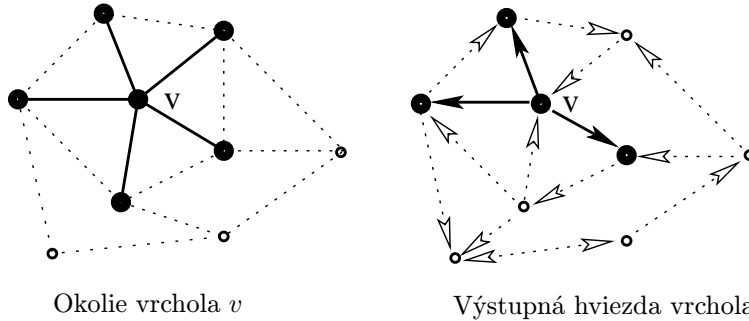
**Definícia 1.19.** Podgraf grafu  $G = (V, H)$  indukovaný množinou vrcholov  $V' \subseteq V$  je graf  $G' = (V', H')$ , kde  $H' = H \cap (V' \circ V')$ .

Podgraf digrafu  $\vec{G} = (V, H)$  indukovaný množinou vrcholov  $V' \subseteq V$  je digraf  $\vec{G}' = (V', H')$ , kde  $H' = H \cap (V' \times V')$ .

Podgraf grafu (digrafu)  $G$  indukovaný množinou hrán  $H' \subseteq H$  má za množinu vrcholov  $V'$  práve množinu všetkých vrcholov incidentných s hranami z  $H'$  (a žiadne ďalšie).

**Definícia 1.20.** Nech  $G = (V, H)$  je graf alebo digraf,  $v \in V$ . **Okolím vrchola**  $v$  nazveme graf, resp. digraf  $O(v) = (V(v) \cup \{v\}, H(v))$ , t. j. ktorého vrcholová množina pozostáva z vrchola  $v$  a všetkých s ním susedných vrcholov a ktorého hranová množina je množinou všetkých hrán incidentných s vrcholom  $v$ .

Nech  $\vec{G} = (V, H)$  je digraf,  $v \in V$ . **Výstupnou hviezdou vrchola**  $v$  nazveme digraf  $Fstar(v) = (V^+(v) \cup \{v\}, H^+(v))$ , ktorého vrcholová množina pozostáva z vrchola  $v$  a koncových vrcholov všetkých hrán vychádzajúcich z vrchola  $v$  a hranová množina je množinou všetkých hrán vychádzajúcich z vrchola  $v$ . **Vstupnou hviezdou vrchola**  $v$  nazveme digraf  $Bstar(v) = (V^-(v) \cup \{v\}, H^-(v))$ , ktorého vrcholová množina pozostáva z vrchola  $v$  a začiatkových vrcholov všetkých hrán vchádzajúcich do vrchola  $v$  a ktorého hranová množina je množinou všetkých hrán vchádzajúcich do vrchola  $v$ .



Obr. 1.5: Okolie a výstupná hviezda vrchola  $v$  sú vyznačené hrubou čiarou.

**Dohoda 1.1.** Často budeme pracovať s grafom, resp. digrafom  $G' = (V', H')$ , ktorý vznikol z grafu  $G = (V, H)$  vylúčením niektorého vrchola  $v \in V$  a všetkých hrán jeho okolia. Takýto graf  $G'$  budeme značiť  $G - \{v\}$ .

Ak je  $h \in H$  potom  $G - \{h\}$  bude označovať graf  $G' = (V', H')$ , kde  $V' = V$  a  $H' = H - \{h\}$ . Ak  $x \notin V$ , potom  $G \cup \{x\}$  bude znamenať graf s množinou vrcholov  $V \cup \{x\}$  a množinou hrán  $H$ . Podobne, ak  $h \in V \circ V$ ,  $h = \{u, v\}$ ,  $u \neq v$ , resp.  $h \in V \times V$ ,  $h = (u, v)$ ,  $u \neq v$ , potom  $G \cup \{h\}$  je grafom, resp. digrafom s množinou vrcholov  $V$  a množinou hrán  $H' = H \cup \{h\}$ .



Majme graf  $G = (V, H)$ . Ak nedôjde k nedorozumeniu, použijeme niekedy skrátene vrchol  $v \in G$  alebo hrana  $h \in G$  namiesto  $v \in V$  alebo  $h \in H$ . Toto označenie sa nám bude hodiť najmä v situáciách, kedy nebudeme mať špecifikovanú vrcholovú alebo hranovú množinu grafu  $G$ . V tejto situácii využijeme symbol  $V_G$  pre vrcholovú, resp.  $H_G$  pre hranovú množinu grafu  $G$ .

**Definícia 1.21. Stupeň  $\deg(v)$  vrchola  $v$  v grafe  $G = (V, H)$  je počet hrán incidentných s vrcholom  $v$ .**

**Výstupný stupeň  $\text{odeg}(v)$  vrchola  $v$  v digrafe  $\vec{G} = (V, H)$  je počet hrán digrafu  $\vec{G}$  z vrchola  $v$  vychádzajúcich.**

**Vstupný stupeň  $\text{ideg}(v)$  vrchola  $v$  v digrafe  $\vec{G}$  je počet hrán digrafu  $\vec{G}$  do vrchola  $v$  vchádzajúcich.**

*Poznámka.* Definíciu 1.21 možno rozšíriť i na multigrafy a multidigrafy. Pri použití označení zavedených v definícii 1.18 platí:

$$\deg(v) = |H(v)|, \quad \text{ideg}(v) = |H^-(v)|, \quad \text{odeg}(v) = |H^+(v)|,$$

kde zápis  $|M|$  znamená počet prvkov množiny  $M$ .

Viacere matematické zdroje (napríklad [9]) uvádzajú nasledujúcu vetu ako prvú matematickú vetu v štúdiu teórie grafov.

**Veta 1.3. (Euler.) Súčet stupňov všetkých vrcholov v grafe  $G = (V, H)$  sa rovná dvojnásobku počtu hrán grafu  $G$ , t. j.**

$$\sum_{v \in V} \deg(v) = 2 \cdot |H|. \quad (1.7)$$

**DŮKAZ.**

Každá hrana grafu  $G$  je incidentná s práve dvoma vrcholmi (totiž so svojimi krajnými vrcholmi). Preto každá hrana prispieva do celkového súčtu stupňov vrcholov  $\sum_{v \in V} \deg(v)$  číslom 2. Preto je

$$\sum_{v \in V} \deg(v) = 2 \cdot |H|. \quad (1.8)$$

■

**Dôsledok 1.1.** *Nech  $G = (V, H)$  je graf,  $n = |V|$ . Potom*

$$\sum_{v \in V} \deg(v) \leq n(n-1). \quad (1.9)$$

DŔKAZ.

Počet hrán  $|H|$  grafu  $G$  je menší alebo rovný počtu hrán úplného grafu  $K_n$ . Počet hrán úplného grafu je rovný počtu všetkých kombinácií dvoch prvkov z  $n$  prvkov, t. j.  $\binom{n}{2}$ . Máme teda

$$|H| \leq \binom{n}{2} = \frac{n(n-1)}{2},$$

čo spolu s (1.7) dáva (1.9). ■

Ďalším dôsledkom Eulerovej vety je nasledujúce tvrdenie.

**Veta 1.4.** *Počet vrcholov nepárneho stupňa v ľubovoľnom grafe  $G = (V, H)$  je párnny.*

DŔKAZ.

Označme  $V_1$  podmnožinu všetkých vrcholov množiny  $V$  s nepárnym stupňom. Potom  $V_2 = V - V_1$  obsahuje všetky vrcholy z  $V$  párneho stupňa. Použitím (1.7) môžeme písať

$$\sum_{v \in V} \deg(v) = \sum_{v \in V_1 \cup V_2} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v) = 2 \cdot |H|, \quad (1.10)$$

a teda

$$\sum_{v \in V_1} \deg(v) = 2 \cdot |H| - \sum_{v \in V_2} \deg(v). \quad (1.11)$$

Na pravej strane od párneho čísla  $2|H|$  odčítavame súčet párných čísel ( $V_2$  bolo definované ako množina vrcholov párneho stupňa), preto musí byť aj súčet na ľavej strane vzťahu (1.11) párne číslo.  $\sum_{v \in V_1} \deg(v)$  je súčet istého počtu  $k$  nepárných čísel. Nech  $V_1 = \{v_1, v_2, \dots, v_k\}$ , nech  $k$  je nepárne číslo. Potom

$$\begin{aligned} \sum_{v \in V_1} \deg(v) &= \underbrace{(\deg(v_1) + \deg(v_2))}_{\text{párne}} + \underbrace{(\deg(v_3) + \deg(v_4))}_{\text{párne}} + \dots \\ &\quad \dots + \underbrace{(\deg(v_{k-2}) + \deg(v_{k-1}))}_{\text{párne}} + \underbrace{\deg(v_k)}_{\text{nepárne}} \end{aligned} \quad (1.12)$$

Pretože v množine  $V_1$  sú vrcholy nepárneho stupňa (tak totiž ona bola definovaná), vo vzťahu (1.12) je súčet ľubovoľných dvoch členov párnny. Z posledného vzťahu je však vidieť, že pre nepárne  $k$  je jeho pravá strana nepárna. Keďže  $\sum_{v \in V_1} \deg(v)$  je párne číslo,  $|V_1| = k$  musí byť len párne číslo. ■

**Veta 1.5.** *Nech  $G = (V, H)$  je netriviálny graf (t. j.  $|V| \geq 2$ ). Potom  $V$  obsahuje aspoň dva vrcholy rovnakého stupňa.*

DŮKAZ.

Nech  $|V| = n$ . Stupne vrcholov v grafe  $G$  môžu nadobúdať len hodnoty  $0, 1, \dots, n - 1$  – t. j.  $n$  hodnôt. Avšak medzi týmito hodnotami nemôže byť súčasne aj  $0$ , aj  $n - 1$ . Ak by totiž existoval vrchol  $v$  stupňa  $n - 1$ , potom by musel byť susedný so všetkými ostatnými vrcholmi – t. j. žiaden vrchol by nemohol mať stupeň  $0$ . V jednom grafe s  $n$  vrcholmi stupne vrcholov môžu teda nadobúdať nanaajvyš  $n - 1$  hodnôt. Keďže hodnôt stupňov vrcholov je menej ako vrcholov, aspoň dva vrcholy musia mať rovnaký stupeň. ■

Dôkaz predchádzajúcej vety je pekným príkladom použitia tzv. princípu holubích hniezd (pigeonhole principle), ktorý hovorí, že ak je menej hniezd ako holubov, potom aspoň dva holuby musia obývať to isté hniezdo. Tento princíp je známy aj ako Dirichletov princíp, kde sa hovorí o priehradkách (namiesto hniezd) a do nich vkladáných objektoch (namiesto holubov).

**Definícia 1.22.** Nech  $G = (V, H)$  je graf,  $n = |V|$ . Nech  $\mathbf{v} = (v_1, v_2, \dots, v_n)$  je postupnosť práve všetkých vrcholov grafu  $G$ . Postupnosť

$$\mathbf{p} = (\deg(v_1), \deg(v_2), \dots, \deg(v_n)) \quad (1.13)$$

nazveme **valenčnou postupnosťou grafu  $G$** .

**Neklesajúca valenčná postupnosť** grafu  $G$  je  $n$ -prvková neklesajúca postupnosť, ktorá vznikne usporiadaním postupnosti (1.13) neklesajúco.

Valenčná postupnosť triviálneho grafu je jednoprvková a obsahuje iba nulu. Valenčná postupnosť úplného grafu  $K_5$  je  $(4, 4, 4, 4, 4)$ . Graf, ktorého diagram je na obrázku 1.7 na strane 30 prvý zľava má túto neklesajúcu valenčnú postupnosť:  $(1, 2, 2, 3, 4)$ . Neklesajúca valenčná postupnosť druhého grafu z obrázku 1.7 je  $(0, 1, 2, 2, 3)$ .

Čo doteraz vieme o valenčnej postupnosti grafu? Každý člen  $n$ -prvkovej valenčnej postupnosti grafu musí byť menší alebo rovný než  $n - 1$ . Podľa vety 1.3 súčet prvkov valenčnej postupnosti je rovný dvojnásobku počtu hrán – teda musí byť párný a podľa dôsledku 1.1 (str. 25) menší alebo nanaajvyš rovný číslu  $n \cdot (n - 1)$ . Podľa vety 1.4 musí byť počet nepárnych členov valenčnej postupnosti párný a podľa vety 1.5 musí valenčná postupnosť obsahovať aspoň dva rovnaké prvky. Podľa dôkazu vety 1.5 súčasný výskyt čísel  $0$  a  $n - 1$  v jednej valenčnej postupnosti je vylúčený. Nasledujúca veta dáva algoritmus na zistenie, či je daná  $n$  prvková postupnosť valenčnou postupnosťou nejakého grafu:

**Veta 1.6. (Havel.)** *Neklesajúca  $n$ -prvková postupnosť  $\mathbf{p}_n$*

$$\mathbf{p}_n = (d_1, d_2, \dots, d_{n-d_n-1}, \underbrace{d_{n-d_n}, d_{n-d_n+1}, d_{n-d_n+2}, \dots, d_{n-1}, d_n}_{\text{predposledných } d_n \text{ členov postupnosti } \mathbf{p}_n}),$$

kde  $0 \leq d_i \leq n-1$  pre  $i = 1, 2, \dots, n$ , je valenčnou postupnosťou pre nejaký graf  $G$  práve vtedy, keď  $(n-1)$ -prvková postupnosť  $\mathbf{p}_{n-1}$  definovaná ako

$$\mathbf{p}_{n-1} = (d_1, d_2, \dots, d_{n-d_n-1}, \underbrace{(d_{n-d_n} - 1), (d_{n-d_n+1} - 1), (d_{n-d_n+2} - 1), \dots, (d_{n-1} - 1)}_{\text{predposledných } d_n \text{ členov postupnosti } \mathbf{p}_n \text{ zmenšených o } 1})$$

je valenčnou postupnosťou nejakého  $(n-1)$ -vrcholového grafu  $G'$ .

**DŮKAZ.**

Nech postupnosť  $\mathbf{p}_{n-1}$  je valenčnou postupnosťou nejakého grafu  $G' = (V', H')$ . Pridaním vrchola  $v_n$  takého, že  $v_n \notin V'$ , k množine  $V'$  dostaneme množinu  $V = V' \cup \{v_n\}$ . Vytvorme množinu hrán  $H$  tak, že k hranám množiny  $H'$  pridáme všetkých  $d_n$  dvojíc typu  $\{v_i, v_n\}$ , kde  $v_i$  prebieha posledných  $d_n$  vrcholov grafu  $G'$  príslušných postupnosti  $\mathbf{p}_{n-1}$ . Potom graf  $G = (V, H)$  má valenčnú postupnosť  $\mathbf{p}_n$ .

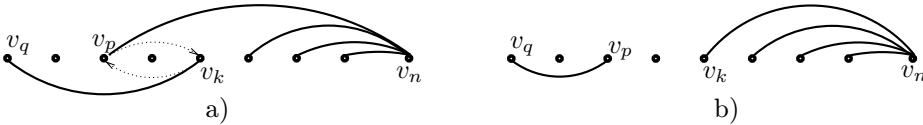
Nech  $\mathbf{p}_n$  je neklesajúca valenčná postupnosť grafu  $G = (V, H)$ . Chceme dokázať, že existuje graf  $G' = (V', H')$  s valenčnou postupnosťou  $\mathbf{p}_{n-1}$ . Nech  $\mathbf{v} = (v_1, v_2, \dots, v_n)$  je postupnosť vrcholov množiny  $V$  príslušných členom postupnosti  $\mathbf{p}_n$ , t. j.  $\deg(v_1) = d_1, \deg(v_2) = d_2, \dots, \deg(v_n) = d_n$ .

Predpokladajme, že vrchol  $v_n$  grafu  $G$  príslušný poslednému členu postupnosti  $\mathbf{p}_n$  je susedný so všetkými bezprostredne predchádzajúcimi  $d_n$  vrcholmi postupnosti  $\mathbf{v}$ , potom graf  $G' = G - \{v_n\}$  vzniknutý z grafu  $G$  odstránením vrchola  $v_n$  a všetkých s ním incidentných hrán má valenčnú postupnosť  $\mathbf{p}_{n-1}$ .

Potiaž je však v tom, že vrchol  $v_n$  nemusí byť susedný so všetkými  $d_n$  predchádzajúcimi vrcholmi v postupnosti  $\mathbf{v}$  — pozri obrázok 1.6 a). V takomto prípade zostrojíme z grafu  $G$  graf  $\overline{G}$  s rovnakou množinou vrcholov a s rovnakou valenčnou postupnosťou ako v grafe  $G$ , v ktorom už vrchol  $v_n$  bude susedný so všetkými  $d_n$  bezprostredne predchádzajúcimi vrcholmi v postupnosti  $\mathbf{v}$ .

Nech  $v_k$  je posledný (odpredu) vrchol postupnosti  $\mathbf{v}$ , ktorý nie je susedný s  $v_n$  a pred ktorým ešte existuje  $v_p$  susedný s  $v_n$ . Na obrázku 1.6 sú vrcholy postupnosti  $\mathbf{v}$  usporiadané podľa poradia vo  $\mathbf{v}$  (a teda aj podľa stupňa neklesajúco). Pretože  $v_p$  predchádza  $v_k$ , je  $\deg(v_p) \leq \deg(v_k)$ . Keďže  $v_p$  má jednu

hranu incidentnú s  $v_n$ , ostáva mu menej ostatných s ním incidentných hrán ako  $\deg(v_k)$ , z čoho vyplýva, že  $v_k$  musí byť susedný s aspoň jedným vrcholom, s ktorým nie je susedný vrchol  $v_p$ . Nech je to vrchol  $v_q$ . Urobme nasledujúcu zmenu: Zrušme hranu  $\{v_q, v_k\}$  a pridajme hranu  $\{v_q, v_p\}$ , zrušme hranu  $\{v_p, v_n\}$  a pridajme hranu  $\{v_k, v_n\}$ . Formálne zapísané  $\overline{H} = H - \{\{v_q, v_k\}, \{v_p, v_n\}\} \cup \{\{v_q, v_p\}, \{v_k, v_n\}\}$ . Výsledok znázorňuje obrázok 1.6 b). Stupeň žiadneho z vrcholov sa pri tejto operácii nezmení. Graf  $\overline{G} = (V, \overline{H})$  má viac členov postupnosti  $\mathbf{v}$  z predposledných  $d_n$  susedných s vrcholom  $v_n$  ako graf  $G$ , pričom oba majú rovnakú valenčnú postupnosť  $\mathbf{p}_n$ . Takto pokračujeme dovtedy, kým nedostaneme graf v ktorom je  $v_n$  susedný so všetkými  $d_n$  predchádzajúcimi vrcholmi postupnosti  $\mathbf{v}$ . ■



Obr. 1.6: K dôkazu vety 1.6.

Na obrázku sú zobrazené iba hrany incidentné s vrcholom  $v_n$  a hrana  $\{v_q, v_k\}$ .

Ostatných hrán sa popisovaný postup netýka, preto nie sú vyznačené.

Myšlienka dôkazu je ilustrovaná bodkovanými šípkami: vrchol  $v_k$  hrany  $\{v_q, v_k\}$  „prepne“ do vrchola  $v_p$  a vrchol  $v_p$  hrany  $\{v_p, v_n\}$  „prepne“ do vrchola  $v_k$ , čím sa stupne vrcholov  $v_p$ ,  $v_k$  nezmenia, ale vrchol  $v_k$  už susedí s vrcholom  $v_n$ .

**Definícia 1.23. Pravidelný graf stupňa  $k$**  je taký graf  $G = (V, H)$ , v ktorom má každý vrchol  $v \in V$  stupeň  $k$ .

**Definícia 1.24.** Grafy  $G = (V, H)$ ,  $\overline{G} = (\overline{V}, \overline{H})$  nazveme **komplementárne**, ak  $V = \overline{V}$  a pre každú dvojicu vrcholov  $u, v \in V$  takých, že  $u \neq v$ , platí:

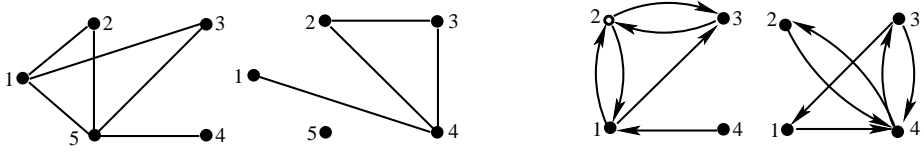
$$\{u, v\} \in H \text{ práve vtedy, keď } \{u, v\} \notin \overline{H}.$$

Analogicky definujeme dvojicu komplementárnych digrafov.

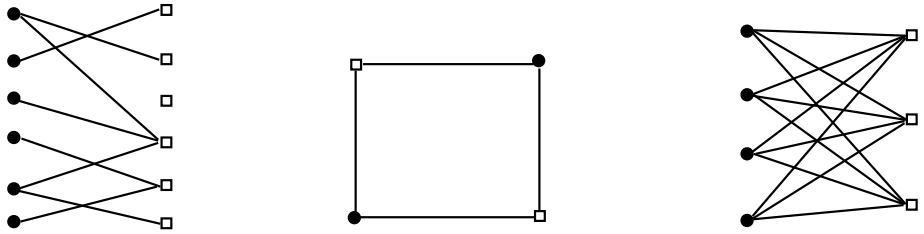
**Definícia 1.25.** Graf  $G = (V, H)$  nazveme **bipartitný**, ak jeho množinu vrcholov  $V$  možno rozdeliť na dve disjunktné neprázdne podmnožiny (partie alebo časti)  $V_1$ ,  $V_2$  tak, že žiadne dva vrcholy z tej istej časti nie sú susedné.

**Úplný bipartitný graf  $K_{mn}$**  je taký bipartitný graf s časťami  $V_1$ ,  $V_2$ , v ktorom  $|V_1| = m$ ,  $|V_2| = n$  a v ktorom je každý vrchol množiny  $V_1$  susedný s každým vrcholom množiny  $V_2$ .

*Poznámka.* Analogicky možno definovať  $k$ -partitný graf.



Obr. 1.7: Dvojice komplementárnych grafov a digrafov.



Obr. 1.8: Diagramy bipartitných grafov.  
 Vrcholy častí  $V_1$ ,  $V_2$  sú znázornené odlišne.  
 Prostredný diagram prislúcha grafu  $K_{2,2}$ ,  
 tretí diagram zľava je diagram grafu  $K_{4,3}$ .

**Definícia 1.26.** Nech  $G = (V, H)$  je graf. Jeho **hranovým grafom** nazveme graf  $L(G) = (H, E)$ , ktorého vrcholovú množinu tvorí hranová množina grafu  $G$  a ktorého hranová  $E$  množina je definovaná nasledovne:  $(h_1, h_2) \in E$  práve vtedy, keď sú hrany  $h_1$ ,  $h_2$  susedné.

**Definícia 1.27.** Graf, resp. digraf  $G = (V, H)$  nazveme **hranovo ohodnoteným**, ak každej hrane, resp. orientovanej hrane  $h \in H$  je priradené reálne číslo  $c(h)$  nazývané **cena hrany  $h$**  alebo tiež **ohodnotenie hrany  $h$** . Za hranovo ohodnotený graf budeme teda pokladať usporiadanú trojicu  $G = (V, H, c)$ , kde  $V$  je množina vrcholov,  $H$  množina hrán a  $c : H \rightarrow \mathbb{R}$  je reálna funkcia definovaná na množine  $H$ .

Podobne možno definovať **vrcholovo ohodnotený graf (digraf)** ako usporiadanú trojicu  $G = (V, H, d)$ , kde  $V$  je množina vrcholov,  $H$  množina hrán a  $d : V \rightarrow \mathbb{R}$  je reálna funkcia definovaná na množine  $V$ . Číslo  $d(v)$  nazveme **ohodnotenie vrchola  $v$**  alebo tiež **cena vrchola  $v$** .

*Poznámka.* Ohodnotenie – cena hrany môže predstavovať nejaký číselný parameter hrany – napr. dĺžku, kapacitu, nosnosť, útlm signálu. Podobne ohod-

notenie vrchola. Pri grafových modeloch reálnej dopravnej siete sa stretáme s grafmi, ktoré budú aj vrcholovo aj hranovo ohodnotené. Bežné sú aj príklady viacerých ohodnotení hrán, resp. vrcholov. Ak hrana modeluje cestný úsek, tento môže byť charakterizovaný svojou dĺžkou, šírkou, polomerom zákrut, svetlou výškou atď.

## 1.4 Rovnosť a izomorfizmus grafov

Pretože graf bol definovaný ako usporiadaná dvojica vrcholov a hrán, rovnosť (totožnosť) grafov  $G = (V, H)$ ,  $G' = (V', H')$  je vlastne rovnosťou usporiadaných dvojíc. Podľa toho sa grafy  $G$  a  $G'$  rovnajú práve vtedy, keď  $V = V'$  a  $H = H'$ . Existujú však dvojice grafov, ktoré majú všetky vlastnosti rovnaké a líšia sa nanajvýš – voľne povedané – pomenovaním vrcholov. Tieto grafy však nemusia byť rovnaké, pretože sa môžu líšiť v množine vrcholov i hrán. Pre takéto prípady sa zavádza pojem izomorfizmu.

**Definícia 1.28.** Graf  $G = (V, H)$  je **izomorfný s grafom**  $G' = (V', H')$ , ak existuje také vzájomne jednoznačné zobrazenie  $f : V \leftrightarrow V'$ , že pre každú dvojicu vrcholov  $u, v \in V$  platí:

$$\{u, v\} \in H \quad \text{práve vtedy, keď} \quad \{f(u), f(v)\} \in H'. \quad (1.14)$$

Zobrazenie  $f$  sa volá **izomorfizmus grafov**  $G$  a  $G'$ .

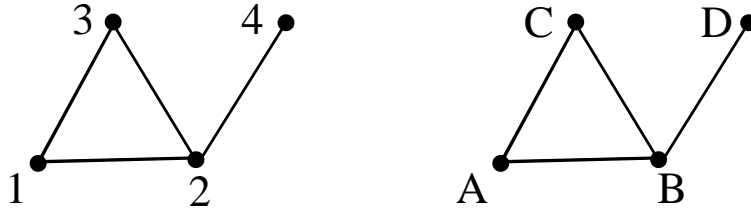
Digraf  $\vec{G} = (V, H)$  je **izomorfný s digrafom**  $\vec{G}' = (V', H')$ , ak existuje také vzájomne jednoznačné zobrazenie  $f : V \leftrightarrow V'$ , že pre každú dvojicu vrcholov  $u, v \in V$  platí:

$$(u, v) \in H \quad \text{práve vtedy, keď} \quad (f(u), f(v)) \in H'. \quad (1.15)$$

Zobrazenie  $f$  sa volá **izomorfizmus digrafov**  $\vec{G}$  a  $\vec{G}'$ .

*Poznámka.* Izomorfizmus grafov je reflexívna, symetrická a tranzitívna relácia – je to teda relácia ekvivalencie na triede všetkých grafov.

Ak sú grafy  $G$ ,  $G'$  izomorfné, musia mať všetky grafové charakteristiky rovnaké – napr. počet vrcholov, počet hrán, valenčné postupnosti, počet komponentov, počet cyklov s  $k$  hranami, počet ciest s  $k$  hranami, počet úplných podgrafov typu  $K_p$  atď. Takéto charakteristiky nazývame **invarianty izomorfizmu**. Invarianty izomorfizmu možno využiť na dôkaz toho, že grafy  $G$ ,  $G'$  nie sú izomorfné



Obr. 1.9: Dvojica izomorfných grafov.

Zobrazenie  $f$  definované rovnosťami $f(1) = A, f(2) = B, f(3) = C, f(4) = D$  je izomorfizmom.

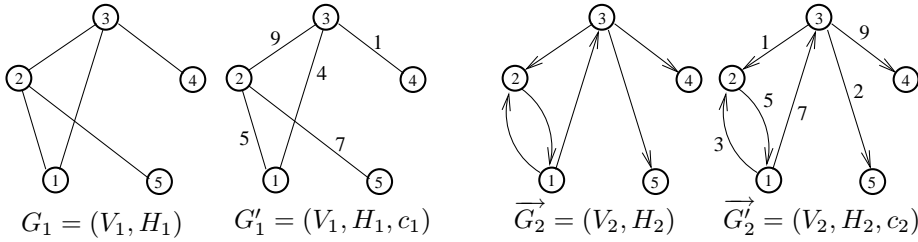
– ak sa ukáže, že  $G$  má niektorú vlastnosť inú ako  $G'$ , takéto grafy nemôžu byť izomorfné.

Na dôkaz izomorfности dvoch grafov, resp. digrafov treba zostrojiť konkrétne zobrazenie  $f$  s vlastnosťami (1.14), resp. (1.15). Zatiaľ na to nepoznáme iný spôsob ako vyskúšať všetky vzájomne jednoznačné zobrazenia množiny  $V$  na množinu  $V'$ , ktorých je  $n!$  (kde  $n = |V|$ ). **Problém grafového izomorfizmu** je navrhnúť prakticky realizovateľný všeobecný algoritmus, ktorý by pre ľubovoľné dva grafy rozhodol, či sú izomorfné alebo nie, alebo dokázať, že žiaden taký algoritmus neexistuje.

## 1.5 Reprezentácia grafov a digrafov

Existuje viacero spôsobov na reprezentáciu grafov a digrafov. Niektoré sú vhodné na ilustrovanie niektorých pojmov a postupov, iné sa používajú pri ukladaní grafových štruktúr do pamäte počítača alebo na záznamové médium. Rôzne spôsoby reprezentácie sú rôzne náročné nielen na pamäť, ktorú potrebujú ale aj na čas prístupu k žiadanej informácii. Voľba spôsobu uloženia grafu alebo digrafu v počítači by mala závisieť aj od spôsobu práce algoritmu, pre ktorý uvažovanú reprezentáciu robíme. Voľba údajovej štruktúry pre reprezentáciu grafu má veľký vplyv na výslednú zložitosť algoritmu. Spomenieme niektoré z najčastejších reprezentácií grafov a digrafov.





Obr. 1.10: Diagramy grafu, hranovo ohodnoteného grafu, digrafu a hranovo ohodnoteného digrafu.

## 1. Repräsentácia diagramom grafu

Diagramy sú veľmi výhodné pre znázornenie rôznych vlastností grafov a digrafoV, situácií a obrátov pri dôkazoch viet a na ilustrovanie postupu algoritmov. S rastúcim počtom vrcholov a hrán sa stávajú neprehľadné a pre algoritmické výpočty nevhodné. Veľkým nebezpečením postupov prezentovaných len na diagramoch je nevedomé používanie intuície (napríklad pri kontrolovaní súvislosti alebo vzniku cyklu v nejakej štruktúre). Preto odporúčam čitateľovi vyskúšať si každý algoritmus bez použitia diagramu s niektorou ďalšou repräsentáciou. Za najlepší spôsob na pochopenie činnosti algoritmu považujem napísať a odladiť preň program, ktorý dokáže vyriešiť daný problém aj pre rozsiahly prípad.

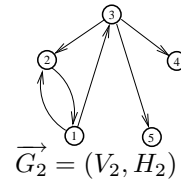
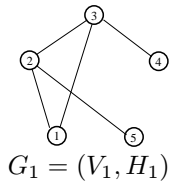
## 2. Repräsentácia množinami vrcholov a hrán, resp. orientovaných hrán.

Nech  $V_1 = \{1, 2, 3, 4, 5\}$ ,  $H_1 = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 4\}\}$ . Množinami  $V_1$  a  $H_1$  je jednoznačne určený graf  $G_1 = (V_1, H_1)$ . Podobne nech  $V_2 = \{1, 2, 3, 4, 5\}$  a  $H_2 = \{(1, 2), (1, 3), (2, 1), (3, 2), (3, 4), (3, 5)\}$ , potom množinami  $V_2, H_2$  je jednoznačne určený digraf  $\vec{G}_2 = (V_2, H_2)$ .

V počítači môžeme množinu vrcholov  $V$  repräsentovať ako jednorozmerné pole  $V$  s  $n = |V|$  prvkami, kde  $V[i]$  je  $i$ -tý vrchol. Množinu hrán môžeme uložiť do dvojrozmerného poľa  $H$  typu  $(m \times 2)$ , kde  $m = |H|$  je počet hrán,  $H[j, 1]$  je začiatočný a  $H[j, 2]$  koncový vrchol  $j$ -tej hrany, čím je daná aj orientácia tejto hrany v prípade digrafu. V prípade grafu nezáleží na poradí vrcholov  $H[j, 1]$ ,

$H[j, 2]$ , (môžeme sa však dohodnúť napríklad na poradí  $H[j, 1] < H[j, 2]$ , čo sa niekedy môže s výhodou využiť).

Ak ide navyiac o hranovo ohodnotený graf alebo digraf, ohodnotenia hrán môžeme ukladať do zvláštneho jednorozmerného poľa  $C[\ ]$  dĺžky  $m = |H|$  (kde  $C[j]$  je ohodnotenie  $j$ -tej hrany), alebo hrany ukladať do dvojrozmerného poľa  $H$  typu  $m \times 3$ , kde  $H[j, 1]$ ,  $H[j, 2]$ , sú začiatočný a koncový vrchol  $j$ -tej hrany a  $H[j, 3]$  je ohodnotenie  $j$ -tej hrany.



$i$	1	2	3	4	5
$V[i]$	1	2	3	4	5

$i$	1	2	3	4	5
$V[i]$	1	2	3	4	5

$j$	1	2	3	4	5
$H[j, 1]$	1	1	2	2	3
$H[j, 2]$	2	3	3	5	4
$C[j] = H[j, 3]$	5	4	9	7	1

$j$	1	2	3	4	5	6
$H[j, 1]$	1	1	2	3	3	3
$H[j, 2]$	2	3	1	2	4	5
$C[j] = H[j, 3]$	3	7	5	1	9	2

Tabuľka 1.1:  $\vec{G}_2$   
Reprezentácia grafu  $G_1$  a digrafu  $\vec{G}_2$  z obrázku 1.10.

Z mnohých dôvodov býva výhodné, aby množina  $V$  vrcholov grafu bola množinou  $\{1, 2, \dots, n\}$ , kde  $n = |V|$ . V takomto prípade môžeme totiž vrcholy priamo používať ako indexy matic alebo vektorov. Často však tomu tak nie je, mnohokrát sú vrcholy očíslované veľkými číslami nesúvisle alebo dokonca vrcholy ani nie sú čísla, ale textové reťazce či iné objekty, napríklad:  $V = \{\text{„Banská Bystrica“}, \text{„Bratislava“}, \text{„Brno“}, \dots\}$ . V takýchto prípadoch je vhodné premenovať vrcholy číslami od 1 do  $n$  a pracovať s grafom izomorfným s pôvodným, ktorý už má množinu vrcholov  $\{1, 2, \dots, n\}$ .

Reprezentácia grafu množinou hrán je pamäťovo nenáročná a je vhodná pre algoritmy založené na postupnom systematickom prehľadávaní všetkých hrán, ako je napríklad Kruskalov algoritmus na hľadanie najlacnejšej kostry grafu

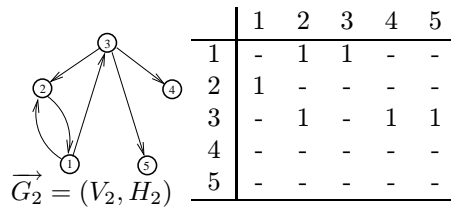
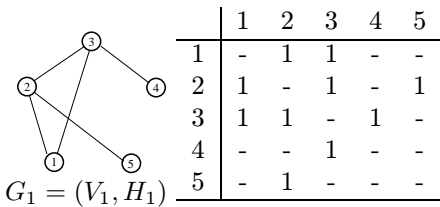
alebo základný algoritmus na hľadanie najkratších ciest z jedného vrchola do ostatných vrcholov grafu. Nehodí sa pre algoritmy, ktoré potrebujú zisťovať pre náhodné  $u, v \in V$  existenciu hrany  $\{u, v\}$  – odpoveď na každú takúto otázku by si vyžadovala prezrieť celý zoznam hrán.

Nakoniec poznamenajme, že táto reprezentácia sa hodí aj pre pseudografy a pseudodigrafy (dovolené viacnásobné hrany a slučky). Pre pseudomigrafy bude vhodné udržiavať dve hranové štruktúry – jednu pre neorientované a druhú pre orientované hrany.

### 3. Reprezentácia maticou príľahlosti.

Matica príľahlosti  $\mathbf{M} = (m_{ij})$  je štvorcová matica typu  $n \times n$ , kde  $n = |V|$  je počet vrcholov grafu, resp. digrafu  $G$ , ktorej prvky sú definované nasledovne:

$$m_{ij} = \begin{cases} 1 & \text{ak } \{i, j\} \in H \\ 0 & \text{inak} \end{cases} \quad m_{ij} = \begin{cases} 1 & \text{ak } (i, j) \in H \\ 0 & \text{inak} \end{cases} \quad (1.16)$$



Matica príľahlosti grafu  $G_1$ .

Matica príľahlosti digrafu  $\vec{G}_2$ .

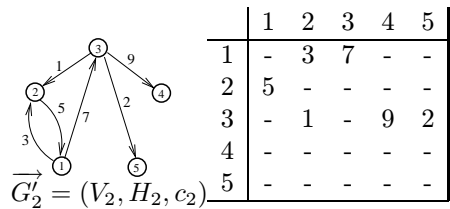
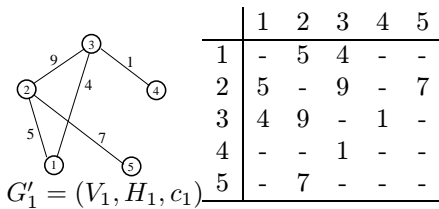
Pre väčšiu prehľadnosť sme v oboch predchádzajúcich maticiach písali pomlčku namiesto nuly. Všimnime si, že matica príľahlosti grafu je symetrická podľa hlavnej diagonály. Skutočne, ak  $m_{ij} = 1$ , potom aj  $m_{ji} = 1$ . Matica príľahlosti digrafu však vo všeobecnom prípade nemusí byť symetrická, ako je vidieť z predchádzajúceho príkladu. Poznamenajme ešte, že v definícii (1.16) hodnôt  $m_{ij}$  prvkov matíc príľahlosti grafu a digrafu môžeme nahradiť 1 napríklad logickou hodnotou TRUE a 0 logickou hodnotou FALSE, hodnotou  $\infty$ , alebo dokonca ľubovoľnými dvoma rôznymi hodnotami, na základe ktorých rozlíšime, či sú dva vrcholy príľahlé alebo nie.

Na reprezentácii grafu, resp. digrafu maticou príľahlosti vidíme úzku súvislosť medzi binárnymi reláciami a digrafmi. Množina hrán digrafu je vlastne antireflexívnou reláciou na množine vrcholov  $V$ . Podobne existuje vzájomne jednoznačný vzťah medzi grafmi a antireflexívnymi symetrickými binárnymi reláciami na množine vrcholov  $V$ .

#### 4. Reprezentácia maticou ohodnotení hrán hranovo ohodnoteného grafu alebo digrafu

Pre hranovo ohodnotené grafy a digrafy definujeme maticu  $\mathbf{M}$  ohodnotení hrán grafu, resp. digrafu ako štvorcovú maticu typu  $n \times n$ , kde  $n = |V|$  je počet vrcholov grafu, resp. digrafu a prvky ktorej sú definované nasledovne:

$$m_{ij} = \begin{cases} c(\{i, j\}) & \text{ak } \{i, j\} \in H \\ \infty & \text{inak} \end{cases} \quad m_{ij} = \begin{cases} c((i, j)) & \text{ak } (i, j) \in H \\ \infty & \text{inak} \end{cases} \quad (1.17)$$



Matica ohodnotení hrán grafu  $G'_1$ .

Matica ohodnotení hrán digrafu  $\overrightarrow{G'_2}$ .

Opäť si všimnime, že matica ohodnotení hrán grafu je symetrická podľa hlavnej diagonály. Je identická s maticou digrafu, ktorý by sme z grafu  $G$  dostali nahradením každej hrany dvojicou opačne orientovaných hrán. Matica digrafu spravidla nie je symetrická podľa hlavnej diagonály.

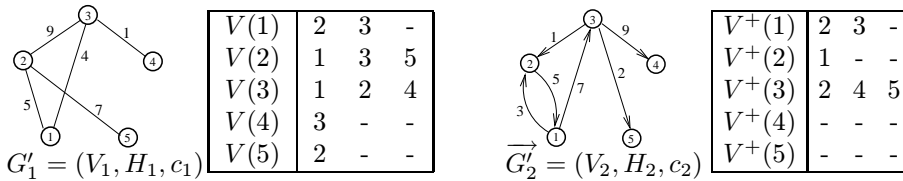
Reprezentácia maticou príľahlosti alebo maticou ohodnotení hrán nešetří pamäťou – nezávisle od počtu hrán grafu spotrebuje  $n^2$  pamäťových miest. Hodí sa pre algoritmy, ktoré potrebujú často zisťovať existenciu raz tej inokedy inej hrany. Tiež je vhodná na ukladanie úplných grafov a digrafov alebo skoro úplných grafov a digrafov. Nehodí sa pre grafy s malým počtom hrán (napríklad pre grafy blízke rovinným) a pre algoritmy založené na postupnom systematickom prehľadávaní okolí vrcholov ako sú Tarryho algoritmus, Dijkstrov algoritmus,

Ford-Fulkersonov algoritmus, CPM-metóda atď. Na zistenie všetkých vrcholov okolia nejakého vrchola treba totiž pri tejto implementácii prehľadať celý riadok matice.

Pretože hodnota na mieste  $(i, j)$  matice príľahlosti, resp. matice ohodnotení hrán hovorí len o existencii alebo neexistencii hrany medzi vrcholmi  $i, j$ , maticové reprezentácie sa nehodia pre grafové štruktúry s viacnásobnými hranami ako sú multigrafy a multidigrafy.

### 5. Reprezentácia zoznamom vrcholov okolia každého vrchola grafu alebo vrcholov výstupnej hviezdy každého vrchola digrafu.

Graf možno reprezentovať tak, že ku každému vrcholu  $v$  zadáme množinu  $V(v)$  — t. j. zoznam jeho najbližších susedov. Podobne digraf možno reprezentovať tak, že ku každému vrcholu  $v$  zadáme množinu  $V^+(v)$  — t. j. množinu koncov hrán vychádzajúcich z vrchola  $v$ . Pre graf  $G_1$  a digraf  $\vec{G}_2$  z obrázku 1.10 sú tieto zoznamy v nasledujúcich tabuľkách:



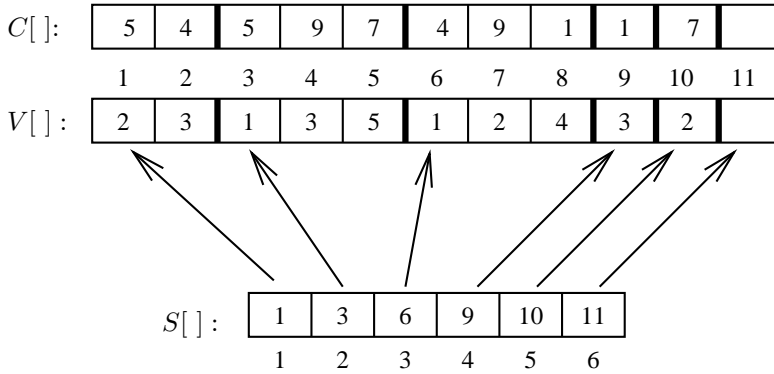
Vrcholy okolí pre graf  $G'_1$ .

Vrcholy výstupných hviezd pre digraf  $\vec{G}'_2$ .

Všimnime si, že zoznamy vrcholov okolí ľubovoľného grafu  $G$  sú také isté ako zoznamy vrcholov výstupných hviezd digrafu, ktorý dostaneme z grafu  $G$  tak, že každú jeho hranu nahradíme dvojicou opačne orientovaných hrán.

Pre neorientované grafy táto reprezentácia ukladá každú hranu  $\{u, v\}$  dvakrát — raz v popise okolia vrchola  $u$  a druhýkrát v popise okolí vrchola  $v$ . Pri súčasnom stave výpočtovej techniky, keď jeden megabyte pamäte stojí rádovo koruny, však táto mierna neefektívnosť väčšinou nevaďí.

Veľmi efektívne možno zoznamy najbližších susedov implementovať tak, že do poľa  $V[ ]$  najprv zapíšeme najbližších susedov vrchola 1, potom najbližších susedov vrchola 2 atď., až nakoniec najbližších susedov posledného vrchola.



Obr. 1.11: Reprezentácia zoznamov susedov pomocou smerníkov.

Na obrázku 1.11 sú okolia jednotlivých vrcholov oddelené v poli  $V[]$  hrubou čiarou. Súčasne do poľa smerníkov  $S[]$  zapisujeme pozíciu prvého suseda prvého vrchola  $S[1] := 1$ , potom pozíciu prvého suseda druhého vrchola  $S[2]$  až nakoniec pozíciu prvého suseda  $n$ -tého vrchola. Aby sme vedeli, kde končí zápis susedov posledného vrchola, do  $S[n + 1]$  zapíšeme pozíciu prvého nepoužitého miesta vo vektore  $V[]$ . Pre hranovo ohodnotený grafy a digrafy môžeme spoločne s vektorom  $V[]$  udržiavať aj paralelný vektor ohodnotení príslušných hrán  $C[]$ . Tento spôsob je ilustrovaný na obrázku 1.11. Pretože  $S[3] = 6$  a  $S[4] = 9$ , všetci najbližší susedia vrchola 3 začínajú na pozícii 6 a končia na pozícii  $S[4] - 1 = 8$  vo vektore  $V[]$ . Susedia vrchola 3 sú teda  $V[6] = 1$ ,  $V[7] = 2$  a  $V[8] = 4$ , ohodnotenia hrán  $\{3, 1\}$ ,  $\{3, 2\}$  a  $\{3, 4\}$  sú  $C[6] = 4$ ,  $C[7] = 9$  a  $C[8] = 1$ .

Ak by nejaký vrchol  $i$  grafu  $G$  nemal žiadnych susedov, smerník  $S[i + 1]$  bude ukazovať to isté miesto ako smerník  $S[i]$ , t. j.  $S[i] = S[i + 1]$ .

Pri takejto reprezentácii grafov existuje istá duplicita, pretože hrana  $\{u, v\}$  prispieva do množiny  $V(u)$  vrcholom  $v$  a do množiny  $V(v)$  vrcholom  $u$ . Ak udržiavame súčasne aj vektor cien, ten takisto bude obsahovať ocenenie hrany  $h$  dvakrát. Algoritmy založené na prehľadávaní okolí vrcholov však s výhodou využijú skutočnosť, že dáta charakterizujúce okolia sú pohromade a rýchlo dostupné, a odplatia sa tak zvýšenou výpočtovou rýchlosťou i jednoduchosťou implementácie. Pri digrafoch takáto duplicita nie je. Všimnime si ešte, že práve opísaná reprezentácia hranovo ohodnoteného grafu  $G = (V, H, c)$  je totožná s reprezentáciou digrafu  $\vec{G} = (V, H', c')$  s rovnakou množinou vrcholov  $V$ ,

množinou hrán  $H'$ , ktorá obsahuje ku každej hrane  $h \in H$  dvojicu opačne orientovaných hrán, obe s rovnakou cenou  $c(h)$ .

Tento spôsob implementácie sa zvlášť hodí pre grafy s malým počtom hrán a pre algoritmy, založené na postupnom systematickom prehľadávaní okolí vrcholov ako sú Tarryho algoritmus, Dijkstrov algoritmus, Ford-Fulkersonov algoritmus, CPM-metóda atď, ale aj pre algoritmy, ktoré potrebujú systematicky prehľadávať všetky hrany grafu.

Pretože táto reprezentácia môže modelovať aj viacnásobné hrany, je vhodná aj pre multigrafy a multidigrafy. Pre pseudomigrafy bude vhodné udržiavať dve štruktúry susedov – koncov orientovaných a koncov neorientovaných hrán vychádzajúcich z každého vrchola.

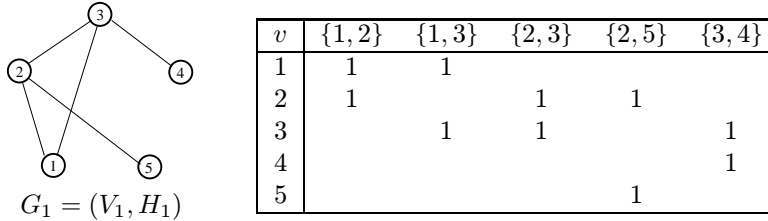
## 6. Reprezentácia incidenčnou maticou vrcholov a hrán

Mnoho literatúry uvádza ako možný spôsob reprezentácie grafových štruktúr tzv. incidenčnú maticu vrcholov a hrán. **Incidenčná matica vrcholov a hrán** je matica  $\mathbf{B}$  typu  $n \times m$ , kde  $n$  je počet vrcholov a  $m$  počet hrán reprezentovaného grafu alebo digrafu. Každý prvok  $b_{ij}$  matice  $\mathbf{B}$  hovorí o spôsobe incidencie vrchola  $i$  s hranou  $j$  nasledovne:

$$b_{ij} = \begin{cases} 1 & \text{ak vrchol } i \text{ je incidentný s hranou } j \text{ v grafe } G \\ 0 & \text{inak} \end{cases}$$

$$b_{ij} = \begin{cases} 1 & \text{ak vrchol } i \text{ je začiatočným vrcholom hrany } j \text{ v digrafe } \vec{G} \\ -1 & \text{ak vrchol } i \text{ je koncovým vrcholom hrany } j \text{ v digrafe } \vec{G} \\ 0 & \text{inak} \end{cases}$$

Tento spôsob je vhodný aj pre multigrafy, multidigrafy a multimigrafy. Pre pseudomigrafy sa dá dodefinovať  $b_{ij}$  aj pre slučky vzťahom  $b_{ij} = 2$ , ak  $j$  je neorientovaná slučka začínajúca a končiaca vo vrchole  $i$  a vzťahom  $b_{ij} = -2$ , ak  $j$  je orientovaná slučka začínajúca a končiaca vo vrchole  $i$ .



Tabuľka 1.2: Incidenčná matica grafu  $G_1 = (V_1, H_1)$  z obrázku 1.10.  
 $(V_1 = \{1, 2, 3, 4, 5\}, H_1 = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 4\}\})$ .

Incidenčná matica vrcholov a hrán má v každom stĺpci nana najviac dva nenulové prvky a súčet absolútnych hodnôt týchto prvkov je 2. Incidenčná matica plytvá miestom, jej rozmer je  $n \times m$ , pričom počet nenulových prvkov je  $2m$  (pre pseudomigrafy  $\leq 2m$ ). Zisťovanie koncových vrcholov hrany  $j$  vyžaduje v najhoršom prípade prezretie celého  $j$ -teho stĺpca, na čo je potrebných  $n$  porovnaní. Zistenie, s ktorými hranami je vrchol  $j$  incidentný vyžaduje v najhoršom prípade prezretie  $m$  hodnôt v riadku  $i$ .

Napriek svojej priestorovej náročnosti incidenčná matica vrcholov a hrán neponúka žiadne časové úspory – práve naopak. Preto jej praktické využitie je veľmi zriedkavé, avšak táto reprezentácia môže byť užitočná pri niektorých teoretických úvahách.

Doteraz spomínané reprezentácie grafov a digrafov zďaleka nevyčerpávajú všetky možnosti. V prípade reprezentácie špeciálnych grafov možno využiť ich vlastnosti na voľbu štruktúry, do ktorej ich budeme ukladať. Tak napríklad v koreňovom strome (pozri definíciu 4.3 na str. 110) má každý vrchol okrem koreňa práve jedného otca. Za predpokladu, že množina vrcholov koreňového stromu je  $V = \{1, 2, \dots, n\}$ , môžeme ho reprezentovať jednorozmerným poľom  $X[\ ]$ , v ktorom pre  $i \in V$   $X[i] = 0$ , ak  $i$  je koreň, inak  $X[i]$  je otec vrchola  $i$ .

V časti 5.5.1 (str. 156) je uvedená definícia prioritného stromu a jeho reprezentácia pomocou jednorozmerného poľa.

Exaktný algoritmus 8.4 (str. 233) na farbenie grafu  $G = (V, H)$  predpokladá, že  $V = \{1, 2, \dots, n\}$ . Pre každé  $i \in V$  udržuje množinu susedov vrchola  $i$  s menším poradovým číslom než  $i$ , t. j.:

$$P(i) = V(i) \cap \{1, 2, \dots, i - 1\}.$$

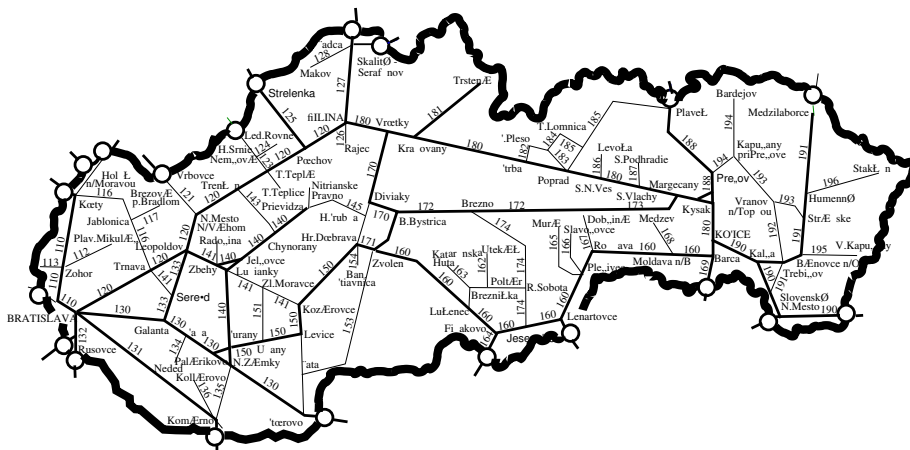


Množiny  $P(1), P(2), \dots, P(n)$  úplne definujú hranovú množinu  $H$  a takáto reprezentácia sa ukazuje efektívnou pre príslušný algoritmus.

## 1.6 Aplikácie

### 1.6.1 Modelovanie reálnej dopravnej siete

Veľmi často sa prostriedky teórie grafov využívajú pri modelovaní dopravnej siete. Cestovný poriadok Železníc Slovenskej republiky obsahuje na jednej zo svojich prvých strán nasledujúci obrázok:

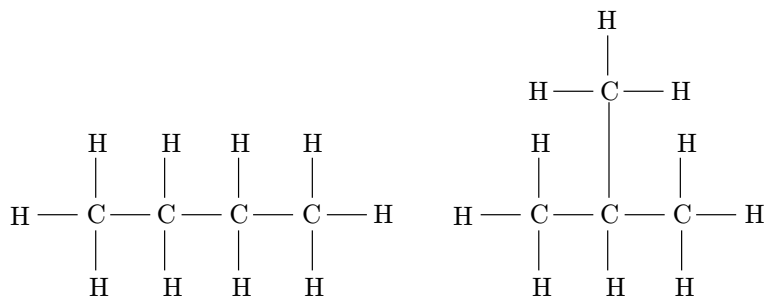


Obr. 1.12: Model železničnej siete na Slovensku.

Tento obrázok môžeme považovať za diagram hranovo ohodnoteného grafu  $G$ . Ohodnotenie hrany grafu vyjadruje príslušnosť modelovaného úseku k trati a slúži na rýchle nájdenie spojov, ktoré cez úsek modelovaný príslušnou hranou premávajú.

### 1.6.2 Chemické grafy

Chemické zlúčeniny sa vyjadrujú jednak sumárnym vzorcom - napríklad  $C_4H_{10}$ , ale viac informácií dávajú štruktúrne vzorce.



Obr. 1.13: Dva štruktúrne vzorce pre  $C_4H_{10}$ .

Otázka, koľko štruktúrnych vzorcov existuje k spomínanému sumárnemu vzorcu, je skôr otázka pre teóriu grafov ako pre chémiu samotnú. V teórii grafov sa tento problém preformuluje na otázku, koľko je neizomorfných grafov, resp. multigrafov so štyrmi vrcholmi stupňa 4 a desiatimi vrcholmi stupňa 1. Otázka počtu rôznych štruktúrnych vzorcov k danému sumárnemu vzorcu je vo všeobecnosti ťažká otázka. Pre uhľovodíky sa ňou zaoberal už Cayley v r. 1874, pričom grafické zobrazovanie mu bolo dobrou pomôckou.

Zo samých základov teórie grafov vyplýva, že neexistuje zlúčenina kyslíka, vodíka a uhlíka s nepárnym počtom vodíkov (aplikácia vety 1.4). Všeobecnejšie — každá molekula musí obsahovať párny počet atómov s nepárnou valenciou.

Na otázku, či možno vytvoriť molekulu s daným počtom daných prvkov bez viacnásobných väzieb, odpovedá veta 1.6. Nutnou podmienkou pre to je, aby postupnosť valencií týchto prvkov vytvárala valenčnú postupnosť pre nejaký graf.

Na záver tejto časti poznamenajme, že pre chemické modely sú veľmi často potrebné multigrafy, pretože v mnohých chemických (najmä organických) zlúčeninách sa vyskytujú viacnásobné chemické väzby.

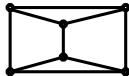
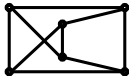
### 1.6.3 Intelektuálne vlastníctvo počítačového čipu

Nedlho po uvedení nového čipu spoločnosti ABC spoločnosť CDE uvedie čip so zarážajúco podobnými operačnými vlastnosťami. Technici spoločnosti ABC majú možnosť preštudovať rozloženie súčiastok na čipe spoločnosti CDE, ale táto by si v prípade použitia ukradnutej schémy dala zvlášť záležať na tom, aby rozloženie súčiastok na čipe bolo odlišné od svojho vzoru. Ak by sa spoločnosti ABC podarilo dokázať, že čip CDE je púhym prearanžovaním pôvodného čipu, bol by to základ pre súdne konanie.

Na čipe sú súčiastky a vedenia medzi nimi. Súčiastky môžeme modelovať vrcholmi grafu a dvojica súčiastok bude tvoriť hranu grafu, ak je medzi nimi vedenie. Ak by sa podarilo dokázať, že medzi grafom pre čip ABC a grafom pre čip CDE existuje izomorfizmus, dokázali by sme, že čip CDE je skonštruovaný podľa tej istej schémy ako čip ABC. Keďže však dnes dosahujú počty súčiastok na čipoch rádovo číslo  $10^6$ , ide o mimoriadne ťažkú úlohu a jej riešenie by mohlo trvať veľmi dlho. Znalosť organizácie čipu by možno mohla pomôcť technikom spoločnosti ABC nájsť spôsob na zrýchlenie hľadania izomorfizmu.

## 1.7 Cvičenia

1. Cestnú sieť s obojsmernými ulicami možno modelovať grafom. Ako by ste modelovali cestnú sieť, obsahujúcu jednosmerné i obojsmerné komunikácie? Dajú sa nejako modelovať zákazy odbočenia? Ako modelovať železničné koľajisko?
2. Existuje pravidelný graf tretieho stupňa s piatimi vrcholmi? Nakreslite si diagramy niektorých pravidelných grafov  $k$ -teho stupňa.
3. Existuje graf (digraf), ktorý je izomorfný so svojim komplementárnym grafom (digrafom)? Skúste formulovať niektoré nutné podmienky, ktoré musí spĺňať graf, aby bol izomorfný so svojím komplementom.

4. Sú grafy s diagramami   izomorfné?  
(Návod: Porovnajete počty podgrafov typu  $K_3$ .)

**Počítačové cvičenia**

5. Vytvorte (napríklad editorom) textový vstupný súbor `VRCHOLY.TXT` obsahujúci zoznam všetkých vrcholov grafu  $G$  a súbor `HRANY.TXT` obsahujúci zoznam hrán grafu  $G$ . Pre každý vrchol súbor `VRCHOLY.TXT` obsahuje jeden riadok, v ktorom je číslo vrchola v dekadickom formáte. Pre každú hranu bude súbor `HRANY.TXT` obsahovať jeden riadok obsahujúci tri dekadické čísla: prvý vrchol hrany, druhý vrchol hrany, ohodnotenie hrany. Spočiatku zvolte za množinu vrcholov množinu typu  $V = \{1, 2, \dots, n\}$ . Napíšte program v zvolenom jazyku na načítavanie súboru  $G$  zo súborov `VRCHOLY.TXT` a `HRANY.TXT`. V pamäti počítača uložte graf  $G$  vo forme

- zoznamov vrcholov a hrán
- matice príľahlosti
- zoznamov okolí vrcholov

Porovnajete pamäťové nároky jednotlivých reprezentácií. Premyslite, aké pamäťové nároky by mala matica incidencie vrcholov a hrán.

Napíšte analogické programy aj pre digrafy.

6. Je vôbec nutné zadávať množinu vrcholov v príklade 5? (Čo s izolovanými vrcholmi?) Čo by bolo treba urobiť v prípade, keby množina  $V$  obsahovala ako vrcholy ľubovoľné prirodzené čísla napríklad  $V = \{5457, 304, 3879, 2099, 9423, \text{atď.}\}$ ? Čo v prípade, keď by  $V = \{\text{Košice, Žilina, Čadca, Ružomberok, Poprad, atď.}\}$ ?
7. Napíšte program pre generovanie náhodného grafu alebo digrafu s  $n$  vrcholmi a  $m$  hranami ( $n, m$  budú vstupné parametre programu). Výsledný graf, resp. digraf uložte vo vhodnom formáte na disk.
8. Pre niektoré ťažké úlohy (Travelling Salesman Problem, Graph Coloring Problem, Finding All Cliques atď.) existujú na internete `www`-stránky s testovacími príkladmi. Nájdite niektoré z nich, „stiahnite“ si z nich niektoré súbory s testovacími grafmi. Zistite, v akom formáte ukladajú testovací graf alebo digraf a prípadne skúste načítať testovací graf zo získaného súboru.

## Kapitola 2

# Algoritmy a ich zložitosť

Neodmysliteľnou súčasťou modernej algoritmickej teórie grafov je rozbor obťažnosti riešeného problému a stanovenie výpočtovej zložitosti navrhnutého algoritmu. Pojmy ako „obťažnosť“, „výpočtová zložitosť“ treba najprv korektne matematicky definovať a potom na základe týchto definícií ukázať vlastnosti skúmaných problémov. Týmito záležitosťami sa zaoberá teória výpočtovej zložitosti, ktorá dnes dospela do značnej dokonalosti. Zistila, že existujú „ľahké“ a „ťažké“ problémy, a zostavila celú hierarchiu obťažnosti problémov. Vďaka nej máme dnes celé zoznamy „ťažkých“ problémov.

Pre naše účely však bude podstatné rozoznať základný rozdiel medzi „ľahkými“ a „ťažkými“ problémami. Preto cieľom tejto kapitoly je podať základy výpočtovej zložitosti algoritmov v najjednoduchšej možnej forme. Táto kapitola je spracovaná podľa Plesnikovej knihy [13].

### 2.1 Algoritmy

Pod pojmom **algoritmus** rozumieme postupnosť krokov, ktorá nás dovedie k žiadanému riešeniu daného problému. Žiada sa, aby algoritmus mal tieto vlastnosti:

- determinovanosť – má byť zadaný konečným počtom jednoznačných pravidiel

- efektívnosť – má zaručiť vyriešenie úlohy po konečnom počte krokov
- hromadnosť – má byť použiteľný na celú triedu prípadov úlohy svojho typu

Pravidlá v algoritme musia byť rozhodnuteľné v okamihu výpočtu. Pekný príklad nekorektného pravidla uvádza Plesník v [13]: „Ak existujú mimozemské civilizácie, tak choď domov, inak zostaň tu.“ Pretože nevieme, či mimozemské civilizácie existujú, alebo nie, nevieme v algoritme pokračovať. Jedným z najznámejších algoritmov je Euklidov algoritmus na hľadanie najväčšieho spoločného deliteľa  $\text{NSD}(a, b)$  dvoch prirodzených čísel  $a, b$ .

**Algoritmus 2.1. Euklidov algoritmus na hľadanie najväčšieho spoločného deliteľa čísel  $a, b$ .**

- **Krok 1.** Polož  $r_0 := a, r_1 := b, i := 0$ .
- **Krok 2.** Nájdi celé čísla  $k, r_{i+2}$  také, že  $r_i = k \cdot r_{i+1} + r_{i+2}$ , kde  $0 \leq r_{i+2} < r_{i+1}$ .
- **Krok 3.** Ak  $r_{i+2} = 0$ , polož  $\text{NSD}(a, b) := r_{i+1}$ , STOP. Inak polož  $i := i+1$  a GOTO Krok 2.



Už v 19. storočí sa francúzsky matematik G. Lamé (1795-1870) zaujímal o výpočtovú zložitosť Euklidovho algoritmu a ukázal, že počet opakovaní kroku 2 (t. j. počet celočíselných delení so zvyškom) nepresiahne päťnásobok počtu dekadických cifier menšieho z čísel  $a, b$ .

Ukázalo sa rozumné posudzovať algoritmy podľa počtu elementárnych krokov, ktoré potrebuje na vyriešenie daného problému. Tieto elementárne kroky môžu byť sčítanie, odčítanie, násobenie, delenie, porovnávanie s vetvením atď. Jednotlivé elementárne kroky považujeme za rovnako časovo náročné. Budeme hovoriť, že **algoritmus vyrieši danú konkrétnu úlohu  $U$  v čase  $T$** , ak na jej vyriešenie potrebuje  $T$  elementárnych krokov.

Pre ohodnotenie výpočtovej zložitosti algoritmu nás však viac ako jeden konkrétny prípad zaujíma závislosť počtu elementárnych krokov algoritmu na veľkosti resp. rozsahu počítanej úlohy. Budeme definovať **dĺžku úlohy** ako množstvo vstupných dát príslušnej úlohy. S istým zjednodušením možno za množstvo dát považovať počet čísel, ktorými je úloha charakterizovaná (bez ohľadu na ich veľkosť). Tak napríklad úlohu usporiadať množinu prirodzených

čísel podľa veľkosti možno zadať  $n$  číslami, a preto jej dĺžka je  $n$ . Graf možno zadať  $n$  vrcholmi a  $m$  hranami, preto dĺžka každej úlohy o grafe bude  $(n + m)$ . V grafových štruktúrach budeme obyčajne brať za dĺžku úlohy  $(n + m)$  aj keď pôjde o ohodnotené grafové štruktúry.

Ak máme vyhodnotiť závislosť počtu krokov  $T(n)$  algoritmu na dĺžke  $n$  úlohy, stojíme pred zložitým problémom. Počet krokov algoritmu môže totiž závisieť nielen od množstva vstupných dát, ale aj od ich vzájomnej konfigurácie. Tak napríklad niektorý algoritmus na usporiadanie množiny  $n$  čísel môže riešiť túto úlohu v diametrálne odlišnom čase, ak dostane ako vstup už usporiadanú množinu, ako keď mu necháme zotriediť množinu usporiadanú v opačnom poradí. V prípade grafových algoritmov môže byť situácia podobná, ak nie zložitejšia. Preto funkcia  $T(n)$  môže vyjadrovať iba hornú hranicu počtu krokov algoritmu pre najhorší prípad úlohy s dĺžkou  $n$  (worst case analysis).

**Definícia 2.1.** Nech  $g(n)$ ,  $h(n)$  sú dve kladné funkcie definované na množine prirodzených čísel. Budeme písať  $g(n) = O(h(n))$  a hovoriť, že **funkcia**  $h(n)$  **asymptoticky dominuje funkcii**  $g(n)$ , ak existuje konštanta  $K$  a prirodzené číslo  $n_0$  také, že

$$g(n) \leq K \cdot h(n) \quad \forall n \geq n_0.$$

Hovoríme, že **algoritmus**  $\mathcal{A}$  **má zložitosť**  $O(f(n))$ , ak pre horný odhad  $T(n)$  počtu krokov algoritmu  $\mathcal{A}$  pre úlohu dĺžky  $n$  platí

$$T(n) = O(f(n)).$$

Skrátene hovoríme o  $O(f(n))$  algoritme. Špeciálne ak  $f(n) \leq n^k$  pre nejaké konštantné  $k$ , hovoríme, že  $\mathcal{A}$  je **polynomiálny algoritmus**.

Hovoríme, že **problém má zložitosť nanajvýš**  $O(f(n))$ , ak preň existuje  $O(f(n))$  algoritmus.

Majme triediaci algoritmus  $\mathcal{A}$ , ktorý  $n$  čísel usporiada vzostupne podľa veľkosti tak, že v kroku  $i$  nájde z množiny nezaradených čísel to najmenšie a to zaradí na miesto  $i$ . Algoritmus  $\mathcal{A}$  potrebuje v prvom kroku prezrieť  $n$  čísel, v druhom  $n - 1$  čísel atď. až v kroku  $n$  jedno číslo t. j. spolu  $n + (n - 1) + \dots + 1 = \frac{n \cdot (n + 1)}{2}$  čísel. Ak na prezretie jedného čísla potrebuje najviac  $K$  elementárnych krokov, potom musí urobiť najviac  $K \cdot \frac{n \cdot (n + 1)}{2}$  elementárnych krokov. Pre počet krokov  $T(n)$  algoritmu  $\mathcal{A}$  pre usporiadanie postupnosti dĺžky  $n$  platí

$$T(n) \leq K \cdot \frac{n \cdot (n + 1)}{2} \leq K \cdot n^2.$$







**Úloha bivalentného (alebo binárneho) lineárneho programovania** (úloha BLP) je nájsť takú  $n$ -ticu celých čísel  $\mathbf{x}$ , pre ktorú je  $\mathbf{c}^T \cdot \mathbf{x}$  minimálne za predpokladov  $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ ,  $x_i \in \{0, 1\}$  pre  $i = 1, 2, \dots, n$ .

Úloha celočíselného lineárneho programovania je už podstatne ťažšia ako úloha lineárneho programovania. Úloha bivalentného programovania je špeciálnym prípadom úlohy celočíselného lineárneho programovania avšak rovnako patrí medzi najťažšie úlohy kombinatorickej matematiky. Existujú metódy a programy na riešenie oboch typov úloh, rozsah riešených príkladov je však podstatne menší, než príkladov lineárneho programovania, pri ktorom sa nežiada celočíselnosť, resp. bivalentnosť riešenia.

## 2.3 Polynomiálne transformácie

Častým postupom pri riešení nejakého problému je previesť ho na jedno alebo viac riešení iného problému. Tak napríklad násobenie dvoch celých čísel vieme previesť na sériu sčítaní. Hľadanie najväčšieho spoločného deliteľa dvoch čísel prevedieme na postupnosť delení so zvyškom.

Na tomto mieste trochu predbehneme výklad teórie grafov a ukážeme si, ako možno úlohu hľadania najkratšej  $u$ - $v$  cesty v hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$ , kde  $c(h) > 0$ , previesť na úlohu bivalentného programovania. Uvedieme najprv niekoľko definícií:  $u$ - $v$  cestou v digrafe  $\vec{G} = (V, H, c)$  nazveme ľubovoľnú striedavú postupnosť  $\mu(u, v)$  vrcholov a hrán digrafu  $\vec{G}$  typu

$$(u = v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \dots, (v_{k-1}, v_k), v_k = v),$$

v ktorej sa žiaden vrchol neopakuje. Dĺžka  $u$ - $v$  cesty  $\mu(u, v)$  je súčet ohodnotení jej hrán. Najkratšia  $u$ - $v$  cesta v digrafe  $\vec{G}$  je tá  $u$ - $v$  cesta, ktorá má zo všetkých  $u$ - $v$  ciest najmenšiu dĺžku.<sup>1</sup>

Označme:

$$c_{ij} = \begin{cases} c(i, j) & \text{ak } (i, j) \in H \\ \infty & \text{ak } (i, j) \notin H \end{cases} \quad (2.2)$$

<sup>1</sup>Systematický výklad o spôsoboch cestovania po grafoch a digrafoch nájde čitateľ v kapitole 3.

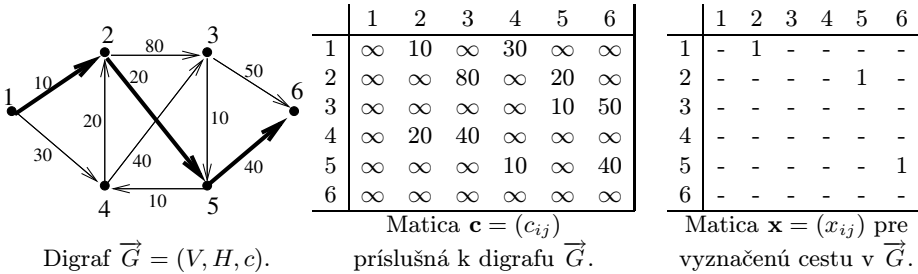
Nech  $x_{ij}$  je bivalentná premenná, ktorá nadobúda hodnoty nasledovne:

$$x_{ij} = \begin{cases} 1 & \text{ak } u-v \text{ cesta obsahuje hranu } (i, j) \\ 0 & \text{inak} \end{cases} \quad (2.3)$$

Predstavme si, že štvorcová matica typu  $n$  ( $x_{ij}$ ) zložená iba z núl a jedničiek zodpovedá nejakej ceste  $\mu(u, v)$  v digrafe  $\vec{G}$ . Potom pre ľubovoľné  $(i, j) \in V \times V$  je  $c_{ij} \cdot x_{ij}$  rovné dĺžke hrany  $(i, j)$  ak hrana  $(i, j)$  leží na ceste  $\mu(u, v)$ , alebo nula inak. Z toho môžeme usúdiť, že

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (2.4)$$

predstavuje dĺžku cesty  $\mu(u, v)$ . Máme už lineárnu kritériálnu funkciu, ktorú chceme minimalizovať, lebo hľadáme najkratšiu cestu. Korešpodenciu medzi cestou  $\mu(u, v)$  a maticou  $\mathbf{x}$  ilustruje obrázok 2.1.



Obr. 2.1:  
Digraf  $\vec{G}$  s cestou  $\mu(u, v)$  a matice  $\mathbf{c}$  a  $\mathbf{x}$ .

Ak máme maticu  $(x_{ij})$  zloženú z hodnôt 0 alebo 1, táto matica môže definovať ľubovoľnú množinu orientovaných hrán digrafu  $\vec{G}$ , dokonca aj množinu takých dvojíc, ktoré nie sú hranami digrafu  $\vec{G}$ . Aké ohraničenia položiť na  $(x_{ij})$ , aby definovali  $u-v$  cestu? V prvom rade vieme, že z  $u$  vychádza práve jedna hrana, do  $v$  vchádza práve jedna hrana, do  $u$  nevchádza žiadna hrana a z  $v$  nevychádza žiadna hrana čo vyjadríme nasledujúcimi rovnicami:

$$\sum_{j=1}^n x_{uj} = 1 \quad \sum_{i=1}^n x_{iv} = 1 \quad \sum_{j=1}^n x_{ju} = 0 \quad \sum_{i=1}^n x_{vi} = 0 \quad (2.5)$$

Ďalej využijeme skutočnosť, že pre každý vrchol rôzny od  $u$ ,  $v$  do vrchola vchádza práve toľko hrán, koľko z neho vychádza:

$$\sum_{j=1}^n x_{kj} = \sum_{i=1}^n x_{ik} \quad \forall k \in V, k \neq u, k \neq v. \quad (2.6)$$

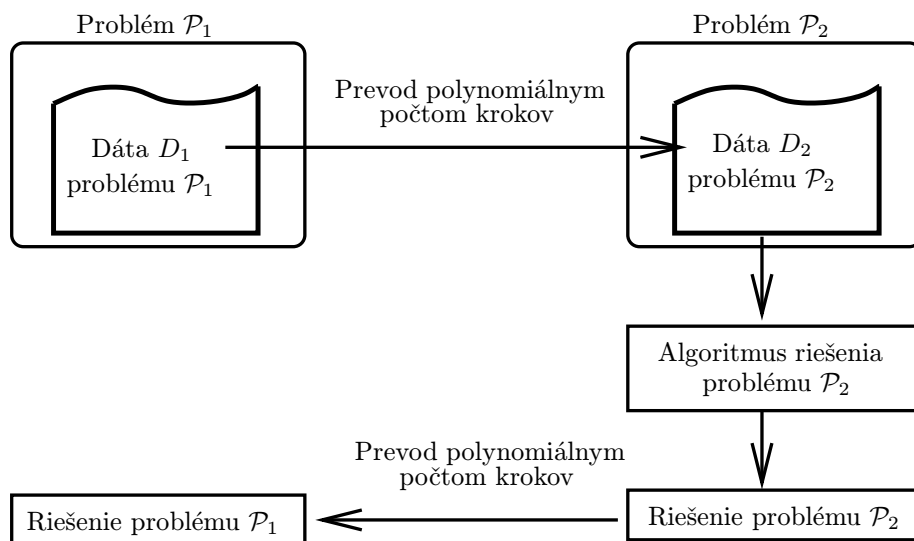
Ak by niekto namietal, že rovnice (2.5), (2.6) ešte nestačia na to, aby  $(x_{ij})$  definovali  $u$ - $v$  cestu, má pravdu. Okrem nej môžu hrany definované premennými  $(x_{ij})$ , ktoré splňujú (2.5), (2.6), tvoriť ešte niekoľko kružníc. Avšak kladná cena hrán digrafu  $\vec{G}$  zaručuje, že pri minimalizácii funkcie (2.4) nebude v optimálnom riešení žiadna zbytočná hrana, pretože by zvyšovala hodnotu kriteriálnej funkcie. Najšť najkratšiu cestu v digrafe  $\vec{G}$  teda znamená najšť také  $(x_{ij})$ , aby hodnota (2.4) bola minimálna za predpokladov (2.5), (2.6),  $x_{ij} \in \{0, 1\}$ .

Úlohu hľadania najkratšej cesty v digrafe sme previedli na úlohu bivalentného programovania, ktorú vyriešime a jej výsledky prevedieme na najkratšiu cestu v pôvodnom digrafe. Aká je výpočtová náročnosť tohto prevodu? Rovnice (2.5) a (2.6) sú rovnicami typu  $\sum_{i=1}^n a_{ij}x_{ij} = b_i$  o  $n^2$  neznámych  $x_{ij}$  – pre každú treba určiť  $n^2$  koeficientov. Najprv musíme určiť  $n^2$  hodnôt  $c_{ij}$  a potom po  $n^2$  hodnôt koeficientov pre každú rovnicu typu (2.5), (2.6), ktorých je  $2+(n-2) = n$ . Prevod úlohy hľadania najkratšej cesty na úlohu bivalentného programovania teda vyžaduje  $O(n^3)$  krokov. Prevod výsledku bivalentného programovania vyžaduje prekontrolovať  $n^2$  hodnôt matice  $(x_{ij})$  – má teda zložitosť  $O(n^2)$ . (Pritom netvrdíme, že uvedený spôsob je najefektívnejší, stačí, že je polynomiálny).

Po takomto prevode môžeme povedať, že úloha hľadania najkratšej cesty v digrafe je v istom zmysle ľahšia ako úloha bivalentného programovania, pretože problém najkratšej cesty možno formulovať ako špeciálny prípad úlohy bivalentného programovania.

Nech je daný problém  $\mathcal{P}_1$  so vstupnými dátami  $D_1$ . Problém  $\mathcal{P}_1$  môžeme riešiť aj tak, že ho pretransformujeme na iný problém  $\mathcal{P}_2$  tak, že dáta  $D_1$  prerobíme na dáta  $D_2$  problému  $\mathcal{P}_2$ . Ak riešenie  $R_2$  problému  $\mathcal{P}_2$  vieme prerobiť na riešenie  $R_1$  problému  $\mathcal{P}_1$ , transformácia je hotová. Ak prerobenie dát  $D_1$  na  $D_2$  a riešenia  $R_2$  na  $R_1$  vyžaduje len polynomiálny počet elementárnych operácií, nazveme túto transformáciu **polynomiálnou transformáciou**.

Ak problém  $\mathcal{P}_1$  možno polynomiálne transformovať na problém  $\mathcal{P}_2$  a problém  $\mathcal{P}_2$  má polynomiálny algoritmus riešenia, potom aj problém  $\mathcal{P}_1$  má polynomiálny algoritmus riešenia, pretože prevod dát  $D_1$  na dáta  $D_2$ , získanie riešenia  $R_2$

Obr. 2.2: Polynomiálna transformácia problému  $\mathcal{P}_1$  na problém  $\mathcal{P}_2$ .

problému  $\mathcal{P}_2$  s dátami  $D_2$  a prevod riešenia  $R_2$  na riešenie  $R_1$  problému  $\mathcal{P}_1$  možno urobiť polynomiálnym počtom krokov.

V niektorých prípadoch vyriešenie problému  $\mathcal{P}_1$  vyžaduje vyriešiť niekoľko prípadov problému  $\mathcal{P}_2$  (napr. násobenie prevádzame na niekoľko sčítaní). Ak prevody dát a riešení vyžadujú len polynomiálny počet elementárnych operácií a výpočet vyžaduje polynomiálny počet výpočtov problému  $\mathcal{P}_2$ , hovoríme, že sme problém  $\mathcal{P}_1$  **polynomiálne redukovali na problém  $\mathcal{P}_2$** . Ak problém  $\mathcal{P}_1$  možno polynomiálne redukovať na problém  $\mathcal{P}_2$  a naopak, problém  $\mathcal{P}_2$  možno polynomiálne redukovať na  $\mathcal{P}_1$  hovoríme, že problémy  $\mathcal{P}_1, \mathcal{P}_2$  sú **polynomiálne ekvivalentné**.

## 2.4 NP-ťažké úlohy

Kombinatorické úlohy, ku ktorým patria aj úlohy teórie grafov, spočívajú vo výbere jedného z konečného počtu kombinatorických objektov. Najjednoduchším spôsobom riešenia takýchto úloh je úplné prehľadanie všetkých objektov (full search, brute force). Klasická matematika takéto problémy skutočne rieši

takto – prehľadanim konečného počtu prvkov nájdeme ten hľadaný. Pozrime sa, koľko objektov treba prehľadať pri riešení niektorých problémov:

		$n = 10$	$n = 50$	$n = 100$
Počet štvorcových matic $n \times n$	$n^2$	100	2500	1.0e+4
Počet podmnožín $n$ -prvkovej množiny	$2^n$	1024	1.1e+15	1.2e+30
Počet permutácií z $n$ prvkov	$n!$	3.6e+6	3.0e+64	9.3e+157
Počet všetkých zobrazení $n$ -prvkovej množiny do seba	$n^n$	1.0e+10	8.9e+84	1.0e+200

Rok má 31.54 milióna sekúnd, doterajší vek vesmíru od Veľkého tresku sa odhaduje na  $4.7e+17$  sekúnd, a ešte asi  $1.0e+18$  sekúnd nám ostáva do jeho konca. Pri porovnaní týchto hodnôt s počtami z predchádzajúcej tabuľky vidíme, že ani najrýchlejší počítač by nám nenašiel optimálnu permutáciu päťdesiatich prvkov do konca existencie vesmíru. Tu vidíme, že jediná šancu na praktické využitie pre úlohy väčšieho rozsahu majú algoritmy s polynomiálnou zložitou.

Edmonds (1965) bol prvý, ktorý si uvedomil rozdiel medzi polynomiálnymi algoritmi (t. j. algoritmi so zložitou typu  $O(n^k)$ ) a nepolynomiálnymi algoritmi, ktorých počet krokov nevieme ohraničiť polynómom. Tie prvé nazýva „dobré“. Podľa neho aj problémy možno rozdeliť na také, pre ktoré existuje polynomiálny algoritmus a tie ostatné – „beznádejné“, pre ktoré takýto algoritmus neexistuje. Túto ideu sa zatiaľ nepodarilo plne realizovať, pretože neexistencia polynomiálneho algoritmu sa pre väčšinu problémov považovaných za „beznádejné“ nepodarilo dokázať.

Ak úlohu  $\mathcal{P}_1$  možno polynomiálne redukovat' na úlohu  $\mathcal{P}_2$ , znamená to nasledovné: Ak existuje polynomiálny algoritmus riešenia úlohy  $\mathcal{P}_2$ , potom existuje aj polynomiálny algoritmus pre úlohu  $\mathcal{P}_1$ . Naopak to však nemusí platiť – ak existuje polynomiálny algoritmus pre  $\mathcal{P}_1$ , polynomiálna redukovateľnosť  $\mathcal{P}_1$  na  $\mathcal{P}_2$  ešte nič nehovorí o zložitosti problému  $\mathcal{P}_2$ .

Ak sú však problémy  $\mathcal{P}_1, \mathcal{P}_2$  polynomiálne ekvivalentné, existencia polynomiálneho algoritmu pre jeden implikuje existenciu polynomiálneho algoritmu pre druhý.

Ako etalón ťažkých úloh bola vybratá úloha bivalentného lineárneho programovania (BLP). Problém, ktorý možno polynomiálne redukovat' na úlohu BLP, je ľahší alebo rovnako ťažký ako úloha BLP. Problém, na ktorý možno polynomiálne redukovat' úlohu BLP je ťažší alebo rovnako ťažký ako úloha BLP.

**Definícia 2.4.** Hovoríme, že **problém  $\mathcal{P}$  je NP–ťažký**, ak úlohu bivalentného lineárneho programovania možno polynomiálne redukovať na  $\mathcal{P}$ .

Hovoríme, že **problém  $\mathcal{P}$  je NP–ľahký**, ak problém  $\mathcal{P}$  možno polynomiálne redukovať na úlohu bivalentného lineárneho programovania.

Hovoríme, že **problém  $\mathcal{P}$  je NP–ekvivalentný**, ak je problém  $\mathcal{P}$  polynomiálne ekvivalentný s úlohou bivalentného lineárneho programovania.

Doteraz sa podarilo pre stovky úloh dokázať, že sú NP–ekvivalentné. Pre ďalšie sa podarilo dokázať, že sú NP–ťažké. Pre žiadnu NP–ťažkú úlohu sa doteraz nepodarilo nájsť polynomiálny algoritmus. Nepodarilo sa však ani dokázať, že polynomiálny algoritmus pre ne neexistuje. Nájdenie takého algoritmu pre jedinú zo stoviek NP–ekvivalentných úloh by znamenalo existenciu polynomiálneho algoritmu pre všetky ostatné. Všeobecne sa neverí, že by sa to mohlo niekomu podať, preto sú na NP–zložitosti niektorých úloh založené napr. aj niektoré kryptografické systémy.

Niektori by na tomto mieste mohli namietat: „Čo mám z algoritmu, aj keď je polynomiálny, keď je zložitosti napríklad  $O(n^{1000})$ ? Veď sa pri ňom nedočkám výsledku ani pri dvoch vstupných údajoch.“ Táto argumentácia je v poriadku, avšak našťastie v praxi to býva tak, že keď už existuje polynomiálny algoritmus, potom jeho zložitosť zriedkakedy presiahne  $O(n^6)$ .

## 2.5 Aproximácia

Mnohé problémy spojené s rozvrhovaním výroby, plánovaním pristávania lietadiel, navrhovaním okružných jász vozidiel, rozhodovaním o umiestnení skladov, pridelovaní frekvencií vysielacom a stoviek (možno tisícok) ďalších sú NP–ťažké. Sme v prípade NP–ťažkých úloh celkom bezmocní?

V prvom rade treba povedať, že v praxi nám stačí, aby sme dostali dobré riešenie, ktoré, aj keď nebude práve optimálne, bude blízko optimálnemu. Algoritmy, ktorými hľadáme riešenie blízke optimálnemu, nazývame aproximačné algoritmy alebo heuristiky.

**Definícia 2.5.** Nech  $\mathcal{P}$  je minimalizačný problém s kladnou kritériálnou funkciou  $f$ ,  $\rho > 1$ . Hovoríme, že **algoritmus  $\mathcal{A}$  je  $\rho$ –aproximačný algoritmus**, ak pre každý prípad problému  $\mathcal{P}$  jeho výsledkom je riešenie s hodnotou krite-

riálnej funkcie  $f_{\mathcal{A}}$ , pre ktorú platí

$$\frac{f_{\mathcal{A}}}{f_{\text{OPT}}} \leq \rho, \quad (2.7)$$

kde  $f_{\text{OPT}}$  je hodnota kriteriálnej funkcie optimálneho riešenia daného prípadu.<sup>2</sup>

Pre niektoré NP-ťažké problémy existuje  $\rho_0$ -aproximačný polynomiálny algoritmus, ale pre  $1 \leq \rho < \rho_0$  neexistuje polynomiálny  $\rho$ -aproximačný algoritmus. Existujú problémy, pre ktoré je každá  $\rho$ -aproximačná úloha NP-ťažká (napr. úloha bivalentného lineárneho programovania s kladným minimom.) Na druhej strane pre iné úlohy existuje polynomiálny  $\rho$ -aproximačný algoritmus pre každé  $\rho > 1$  (napr. problém binárneho batohu<sup>3</sup>).

Pre mnohé dobré heuristiky sa nepodarí ukázať, ako je nimi vypočítané riešenie ďaleko od skutočného optima a napriek tomu dávajú v praxi dobré riešenie. U mnohých iných sa skutočný pomer  $f_{\mathcal{A}}/f_{\text{OPT}}$  pohybuje v praktických prípadoch hlboko pod hranicou  $\rho$ . Tej sa blížia len výsledky riešenia špeciálnych „zlomyselých“ príkladov.

## 2.6 Heuristiky

V predchádzajúcej časti sme hovorili o heuristikách ako o postupoch či algoritmoch, ktoré dávajú riešenie s hodnotou kriteriálnej funkcie blízku k optimálnej hodnote (presnejšie k hodnote kriteriálnej funkcie optimálneho riešenia). V tejto časti spomenieme niektoré veľmi jednoduché heuristické postupy, ktoré dávajú v mnohých prípadoch uspokojivé riešenia.

Vo všeobecnosti možno heuristiky rozdeliť na **vytvárajúce** a **zlepšujúce**. Vytvárajúca heuristika postupne vytvára riešenie (napríklad najpočetnejšiu

<sup>2</sup>Pre maximalizačnú úlohu s kladnou kriteriálnou funkciou  $f$  má podmienka (2.7) tvar  $\frac{f_{\text{OPT}}}{f_{\mathcal{A}}} \leq \rho$ .

<sup>3</sup>Problém batohu (The knapsack problem) je nasledujúca úloha BLP: Maximalizovať  $\sum_{i=1}^n c_i x_i$  za predpokladov  $\sum_{i=1}^n a_i x_i \leq b$ ,  $x_i \in \{0, 1\}$ . Zlodej má batoh s nosnosťou  $b$  kg. Do batohu môže vložiť  $n$  predmetov, pričom  $i$ -tý predmet váži  $a_i$  kg a má hodnotu  $c_i$ . Ktoré predmety má zlodej vložiť do batoha tak, aby neprekročil jeho nosnosť  $b$  a odniesol predmety s čo najväčším súčtom hodnôt? Nech binárna premenná  $x_i = 1$ , ak zlodej vloží do batoha predmet  $i$ , inak  $x_i = 0$ . Potom hodnota všetkých predmetov v batohu je  $\sum_{i=1}^n c_i x_i$  a podmienka nepreťažitého batohu je  $\sum_{i=1}^n a_i x_i \leq b$ . Na úlohu tohto typu vedú aj početnejšie aplikácie.



podmnožinu vrcholov, z ktorej žiadne dva vrcholy nie sú susedné) tak, že začne najjednoduchším objektom (v našom prípade prázdnu množinou vrcholov), a potom tento objekt rozširuje podľa nejakého pravidla, ktoré dáva nádej, že výsledný objekt bude blízko optimálnemu. Najjednoduchším pravidlom je rozšíriť objekt o prvok (v teórii grafov najčastejšie o vrchol alebo hranu), ktorým sa dosiahne najlepšia kritériálna funkcia rozšíreného objektu. Takúto metódu nazývame **pažravou metódou – greedy algorithm**.

K vytvárajúcim metódam možno priradiť i pravdepodobnostné metódy, ktoré priradujú k vytváranému objektu prvky podľa náhodného generátora a tak vygenerujú pseudonáhodné prípustné riešenie. Keďže vytvorenie takéhoto riešenia obyčajne trvá veľmi krátko, je možné počítačom vytvoriť veľké množstvo takýchto riešení, pre každé vypočítať hodnotu kritériálnej funkcie a zapamätať si najlepšie dosiahnuté riešenie.

Zlepšujúce heuristiky vychádzajú z existujúceho prípustného riešenia a snažia sa z neho jednoduchými operáciami dostať iné blízke tzv. susedné riešenie s lepšou hodnotou kritériálnej funkcie. Tak pokračujú dovtedy, kým nedospejú k lokálnemu optimu, t. j. k takému riešeniu, ktoré už nemá žiadne lepšie susedné riešenie. Táto metóda sa nazýva **metóda prehľadávania okolí – neighborhood search**. Metódu prehľadávania okolí možno kombinovať s pravdepodobnostnou metódou tak, že ju spustíme na niekoľko náhodne vytvorených štartovacích riešení a za výsledok berieme to najlepšie z nájdených lokálnych optimálnych riešení.

V dnešnej dobe sme svedkami veľkého množstva vyspelých heuristických postupov ako napríklad **tabu search**, **simulated anealing**, **genetické algoritmy**, či rôzne modifikácie **metódy vetiev a hraníc**, ktoré však už nepatria do tejto publikácie. O tejto problematike však existuje rozsiahla literatúra, napríklad [14].

## 2.7 Cvičenia

Pri nasledujúcich cvičeniach považujte sčítanie, odčítanie, násobenie, delenie a porovnanie čísel za elementárne operácie bez ohľadu na veľkosť operandov.

1. Určte zložitosť sčítania a násobenia štvorcových matíc typu  $n$ .
2. Aká je zložitosť výpočtu determinantu štvorcovej matice typu  $n$  podľa definície a podľa vety o rozvoji determinantu na subdeterminanty? Navrhnite lepší algoritmus na výpočet determinantu.
3. Aká je zložitosť výpočtu inverznej matice k štvorcovej matici typu  $n$ ?
4. Aká je zložitosť úpravy štvorcovej matice typu  $n$  na diagonálny tvar?
5. Navrhnite niekoľko algoritmov na usporiadanie  $n$  čísel a stanovte ich zložitosť.
6. Vyberte si niektorý zo známych algoritmov na zisťovanie prvočíselnosti čísla  $n$  a stanovte jeho zložitosť v závislosti na  $n$ .
7. Aká je zložitosť Euklidovho algoritmu na zisťovanie najväčšieho spoločného deliteľa čísel  $m$ ,  $n$ ?

# Kapitola 3

## Cesty v grafoch

### 3.1 Sledy, ťahy a cesty v grafoch a digrafoch

Grafy a digrafy veľmi často modelujú dopravné, spojovacie, počítačové či elektrifikačné siete. Preto potrebujeme popísať spôsob, akým po sieti prechádza dopravný prostriedok, telegrafická správa, dátový paket či elektrická energia. Spôsob prechodu všetkých spomínaných objektov po sieti je veľmi podobný – je možný len tak, že z vrchola siete sa objekt dostane na niektorú s ním incidentnú hranu a dôjde do druhého vrchola hrany, odtiaľ do ďalšej incidentnej hrany atď. Objekty na takýchto sieťach nemajú dovolené „preskakovať“ z vrchola alebo hrany na nesusedný vrchol alebo neincidentnú hranu. Spôsoby prechádzania sieťou sa rozlišujú podľa toho, či je alebo nie je dovolené ísť po jednom úseku viackrát a podľa toho, či je alebo nie je dovolené navštíviť jeden vrchol viackrát. V digrafoch k spomínanému rozlišovaniu pristupuje i hľadisko, či je alebo nie je dovolené ísť hranou proti jej orientácii. Presnejšie nasledujúce definície.

**Definícia 3.1.** Nech  $G = (V, H)$  je graf. **Sled** v grafe  $G$  je ľubovoľná alternujúca (striedavá) postupnosť vrcholov a hrán tvaru

$$\mu(v_1, v_k) = (v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \dots, \{v_{k-1}, v_k\}, v_k). \quad (3.1)$$

**Ťah** v grafe  $G$  je taký sled v grafe  $G$ , v ktorom sa žiadna hrana neopakuje. **Cesta** v grafe  $G$  je taký sled v grafe  $G$ , v ktorom sa žiaden vrchol neopakuje. Pripúšťame aj tzv. **triviálny sled**, pre  $k = 1$ , t. j. sled tvaru  $(v_1)$ .

Nech  $\vec{G} = (V, H)$  je digraf. **Orientovaný sled** v digrafe  $\vec{G}$  je ľubovoľná alternujúca (striedavá) postupnosť vrcholov a hrán tvaru

$$\mu(v_1, v_k) = (v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \dots, (v_{k-1}, v_k), v_k). \quad (3.2)$$

**Orientovaný ťah** v digrafe  $\vec{G}$  je taký orientovaný sled v digrafe  $\vec{G}$ , v ktorom sa žiadna hrana neopakuje. **Orientovaná cesta** v digrafe  $\vec{G}$  je taký orientovaný sled v digrafe  $\vec{G}$ , v ktorom sa žiaden vrchol neopakuje.

**Definícia 3.2.** Nech  $\vec{G} = (V, H)$  je digraf. **Polosled** v digrafe  $\vec{G}$  je ľubovoľná alternujúca (striedavá) postupnosť vrcholov a hrán tvaru

$$\mu(v_1, v_k) = (v_1, h_1, v_2, h_2, \dots, v_{k-1}, h_{k-1}, v_k),$$

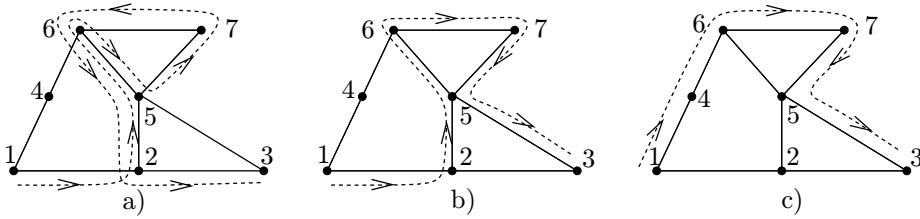
v ktorej je každá hrana  $h_i$  incidentná s oboma susednými vrcholmi  $v_i, v_{i+1}$  tak, že jeden z nich je začiatočným a druhý koncovým vrcholom hrany  $h$ . **Polotáh** v digrafe  $\vec{G}$  je taký polosled v digrafe  $\vec{G}$ , v ktorom sa žiadna hrana neopakuje. **Polocesta** v digrafe  $\vec{G}$  je taký polosled v digrafe  $\vec{G}$ , v ktorom sa žiaden vrchol neopakuje.

Pri digrafoch budeme používať aj skrátene termíny *sled*, (*ťah*, *cesta*) namiesto *orientovaný sled*, (*orientovaný ťah*, *orientovaná cesta*). Tu je totiž z kontextu jasné, o čo sa jedná. V niektorej slovenskej literatúre sa používa pre tieto pojmy *orsled*, *ortáh* a *dráha*, ale širšie sa táto terminológia neujala. V anglickej literatúre sa pre sled, ťah a cestu používajú termíny **walk**, **trail** a **path** (avšak nejednotne, podobne ako v našej slovenskej literatúre).

Čitateľ by sa teraz mohol opýtať, prečo nestačí definovať sled (a potom aj z neho odvodené pojmy) ako postupnosť vrcholov. Veď v grafoch a digrafoch je postupnosťou vrcholov, s ktorých sú vždy dva susedné, jednoznačne definovaný sled.

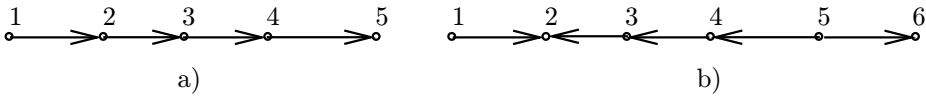
Prvým dôvodom našej definície je, aby sme mohli hovoriť, že sled  $\mu(u, v)$  obsahuje hranu  $h$ . Pretože sled je postupnosťou vrcholov a hrán, má význam hovoriť, že hrana  $h$  je alebo nie je členom tejto postupnosti. Ak by sme sled definovali len ako postupnosť susedných vrcholov, mali by sme tu problémy hneď pri definícii ťahu.

Druhým dôvodom je, že polosled v digrafe by už nebol jednoznačne definovaný postupnosťou vrcholov, pretože každej hrane musíme povedať, či má byť použitá v smere alebo protismeru svojej orientácie – digraf totiž môže s hranou  $(u, v)$  obsahovať aj hranu  $(v, u)$ .



Obr. 3.1: Sled, ťah a cesta v grafe.

- a) 1–3 sled:  $(1, \{1, 2\}, 2, \{2, 5\}, 5, \{5, 6\}, 6, \{6, 5\}, 5, \{5, 7\}, 7, \{7, 6\}, 6, \{6, 5\}, 5, \{5, 2\}, 2, \{2, 3\}, 3)$ .  
 b) 1–3 ťah:  $(1, \{1, 2\}, 2, \{2, 5\}, 5, \{5, 6\}, 6, \{6, 7\}, 7, \{7, 5\}, 5, \{5, 3\}, 3)$ .  
 c) 1–3 cesta:  $(1, \{1, 4\}, 4, \{4, 6\}, 6, \{6, 7\}, 7, \{7, 5\}, 5, \{5, 3\}, 3)$ .



Obr. 3.2: Orientovaná cesta a polocesta v digrafe.

- a) 1–5 orientovaná cesta:  $(1, (1, 2), 2, (2, 3), 3, (3, 4), 4, (4, 5), 5)$ .  
 b) 1–6 polocesta:  $(1, (1, 2), 2, (3, 2), 3, (4, 3), 4, (4, 5), 5, (5, 6), 6)$ .

Tretím dôvodom, prečo sme prijali definíciu sledu v tomto tvare, je snaha o jej kompatibilitu aj pre multigrafy a multidigrafy. Tam už postupnosť vrcholov neurčuje jednoznačne sled, pretože medzi dvoma vrcholmi môže existovať niekoľko „rovnobežných“ hrán a vtedy treba povedať, ktorú z nich sme do sledu vybrali.

V prípadoch, kedy nedôjde k nedorozumeniu, pripustíme v grafoch a digrafoch aj skrátenejší zápis sledu (3.1), resp. (3.2) v tvare

$$(v_1, v_2, \dots, v_k), \quad (3.3)$$

stále však budeme sled považovať za postupnosť vrcholov a hrán.

Ak  $v_1 = u$  a  $v_k = v$ , t. j. ak prvým vrcholom sledu je  $u$  a posledným  $v$ , budeme hovoriť, že ide o sled z  $u$  do  $v$ , alebo skrátene  $u$ - $v$  sled. (Podobne  $u$ - $v$  ťah,  $u$ - $v$  cesta,  $u$ - $v$  polosled,  $u$ - $v$  polofah,  $u$ - $v$  polocesta).

**Definícia 3.3.** Sled (polosled, ťah, polofah)

$$\mu(u, v) = (v_1, h_1, v_2, h_2, \dots, v_{k-1}, h_{k-1}, v_k)$$

nazveme **uzavretý**, ak  $v_1 = v_k$ .

Inak sled (polosled, ťah, poloťah)  $\mu(u, v)$  nazveme **otvorený**.

*Poznámka.* Uzavretú cestu a polocestu nemožno týmto spôsobom definovať, pretože by došlo k sporu s požiadavkou, že jeden vrchol sa v týchto štruktúrach nesmie vyskytovať viackrát. Namiesto uzavretej cesty a polocesty budeme mať cyklus a polocyklus. Presná definícia týchto pojmov je nasledujúca:

**Definícia 3.4.** **Cyklus (orientovaný cyklus, polocyklus)** je uzavretý ťah (orientovaný ťah, poloťah), v ktorom sa okrem prvého a posledného vrchola žiaden vrchol nevyskytuje viac než raz.

**Definícia 3.5.** Nech

$$\begin{aligned}\mu(v_1, v_r) &= (v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \dots, \{v_{r-1}, v_r\}, v_r), \\ \mu(w_1, w_s) &= (w_1, \{w_1, w_2\}, w_2, \{w_2, w_3\}, w_3, \dots, \{w_{s-1}, w_s\}, w_s),\end{aligned}$$

nech  $v_r = w_1$ . **Zreťazením sledov**  $\mu(v_1, v_r)$ ,  $\mu(w_1, w_s)$  nazveme sled

$$\begin{aligned}\mu(v_1, v_r) \oplus \mu(w_1, w_s) &= \\ &= (v_1, \{v_1, v_2\}, v_2, \dots, \{v_{r-1}, v_r\}, v_r = w_1, \{w_1, w_2\}, w_2, \dots, \{w_{s-1}, w_s\}, w_s).\end{aligned}$$

Zreťazenie orientovaných sledov a polosledov definujeme analogicky.

*Poznámka.* Zreťazenie  $\mu(u, w) \oplus \mu(w, v)$  dvoch ciest  $\mu(u, w)$   $\mu(w, v)$  nemusí byť cesta, vo všeobecnosti môžeme dostať sled.

**Definícia 3.6.** Nech  $G = (V, H)$  je graf, resp. digraf, nech  $u, v \in V$ . Hovoríme, že vrchol  $v$  je **dosiahnuteľný** z vrchola  $u$  v grafe, resp. digrafe  $G$ , ak v grafe, resp. digrafe  $G$  existuje  $u$ - $v$  sled, resp.  $u$ - $v$  orientovaný sled.

**Veta 3.1.** Ak v grafe  $G = (V, H)$  existuje  $u$ - $v$  sled pre niektoré  $u, v \in V$ ,  $u \neq v$ , potom v ňom existuje aj  $u$ - $v$  cesta.

DÔKAZ.

Nech  $\mu(u, v)$  je  $u$ - $v$  sledom. Ak sa v ňom žiaden vrchol neopakuje,  $\mu(u, v)$  je cestou. Nech

$$(u = v_1, \{v_1, v_2\}, v_2, \dots, \{v_{k-1}, v_k\}, v_k, \underbrace{\{v_k, v_{k+1}\}, \dots, v_l}_{\text{časť na vylúčenie}}, \{v_l, v_{l+1}\}, \dots, \{v_{r-1}, v_r\}, v_r = v)$$

je  $u$ - $v$  sled, kde sa nejaký vrchol  $w$  vyskytuje viackrát. Nech  $v_k$  je prvý výskyt,  $v_l$  posledný výskyt vrchola  $w$  v slede, t. j.  $v_k = w$ ,  $v_l = w$ . Vytvorme nový sled vynechaním časti daného sledu začínajúc hranou  $\{v_k, v_{k+1}\}$  a končiac vrcholom  $v_l$  včítane. Dostaneme postupnosť vrcholov a hrán

$$(u = v_1, \{v_1, v_2\}, v_2, \dots, \{v_{k-1}, v_k\}, v_k = v_l, \{v_l, v_{l+1}\} \dots, \{v_{r-1}, v_r\}, v_r = v).$$

Pretože  $v_k = v_l = w$ , je posledná postupnosť sledom, v ktorom sa vrchol  $w$  vyskytuje len raz. Uvedený postup opakujeme dovtedy, kým nevytlúčime všetky viacnásobné výskyty vrcholov. ■

**Definícia 3.7.** Nech  $\mu(u, v)$  je sled, resp. polosled v grafe  $G$ . Označme  $V_{\mu(u, v)}$  množinu všetkých vrcholov sledu  $\mu(u, v)$ ,  $H_{\mu(u, v)}$  množinu všetkých hrán sledu  $\mu(u, v)$ . Potom štruktúra  $G_{\mu(u, v)} = (V_{\mu(u, v)}, H_{\mu(u, v)})$  je grafom, ktorý nazveme **graf indukovaný sledom  $\mu(u, v)$** .

## 3.2 Súvislosť grafov

**Definícia 3.8.** Hovoríme, že graf  $G = (V, H)$  je **súvislý**, ak pre každú dvojicu vrcholov  $u, v \in V$  existuje  $u$ - $v$  cesta. Inak hovoríme, že graf  $G$  je **nesúvislý**.

V definícii 1.16 (str.22) sme definovali pojem maximálneho podgrafu grafu  $G$  s vlastnosťou  $\mathcal{V}$ . Tento pojem teraz využijeme v nasledujúcej definícii.

**Definícia 3.9. Komponent grafu  $G = (V, H)$**  je jeho ľubovoľný maximálny súvislý podgraf.

**Definícia 3.10. Mostom** v grafe  $G = (V, H)$  nazveme takú hranu grafu  $G$ , po vylúčení ktorej vzrastie počet komponentov. **Artikuláciou** v grafe  $G$  nazveme taký vrchol, po vylúčení ktorého spolu s incidentnými hranami vzrastie počet komponentov.

**Definícia 3.11. Kružnica** je pravidelný súvislý graf 2. stupňa. Kružnicu o  $n$  vrcholoch budeme označovať  $C_n$ .

**Veta 3.2.** *Kružnica  $C_n = (V, H)$  má  $n$  hrán.*

DŮKAZ.

Platí:  $\sum_{v \in V} \deg(v) = 2 \cdot |H|$  Pretože  $\deg(v) = 2$  pre každý vrchol  $v \in V$ , platí:

$$2 \cdot |H| = \sum_{v \in V} \deg(v) = \sum_{v \in V} 2 = 2 \cdot |V|.$$

Z posledného vzťahu už vidíme, že  $|V| = |H|$  a teda  $|H| = n$ .

**Veta 3.3.** *Všetky vrcholy a hrany kružnice  $C_n$  možno usporiadať do cyklu*

$$(v_1, \{v_1, v_2\}, v_2, \dots, \{v_{n-1}, v_n\}, v_n, \{v_n, v_1\}, v_1).$$

DŮKAZ.

Zostrojme taký ťah

$$(v_1, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}, v_k) \quad (3.4)$$

v kružnici  $C_n$ , ktorý má najväčší počet hrán. Žiaden z vrcholov  $v_i$  pre  $i = 2, 3, \dots, k-1$  sa v ťahu (3.4) nevyskytuje dvakrát, inak by tento vrchol musel mať stupeň väčší alebo rovný 4.

Teraz ukážeme, že ak (3.4) je ťah s najväčším počtom hrán, potom musí byť uzavretý, t. j.  $v_1 = v_k$ . Nech  $v_1 \neq v_k$ . Keďže vrchol  $v_k$  má stupeň 2, musí existovať hrana  $\{v_k, v_l\} \neq \{v_{k-1}, v_k\}$ , kde vrchol  $v_l$  sa nerovná vrcholu  $v_k$  (lebo hrany grafu sú dvojice rôznych vrcholov), ani vrcholu  $v_{k-1}$  (lebo hrany  $\{v_k, v_l\}$ ,  $\{v_{k-1}, v_k\}$  sú rôzne), ani žiadnemu z vrcholov  $v_2, v_3, \dots, v_{k-2}$  (lebo taký vrchol by mal stupeň väčší alebo rovný 3). Preto sa hrana  $\{v_k, v_l\}$  v ťahu nevyskytuje. Ak pridáme hranu  $\{v_k, v_l\}$  k ťahu (3.4), dostaneme znovu ťah

$$(v_1, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}, v_k, \{v_k, v_l\}, v_l),$$

ktorý má väčší počet hrán, ako ťah (3.4). Z predpokladu, že ťah (3.4) je otvorený, sme dostali spor. Ak je ťah (3.4) s najväčším počtom hrán, potom musí byť uzavretý, t. j.  $v_1 = v_k$ , a teda (3.4) je cyklus.

Teraz treba ešte dokázať, že každý vrchol kružnice  $C_n$  leží na cykle (3.4). Všimnime si, že každý vrchol cyklu (3.4) je incidentný s práve dvoma hranami cyklu. Pretože stupne všetkých vrcholov sú rovné číslu 2, žiaden vrchol cyklu (3.4) už nie je susedný so žiadnym iným vrcholom neležiacim na tomto cykle, z čoho vyplýva, že graf  $G_t$  indukovaný cyklom (3.4) je maximálny súvislý podgraf kružnice  $C_n$  – t. j. jej komponent. Keďže  $C_n$  je súvislý graf, má iba jeden komponent, a teda  $G_t = C_n$ . ■

### 3.3 Typy súvislosti digrafov

**Definícia 3.12.** Nech  $\vec{G} = (V, H)$  je digraf. Povieme, že digraf  $\vec{G}$  je **slabo súvislý**, alebo **neorientovane súvislý** ak pre každú dvojicu vrcholov  $u, v \in V$

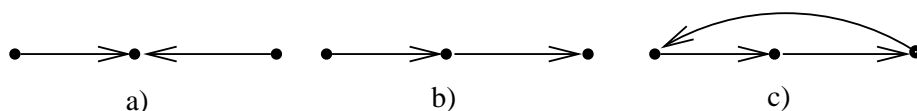


existuje v  $G$   $u-v$  polosled; inak je digraf  $\vec{G}$  **nesúvislý**.

Povieme, že digraf  $\vec{G}$  je **jednostranne súvislý**, alebo **orientovane súvislý**, ak pre každú dvojicu vrcholov  $u, v \in V$  existuje v  $\vec{G}$   $u-v$  sled alebo  $v-u$  sled.

Digraf  $\vec{G}$  je **silne súvislý**, ak pre každú dvojicu vrcholov  $u, v \in V$  existuje aj orientovaný  $u-v$  sled aj orientovaný  $v-u$  sled.

**Komponent digrafu**  $\vec{G}$  je maximálny neorientovane súvislý podgraf digrafu  $\vec{G}$ .



Obr. 3.3: Digrafy s rôznymi typmi súvislosti.

a) neorientovane súvislý    b) orientovane súvislý    c) silne súvislý

Poznamenajme ešte, že ak je digraf  $\vec{G}$  silne súvislý, potom je aj jednostranne, resp. orientovane súvislý, ak je digraf  $\vec{G}$  orientovane súvislý, potom je aj slabo súvislý. Preto digraf s diagramom c) na obrázku 3.3 je aj silne súvislý, aj jednostranne, resp. orientovane súvislý aj slabo súvislý. Podobne digraf s diagramom b) na tom istom obrázku je aj jednostranne súvislý aj slabo súvislý.

### 3.4 Tarryho prieskum grafov

V časti 3.2 sme definovali pojmy súvislosť grafu a komponent grafu. Stojíme teraz pred otázkou, ako zistiť komponenty daného grafu  $G$ . Pre jednoduché grafy s malým počtom vrcholov a hrán sa zdá, že komponenty grafu ihneď vidíme – zvlášť v prípadoch, keď máme k dispozícii jeho prehľadný diagram.

Predstavme si grafový model  $G = (V, H)$  cestnej siete s niekoľkými tisíckami vrcholov a hrán zadaný niektorým spôsobom popísaným v časti 1.3. Tu už odpoveď na otázku, či je príslušný graf súvislý, nie je triviálna a nepomôže nám ani diagram grafu, pretože v ňom nemusia byť jednotlivé komponenty zakreslené oddelene.

Z mytológie čitateľ iste pozná niekoľko príbehov o labyrintoch. Križovatky chodieb či komnaty labyrintu môžeme modelovať vrcholmi grafu  $G$ , chodby labyrintu budú hrany grafu  $G$ . Nešťastník stratený v labyrinte potrebuje nájsť

východ, ktorý môže byť na niektorej hrane alebo v niektorom vrchole grafu  $G$ . Nešťastník však nemá plán celého labyrintu, vie len, ktoré chodby vychádzajú z križovatky, v ktorej sa práve nachádza. Na to, aby našiel východ, potrebuje nešťastník postup, ktorý ho prevedie každým vrcholom a každou hranou labyrintu, pričom využije iba lokálnu informáciu o okolí vrchola, v ktorom práve stojí. Má nešťastník šancu?

Pri zisťovaní súvislosti grafu i pri úniku z labyrintu sme v podobnej situácii — z lokálnych informácií usúdiť na globálnu vlastnosť grafu. Jeden z najjednoduchších algoritmov uvádza Tarry (1895) a je známy pod názvom Tarryho prieskum grafov.

Všeobecnejšie pod prieskumom grafu rozumieme systematické usporiadanie (a teda aj prezretie) všetkých vrcholov alebo všetkých vrcholov a hrán grafu s využitím lokálnej informácie o okolíach vrcholov. Okrem Tarryho prieskumu budeme hovoriť ešte aj o prehľadávaní grafu do hĺbky a do šírky v kapitole 4.

**Algoritmus 3.1. Tarryho algoritmus** na konštrukciu takého sledu v grafe  $G = (V, H)$ , ktorý začína v ľubovoľnom vrchole  $s \in V$ , prejde všetkými hranami komponentu grafu  $G$  a skončí vo vrchole  $s$ . Výsledný sled budeme volať **Tarryho sled**.

- **Krok 1.** Začni z ľubovoľného vrchola  $s \in V$ , polož  $u := s$ ,  $T = (u)$ .  
{ $T$  je inicializačne triviálny sled, obsahujúci jediný vrchol.}
- **Krok 2.** Ak môžeš, vyber k poslednému vrcholu  $u$  sledu  $T$  ďalšiu incidentnú hranu  $\{u, v\}$  podľa nižšie uvedených pravidiel **T1**, **T2** a zaraď ju do sledu  $T$ . Zaznač si smer použitia hrany  $\{u, v\}$ . Ak doteraz vrchol  $v$  ešte nebol zaradený do sledu  $T$ , označ hranu  $\{u, v\}$  ako hranu prvého príchodu. Pri výbere hrany dodržiuj nasledujúce pravidlá:

**T1:** Každú hranu možno v jednom smere použiť iba raz

**T2:** Hranu prvého príchodu možno použiť, iba ak niet inej možnosti

- **Krok 3.** Ak taká hrana neexistuje – STOP.  
Inak polož  $u := v$  a pokračuj Krokom 2.



**Veta 3.4.** *Tarryho algoritmus skončí po konečnom počte krokov. Tarryho sled  $\mu(u, v)$  v súvislom grafe  $G = (V, H)$  je uzavretý a obsahuje každú hranu grafu  $G$  práve dvakrát.*

DŔKAZ.

Nech  $m = |H|$  je počet hrán grafu  $G$ . Pretože každú hranu možno použiť najviac dvakrát, Tarryho algoritmus najneskôr po pridaní  $2m$  hrán do sledu  $T = \mu(s, v)$  skončí.

Ukážeme, že Tarryho sled je uzavretý. Predpokladajme, že sled  $T = \mu(s, v)$  zostrojený Tarryho algoritmom nie je uzavretý. Potom by bol počet hrán použitých v smere do vrchola  $v$  o 1 väčší než počet hrán v smere z vrchola  $v$ . Posledný vrchol  $v$  by mal ešte jednu hranu nepoužitú v smere von z vrchola  $v$ , a teda algoritmus by mohol pokračovať v predlžovaní sledu  $\mu(s, v)$ . Sled  $\mu(s, v)$  je teda uzavretý.

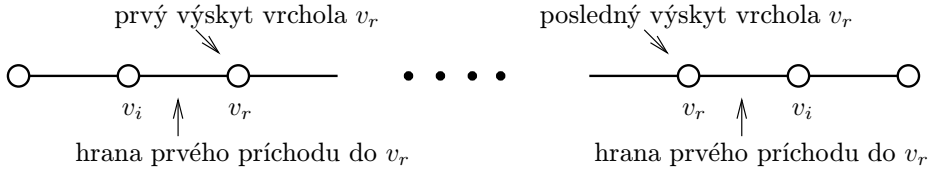
Nech  $(s = v_1, v_2, \dots, v_k)$  je poradie vrcholov, v akom sa v slede  $T$  vyskytujú po prvýkrát. Matematickou indukciou dokážeme, že sled  $T$  obsahuje všetky hrany incidentné s každým vrcholom postupnosti  $(s = v_1, v_2, \dots, v_k)$  použité v oboch smeroch.

Z vrchola  $s = v_1$  sme vyštartovali a doň sa nakoniec vrátili, a keďže už nemožno z neho ďalej pokračovať, musia byť všetky hrany jeho okolia použité v smere von. Počet hrán okolia  $v_1$  použitých v smere von sa rovná počtu hrán použitých v smere dnu a keďže žiadnu hranu nemožno použiť dvakrát v tom istom smere, môžeme uzavrieť, že každá hrana okolia  $v_1$  je použitá v oboch smeroch práve raz.

Nech pre každý vrchol  $v_i$  postupnosti  $(s = v_1, v_2, \dots, v_{r-1})$  sú použité v slede  $T$  všetky hrany okolia  $v_i$  v každom smere práve raz. Nech do vrchola  $v_r$  sme prišli prvýkrát hranou  $\{v_i, v_r\}$ ,  $i < r$ . Keďže podľa indukčného predpokladu sú v slede použité všetky hrany okolia  $v_i$  v každom smere práve raz, existuje v slede aj hrana  $\{v_r, v_i\}$  použitá v smere von z  $v_r$ . Táto hrana však bola hranou prvého príchodu do  $v_r$ , a teda mohla byť použitá až potom, čo boli použité všetky ostatné hrany okolia  $v_r$  v smere von, z čoho vyplýva, že všetky hrany smerom von z  $v_r$  boli použité. Počet príchodov do  $v_r$  je rovný počtu odchodov z  $v_r$ , žiadna hrana nebola použitá v jednom smere dvakrát a teda každá hrana okolia  $v_r$  bola použitá práve raz.

Nakoniec ukážeme, že Tarryho sled obsahuje všetky vrcholy. Nech  $v$  je vrchol, ktorý nie je v Tarryho slede  $T$ . Pretože graf  $G$  je súvislý, existuje  $s$ - $v$  cesta  $\mu(s, v)$ . Nech  $w$  je prvý vrchol na ceste  $\mu(s, v)$  ktorý nepatrí do  $T$ . Označme  $\mu(s, w)$  je úsek cesty  $\mu(s, v)$  začínajúci vrcholom  $s$  a končiaci vrcholom  $w$ . Cesta  $\mu(s, w)$  má tú vlastnosť, že jediný jej vrchol nepatriaci do sledu  $T$  je vrchol  $w$ . Nech  $u$  je predposledný vrchol tejto cesty. Vrchol  $u$  už patrí do sledu  $T$ , ale

hrana  $\{u, w\}$  už nie, čo je spor s tým, že Tarryho sled obsahuje s každým svojím vrcholom aj všetky hrany jeho okolia. ■



Obr. 3.4: Tarryho sled s prvým a posledným výskytom vrchola  $v_r$ .

Hrana  $\{v_i, v_r\}$  je hrana prvého príchodu do vrchola  $v_r$ , preto sled musí po poslednom výskyte vrchola  $v_r$  pokračovať hranou  $\{v_r, v_i\}$ .

**Príklad 3.1.** Urobte prieskum grafu  $G = (V, H)$ , ktorého diagram je na obrázku 3.5.

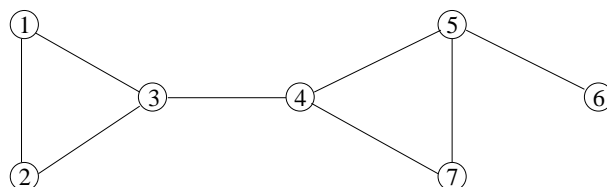
Príslušný graf  $G = (V, H)$  má množinu vrcholov  $V = \{1, 2, 3, 4, 5, 6, 7\}$  a množinu hrán  $H = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 7\}, \{5, 6\}, \{5, 7\}\}$ . Pre prehľadnosť algoritmus realizujeme v tabuľke, ktorá bude mať pre každú hranu a každý vrchol jeden stĺpec, v ktorom zaznamenávame smer použitia hrany, resp. navštívené vrcholy. Ak bola hrana použitá ako hrana prvého príchodu, zaznačíme smer jej použitia dvojitou šípkou, ináč značíme smer použitia hrany jednoduchou šípkou. Prvý stĺpec tabuľky obsahuje číslo riadku, do druhého stĺpca tabuľky postupne zapisujeme hrany zaraďované do Tarryho sledu.

Začneme z vrchola 3. Tejto skutočnosti zodpovedá 1. riadok tabuľky, v ktorom naznačíme, že vrchol 3 už bol preskúmaný.

Systematicky prehľadávame zoznam hrán, kým poprvýkrát nenarazíme na hranu incidentnú s vrcholom 3 zaraditeľnú do Tarryho sledu. Tou je hrana  $\{1, 3\}$ , ktorú zaradíme do sledu v smere  $(3, 1)$ . Tejto operácii zodpovedá druhý riadok tabuľky v ktorom druhý stĺpec obsahuje použitú hranu, v stĺpci hrany  $\{1, 3\}$  dvojitá šípka označuje, že hrana bola použitá ako hrana prvého príchodu do vrcholu 1 a nakoniec značka  $\bullet$  v stĺpci vrcholu 1 znamená, že sme práve preskúmali vrchol 1. Ďalej systematicky prehľadávame zoznam hrán a ako prvú zaraditeľnú hranu nájdeme hranu  $\{1, 2\}$ . Tretí riadok tabuľky zodpovedá zaradeniu hrany  $\{1, 2\}$  v smere  $(1, 2)$  a preskúmaniu vrchola 2 atď.

Ak by sme hrany pridávané do sledu zapisovali mimo tabuľky, potom by nám stačil len jeden jej riadok, do ktorého by sme zapisovali smer použitia

hrany a navštívené vrcholy. Pri realizácii algoritmu v počítači to tak skutočne budeme robiť. Pre ilustráciu výpočtového postupu je však tabuľka výhodnejšia, pretože v nej vidíme vývoj značenia hrán a vrcholov po jednotlivých krokoch.



Obr. 3.5: Diagram grafu k príkladu 3.1.

r		{1,2}	{1,3}	{2,3}	{3,4}	{4,5}	{4,7}	{5,6}	{5,7}	1	2	3	4	5	6	7
1																
2	{3,1}		←							•						
3	{1,2}	⇒									•					
4	{2,3}			→												
5	{3,2}			←												
6	{2,1}	←														
7	{1,3}		→													
8	{3,4}				⇒							•				
9	{4,5}					⇒							•			
10	{5,6}							⇒						•		
11	{6,5}								←							
12	{5,7}									⇒						•
13	{7,4}						←									
14	{4,7}						→									
15	{7,5}								←							
16	{5,4}					←										
17	{4,3}				←											

Hľadaný Tarryho sled je teda  $(3, \{3, 1\}, 1, \{1, 2\}, 2, \{2, 3\}, 3, \{3, 2\}, 2, \{2, 1\}, 1, \{1, 3\}, 3, \{3, 4\}, 4, \{4, 5\}, 5, \{5, 6\}, 6, \{6, 5\}, 5, \{5, 7\}, 7, \{7, 4\}, 4, \{4, 7\}, 7, \{7, 5\}, 5, \{5, 4\}, 4, \{4, 3\}, 3)$ .

Moderná algoritmická teória grafov je nemysliteľná bez odhadu zložitosti navrhnutých algoritmov. Aj my sa budeme snažiť stanoviť zložitosť našich algo-

ritmov. Ako je to z tohto hľadiska s Tarryho algoritmom, ak ho aplikujeme na graf  $G = (V, H)$  s  $n$  vrcholmi a  $m$  hranami? Pred zaradením každej hrany systematicky skúmame množinu hrán  $H$  z hľadiska zaraditeľnosti do sledu, pričom musíme vyskúšať najviac  $m$  hrán. Keďže Tarryho sled má  $2m$  hrán, môžeme zhora odhadnúť počet preskúmaní hrán na  $2m \times m = 2m^2$ . Keďže preskúmanie jednej hrany vyžaduje konštantný počet elementárnych operácií, môžeme uzavrieť, že počet všetkých operácií nutných pre vykonanie Tarryho algoritmu možno zhora ohraničiť číslom  $K.m^2$  a teda Tarryho algoritmus (v implementácii podľa príkladu 3.1) má zložitosť  $O(m^2)$ .

Tu by čitateľ mohol navrhnúť, že nie je nutné prehľadávať celú množinu hrán. Keď doterajší sled končí vrcholom  $u$ , stačí prehľadať len hrany okolia vrcholu  $u$  a tých je najviac  $n - 1$ . Takýmto zlepšením dostaneme algoritmus zložitosti  $O(m.n)$ . Keďže v mnohých grafoch býva  $m > n$ , je to isté zlepšenie pôvodného prístupu.

Ešte lepšiu implementáciou možno urobiť tak, že pre každý vrchol  $u \in V$  budeme udržiavať zoznam  $\mathcal{Z}(u)$  takých hrán jeho okolia, ktoré ešte neboli použité v smere von z vrchola  $u$ . Ďalej si pre každý vrchol  $u$  zapamätáme hranu prvého príchodu do  $u$ , t. j. hranu, ktorou sme sa poprvýkrát dostali do vrchola  $u$ . Ak doterajší Tarryho sled končí vo vrchole  $u$ , ďalej ho rozšírime o prvú hranu zoznamu  $\mathcal{Z}(u)$  rôznu od hrany prvého príchodu do vrchola  $u$  a zaradenú hranu vylúčime zo zoznamu  $\mathcal{Z}(u)$ , ak má  $\mathcal{Z}(u)$  viac ako jeden prvok. Až keď  $\mathcal{Z}(u)$  obsahuje jedinou hranu (je to hrana prvého príchodu do  $u$ ) použijeme na rozšírenie Tarryho sledu túto hranu.

V takejto implementácii Tarryho algoritmu už zaradenie ďalšej hrany do Tarryho sledu vyžaduje počet operácií, ktoré možno zhora ohraničiť konštantou. (Sú to kontrola, či  $\mathcal{Z}(u)$  má nula, jeden alebo viac prvkov, výber prvej hrany zo  $\mathcal{Z}(u)$ , kontrola, či to nie je hrana prvého príchodu, ak áno výber ďalšej hrany zo  $\mathcal{Z}(u)$ , zaradenie vybratej hrany do Tarryho sledu a vylúčenie zaradenej hrany zo zoznamu nezaradených hrán  $\mathcal{Z}(u)$  okolia vrchola  $u$ ). Preto možno počet operácií nutných na vykonanie Tarryho algoritmu zhora ohraničiť výrazom  $K.m$ , kde  $K$  je vhodná konštanta. Zložitosť takto implementovaného algoritmu sa rovná  $O(m)$ . Dostali sme implementáciu Tarryho algoritmu, ktorá je podstatne lepšia ako pôvodná so zložitou  $O(m^2)$ . Keďže do Tarryho sledu treba zaradiť  $2m$  hrán, neexistuje algoritmus na zostrojenie Tarryho sledu s menšou zložitou. Problém zostrojenia sledu, ktorý každú hranu súvislého grafu obsahuje práve dvakrát, má zložitosť  $O(m)$ .

Na tomto príklade vidíme, aký podstatný význam pre efektívnosť algoritmu majú údajové štruktúry použité pri jeho implementácii.

## 3.5 Najkratšia cesta

**Definícia 3.13.** Nech  $\mu(u, v)$  je  $u$ - $v$  sled (resp. orientovaný sled, resp. polosled) v hranovo ohodnotenom grafe  $G = (V, H, c)$  (resp. digrafe  $\vec{G} = (V, H, c)$ ). **Dĺžkou sledu (polosledu)  $\mu(u, v)$**  alebo tiež **cenou sledu (polosledu)** nazveme súčet ohodnotení jeho hrán, pričom ohodnotenie každej hrany započítavame toľkokrát, koľkokrát sa táto hrana v slede vyskytuje. Dĺžku sledu  $\mu(u, v)$  budeme značiť  $d(\mu(u, v))$ .

*Poznámka.* Podľa definície 3.1 (str. 59) sa pripúšťa aj triviálny sled s jediným vrcholom a žiadnou hranou. Dĺžka takéhoto sledu je nulová.

*Poznámka.* Predchádzajúcou definíciou je definovaná i dĺžka ľahu, orientovaného ľahu, cesty, polofahu a polocesty, pretože všetky tieto pojmy sú špeciálnym prípadom sledu, resp. polosledu.

Pre ľah, polofah, cestu, a polocestu, (v ktorých sa podľa ich definície každá hrana môže vyskytovať len raz), môžeme zjednodušene definovať: Dĺžka  $d(\mu(u, v))$  ľahu (polofahu, cesty alebo polocesty)  $\mu(u, v)$  je súčet ohodnotení ich hrán, t. j.

$$d(\mu(u, v)) = \sum_{h \in \mu(u, v)} c(h).$$

Pre sledy kvôli viacnásobnému výskytu hrán v postupnosti  $\mu(u, v)$  posledný zápis nemôžeme použiť. Ten totiž každej hrane  $h \in \mu(u, v)$  započítava jej ohodnotenie iba raz.

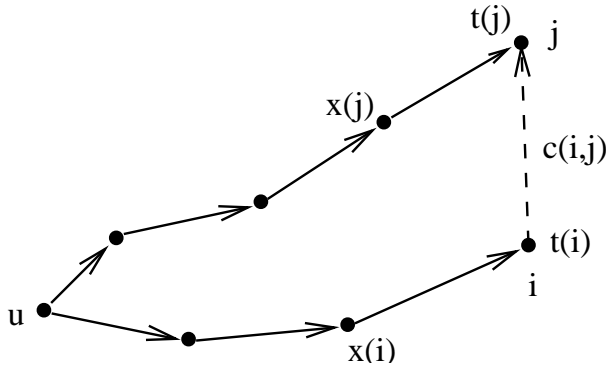
*Poznámka.* Často býva užitočné definovať dĺžku sledu  $\mu(u, v)$  v grafe  $G = (V, H)$ , resp. digrafe  $\vec{G} = (V, H)$ , ktorý nie je hranovo ohodnotený, ako počet hrán sledu  $\mu(u, v)$ . Takto definovaná dĺžka sledu je totožná s dĺžkou sledu v hranovo ohodnotenom grafe, (resp. digrafe)  $G' = (V, H, c)$ , kde  $c(h) = 1$  pre každú hrana  $h \in H$ .

*Poznámka.* Ak  $\mu(u, v) = \mu(u, w) \oplus \mu(w, v)$ , potom

$$d(\mu(u, v)) = d(\mu(u, w)) + d(\mu(w, v)).$$

**Definícia 3.14. Najkratšia  $u$ - $v$  cesta** v hranovo ohodnotenom grafe  $G = (V, H, c)$ , resp. hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$  je  $u$ - $v$  cesta, resp. orientovaná  $u$ - $v$  cesta  $\mu(u, v)$  s najmenšou dĺžkou.

Hľadanie najkratšej cesty v grafe alebo digrafe je frekventovaná úloha – má praktický zmysel sama osebe, ale často je hľadanie takejto cesty súčasťou iných algoritmov. Takmer všetky algoritmy na hľadanie najkratších  $u-i$  ciest pre pevné  $u$  pridelujú vrcholom po dve značky – značka  $t(i)$  predstavuje horný odhad dĺžky doteraz najlepšej nájdenej  $u-i$  cesty a  $x(i)$  je jej predposledný vrchol. Hodnotu značky  $t(i)$  môžeme považovať tiež za horný odhad dĺžky najkratšej  $u-i$  cesty.



Obr. 3.6: K princípu algoritmov na hľadanie najkratšej cesty.

V priebehu výpočtu získame nejakú  $u-i$  cestu  $\mu(u, i)$  s dĺžkou nanajvyš  $t(i)$  a predposledným vrcholom  $x(i)$ , a tiež nejakú  $u-j$  cestu  $\mu(u, j)$  dĺžky nanajvyš  $t(j)$  s predposledným vrcholom  $x(j)$ . Ak  $(i, j)$  je orientovaná hrana, t. j.  $(i, j) \in H$  a platí

$$t(j) > t(i) + c(i, j),$$

znamená to, že cesta  $\mu(u, i)$  predĺžená o hranu  $(i, j)$  do vrchola  $j$  s dĺžkou nanajvyš  $t(i) + c(i, j)$  má menšiu dĺžku než doterajší horný odhad  $t(j)$  dĺžky najkratšej  $u-j$  cesty. Keďže predposledný vrchol práve zlepšenej  $u-j$  cesty je vrchol  $i$ , položíme v tomto prípade  $t(j) := t(i) + c(i, j)$ ,  $x(j) := i$ . Tento princíp zlepšovania ciest je v tej či onej forme obsiahnutý v každom algoritme na hľadanie najkratšej cesty.

**Algoritmus 3.2. Základný algoritmus** na hľadanie najkratších orientovaných  $u-v$  ciest z pevného vrchola  $u \in V$  do všetkých dosiahnuteľných vrcholov  $v \in V$  v hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$  s nezápornou cenou hrany  $c(h)$ .



- **Krok 1. Inicializácia.**

Pre každý vrchol  $i \in V$  priradiť dve značky  $t(i)$  a  $x(i)$ .

{Značka  $t(i)$  predstavuje horný odhad dĺžky doteraz nájdenej najlepšej  $u$ - $i$  cesty a  $x(i)$  jej predposledný vrchol.}

Polož  $t(u) := 0$ ,  $t(i) := \infty$  pre  $i \in V$ ,  $i \neq u$  a  $x(i) := 0$  pre každé  $i \in V$ .

- **Krok 2.** Zisti, či existuje orientovaná hrana  $(i, j) \in H$ , pre ktorú platí

$$t(j) > t(i) + c(i, j). \quad (3.5)$$

Ak taká hrana  $(i, j) \in H$  existuje, potom polož

$$t(j) := t(i) + c(i, j), \quad x(j) := i$$

a opakuj Krok 2.

- **Krok 3.** Ak taká orientovaná hrana (z kroku 2.) neexistuje, najkratšiu  $u$ - $i$  cestu zostroj potom spätne pomocou značiek  $x(i)$  ako cestu prechádzajúcu vrcholmi

$$i, x(i), x(x(i)), x(x(x(i))), \dots, u$$

teda táto najkratšia cesta bude mať tvar

$$(u, \dots, x(x(x(i))), \underbrace{(x(x(x(i))), x(x(i)))}_{\text{hrana}}, x(x(i)), \underbrace{(x(x(i)), x(i))}_{\text{hrana}}, x(i), \underbrace{(x(i), i)}_{\text{hrana}}, i)$$

STOP.

{Po zastavení algoritmu konečná hodnota značky  $t(i)$  predstavuje dĺžku najkratšej  $u$ - $i$  cesty pre každý vrchol  $i$ . Ak  $t(i) = \infty$ , potom vrchol  $i$  nie je dosiahnuteľný z vrchola  $u$ .}



**Veta 3.5.** V digrafe  $\vec{G} = (V, H, c)$  s nezápornou cenou hrany sa základný algoritmus zastaví po konečnom počte krokov.

DŔKAZ.

Konečnosť základného algoritmu ukážeme pre jednoduchosť najprv pre celočíselné ceny hrán. Krok 2 základného algoritmu zníži značku jedného vrchola najmenej o hodnotu  $\Delta = 1$ . Keďže (vzhľadom na nezápornosť ceny hrany) značky

$t(\cdot)$  musia byť nezáporné, po konečnom počte Krokov 2 základný algoritmus musí skončiť.

Pre racionálne ceny hrán číslo položíme  $\Delta = \frac{1}{q}$ , kde  $q$  je spoločný menovateľ cien všetkých hrán.

Pre všeobecné reálne ceny hrán stanovíme  $\Delta$  ako minimum kladných hodnôt všetkých lineárnych kombinácií cien všetkých hrán s koeficientami z množiny  $\{-1, 0, 1\}$ . Hodnota, o ktorú sa zníži značka  $t(j)$  je rozdielom dĺžok pôvodnej a zlepšenej  $u$ - $j$  cesty a dá sa napísať ako lineárna kombinácia cien všetkých hrán s koeficientami z množiny  $\{-1, 0, 1\}$  a je zdola ohraničená práve definovanou hodnotou  $\Delta$ . ■

**Veta 3.6.** *Po skončení práce základného algoritmu na hľadanie najkratšej cesty pre každý vrchol  $v \in V$  platí: Vrchol  $v$  je dosiahnuteľný z vrchola  $u$  práve vtedy, keď  $t(v) < \infty$ . Ak je značka  $t(v)$  konečná, potom  $t(v)$  predstavuje dĺžku najkratšej orientovanej  $u$ - $v$  cesty a značka  $x(v)$  je predposledný vrchol tejto cesty.*

DŮKAZ.

Najprv ukážeme, že ak je značka  $t(i)$  konečná, potom  $t(x(i)) \leq t(i)$ .

Základný algoritmus nájde ako prvú hranu  $(i, j) \in H$ , pre ktorú platí  $t(j) > t(i) + c(i, j)$ , hranu typu  $(u, j)$ . Doteraz bolo  $t(j) = \infty$ , preto položíme  $t(j) := c(u, j)$ ,  $x(j) = u$  a platí  $t(x(j)) = t(u) = 0 \leq c(u, j) = t(j)$ .

Nech v priebehu základného algoritmu platí: Ak je značka  $t(i)$  konečná, potom  $t(x(i)) \leq t(i)$ . Nech pre orientovanú hranu  $(i, j) \in H$  platí 3.5. Nech  $k$  je taký vrchol, že  $x(k) = j$ , podľa predpokladu platí  $t(j) = t(x(k)) \leq t(k)$ . Táto nerovnosť po znížení hodnoty značky  $t(j)$  na hodnotu  $t(j) := t(i) + c(i, j)$  ostáva v platnosti. Pretože  $c(i, j) \geq 0$ ,  $t(i) \leq t(j)$  a po položení  $x(j) := i$  máme  $t(x(j)) \leq t(j)$ . Ak má vrchol  $i$  konečnú značku  $t(i)$  potom postupnosť

$$t(i), t(x(i)), t(x(x(i))), \dots, t(x^{(k)}(i)) \dots, t(u),$$

kde  $x^{(k)}(i)$  je definované takto  $x^{(1)}(i) = x(i)$ ,  $x^{(k)}(i) = x(x^{(k-1)}(i))$ , je nerastúcou postupnosťou čísel, t. j.

$$t(i) \geq t(x(i)) \geq t(x(x(i))) \geq \dots \geq t(x^{(k)}(i)) \geq \dots \geq t(u) = 0.$$

Ďalej ukážeme, že ak  $t(i) < \infty$ , potom postupnosť

$$i, x(i), x(x(i)), \dots, x^{(k)}(i) \dots u, \quad (3.6)$$

v žiadnom štádiu práce základného algoritmu neobsahuje žiaden vrchol viackrát.

Prvá hrana  $(i, j)$  v priebehu výpočtu algoritmom 3.2, pre ktorú  $t(j) > t(i) + c(i, j)$  je hrana typu  $(u, j)$ . Po vykonaní priradenia

$$t(j) := t(u) + c(i, j), \quad \text{a} \quad x(j) := u$$

postupnosť  $(j, x(j) = u)$  neobsahuje žiaden vrchol viackrát.

Nech pred ďalším krokom 2. základného algoritmu 3.2 postupnosť (3.6) neobsahuje žiaden vrchol viackrát. Nech  $(i, j)$  je taká hrana, že  $t(j) > t(i) + c(i, j)$ . Pretože  $c(i, j) \geq 0$ , musí byť  $t(j) > t(i) \geq t(x^k(i))$ , a preto  $j$  nemôže byť prvkom postupnosti (3.6). Po vykonaní kroku 2. bude  $t(j) = t(i) + c(i, j)$  a  $x(j) = i$ . Z dokázaného môžeme uzavrieť, že postupnosť

$$j, x(j) = i, x(x(j)) = x(i), \dots, x^{(k)}(j) = x^{(k-1)}(i) \dots, u \quad (3.7)$$

neobsahuje žiaden vrchol viac ako raz.

Keďže pre každý vrchol  $j$ ,  $j \neq u$  s konečnou značkou  $t(j)$  je definovaný predchodca  $x(j)$ , musí byť posledným vrcholom postupnosti (3.6) vrchol  $u$  – jediný vrchol s konečnou značkou, ktorý nemá definovaného predchodcu.

Doteraz sme dokázali: Ak je značka  $t(i)$  konečná, potom existuje také prirodzené číslo  $r$ , že  $x^{(r)}(i) = u$  a postupnosť vrcholov  $(x^{(r)}(i), x^{(r-1)}(i), \dots, x(i), i)$  definuje  $u$ - $i$  cestu, t. j. vrchol  $i$  je dosiahnuteľný z vrchola  $u$ .

Teraz ukážeme, že ak značka  $t(j)$  je konečná, potom  $t(j)$  je horným odhadom dĺžky najkratšej  $u$ - $j$  cesty. Základný algoritmus nájde ako prvú hranu  $(i, j) \in H$ , pre ktorú platí 3.5, hranu typu  $(u, j)$ . Doteraz bolo  $t(j) = \infty$ , preto položíme  $t(j) := c(u, j)$ , čo je dĺžka cesty  $(u, (u, j), j)$ , a teda  $t(j)$  je horným odhadom dĺžky najkratšej  $u$ - $j$  cesty.

Nech v priebehu základného algoritmu platí: Ak je značka  $t(i)$  konečná, potom  $t(i)$  je horným odhadom najkratšej  $u$ - $i$  cesty. Ak pre hranu  $(i, j) \in H$  platí 3.5, potom doteraz nájdená  $u$ - $i$  cesta predĺžená o hranu  $(i, j)$  je  $u$ - $j$  cestou a má horný odhad dĺžky  $t(i) + c(i, j)$ , čo je hodnota novej značky  $t(j)$ .

Teraz ukážeme, že ak je vrchol  $v$  dosiahnuteľný z vrchola  $u$ , potom  $t(v) < \infty$  a  $t(v)$  je dĺžka najkratšej  $u$ - $v$  cesty pre ľubovoľné  $v \in V$ . Nech

$$\mu(u, v) = (u = v_1, (v_1, v_2), v_2, \dots, v_{k-1}, (v_{k-1}, v_k), v_k = v)$$

je najkratšia  $u$ - $v$  cesta, nech  $t(v) > d(\mu(u, v))$ . Pre  $i = 1, 2, \dots, k$  označme

$$\mu(u, v_i) = (u = v_1, (v_1, v_2), v_2, \dots, v_{i-1}, (v_{i-1}, v_i), v_i)$$

Z definície dĺžky cesty vyplýva  $d(\boldsymbol{\mu}(u, v_1)) = d(\boldsymbol{\mu}(u, u)) = 0$  a pre  $i = 2, \dots, k$

$$d(\boldsymbol{\mu}(u, v_i)) = d(\boldsymbol{\mu}(u, v_{i-1})) + c(v_{i-1}, v_i). \quad (3.8)$$

Je  $t(v_1) = t(u) = 0 = d(\boldsymbol{\mu}(u, v_1))$ , a podľa predpokladu  $t(v_k) = t(v) > d(\boldsymbol{\mu}(u, v_k)) = d(\boldsymbol{\mu}(u, v))$ . Nech  $r$  je najmenší index, pre ktorý platí  $t(v_r) > d(\boldsymbol{\mu}(u, v_r))$ . Z dokázaného vyplýva, že  $1 < r \leq k$ .

Pretože  $t(v_{r-1}) = d(\boldsymbol{\mu}(u, v_{r-1}))$  a podľa 3.8

$$d(\boldsymbol{\mu}(u, v_r)) = d(\boldsymbol{\mu}(u, v_{r-1})) + c(v_{r-1}, v_r),$$

môžeme písať

$$t(v_r) > d(\boldsymbol{\mu}(u, v_r)) = d(\boldsymbol{\mu}(u, v_{r-1})) + c(v_{r-1}, v_r) = t(v_{r-1}) + c(v_{r-1}, v_r)$$

Za predpokladu, že existuje také  $v$ , že  $t(v)$  je väčšie než dĺžka najkratšej  $u-v$  cesty, sme našli hranu  $(v_{r-1}, v_r)$ , pre ktorú platí  $t(v_r) > t(v_{r-1}) + c(v_{r-1}, v_r)$ , čo je však v spore s podmienkou ukončenia práce základného algoritmu. ■

Pokúsme sa odhadnúť zložitosť základného algoritmu. Majme digraf  $\vec{G} = (V, H, c)$ , nech  $|V| = n$ ,  $|H| = m$ . Pre tento účel špecifikujme spôsob prehľadávania hranovej množiny  $H$  tak, že ju usporiadame do postupnosti  $\mathcal{H} = (h_1, h_2, \dots, h_m)$  a v tomto poradí skúšame, či  $t(j) > t(i) + c(i, j)$ , a ak áno, urobíme príslušné zmeny značiek  $t(j)$ ,  $x(j)$ . Ďalej budeme pokračovať ďalšou hranou v poradí. Ak dôjdeme na koniec postupnosti hrán, ďalej pokračujeme zase prvou hranou dovtedy, kým neskončatujeme, že sme po prezretí celej postupnosti ne našli ani jedno zlepšenie. Jedno prezretie celej postupnosti orientovaných hrán urobíme v čase  $O(m)$ . Ukážeme, že postupnosť hrán budeme prezerat' najviac  $n$ -krát.

Po prvom prezretí nájdeme všetky najkratšie orientované cesty, ktoré sú jednohranové. Značky ich koncových vrcholov sa už v ďalších kolách nebudú meniť (lebo sa už nedajú zmenšiť).

Predpokladajme, že po  $(k-1)$ -vom prezretí postupnosti  $\mathcal{H}$  sa už značky  $t(i)$  všetkých vrcholov  $i \in V$ , do ktorých existuje najkratšia  $u-i$  cesta s  $(k-1)$  hranami, rovnajú dĺžke najkratšej  $u-i$  cesty. Nech do  $j \in V$  existuje taká najkratšia cesta  $\boldsymbol{\mu}(u, j)$ , ktorá má práve  $k$  hrán. Nech predposledný vrchol cesty  $\boldsymbol{\mu}(u, j)$  je  $i$ , nech  $(i, j)$  je jej posledná hrana. Skrátením cesty  $\boldsymbol{\mu}(u, j)$  o posledný vrchol a poslednú hranu dostaneme cestu  $\boldsymbol{\mu}(u, i)$ , ktorá je najkratšou  $u-i$  cestou (inak by  $\boldsymbol{\mu}(u, j)$  nebola najkratšia  $u-j$  cesta). Cesta  $\boldsymbol{\mu}(u, i)$  má práve  $(k-1)$  hrán, a preto podľa predpokladu  $t(i) = d(\boldsymbol{\mu}(u, i))$ .

Potom  $d(\mu(u, j)) = d(\overline{\mu(u, i)}) + c(i, j) = t(i) + c(i, j)$ . Pri  $k$ -tom prezretí postupnosti  $\mathcal{H}$  skontrolujeme aj hranu  $(i, j)$ , a ak  $t(j) > t(i) + c(i, j)$ , položíme  $t(j) := t(i) + c(i, j)$ , čím sa značka  $t(j)$  stane rovnou dĺžke najkratšej  $u$ - $i$  cesty.

Pretože však v digrafe  $\vec{G}$  s  $n$  vrcholmi nemôže existovať cesta s viac ako  $n - 1$  hranami, znamená to, že sme po  $(n - 1)$  prezretiach našli všetky najkratšie orientované cesty. Ďalšie  $n$ -té prezeranie postupnosti hrán nám len potvrdí, že už niet čo zlepšovať. Zložitosť základného algoritmu je  $O(n.m)$ . Ak by sme ju chceli vyjadriť len pomocou  $n$ , máme s využitím  $m < n^2$  zložitosť  $O(n^3)$ .

Základný algoritmus je formulovaný pre digrafy, môžeme ho však ľahko prispôbiť pre použitie v hranovo ohodnotenom grafe  $G = (V, H, c)$ , tak, že ho budeme považovať za digraf s rovnakou množinou vrcholov  $V$ , ktorý bude mať ku každej hrane  $h \in H$  dvojicu opačne orientovaných hrán, obe s rovnakou cenou  $c(h)$ .

Základný algoritmus je použiteľný aj na také digrafy, ktoré majú ako cenu hrany všeobecné (teda aj záporné) reálne číslo. Nutnou a postačujúcou podmienkou použiteľnosti tohto algoritmu je, aby digraf  $\vec{G}$  neobsahoval cyklus zápornej dĺžky. Toto triviálne spĺňajú napríklad acyklické digrafy, ktoré neobsahujú žiaden cyklus. Pozor! V neorientovaných grafoch stačí na zacyklenie základného algoritmu jediná hrana so zápornou cenou.

**Príklad 3.2.** V digrafe  $\vec{G} = (V, H, c)$ , ktorý je daný diagramom na obrázku 3.7 zistíte všetky najkratšie cesty z vrchola 5 do ostatných vrcholov použitím základného algoritmu.

Digraf  $\vec{G}$  má množinu hrán s oceneniami podľa nasledujúcej tabuľky:

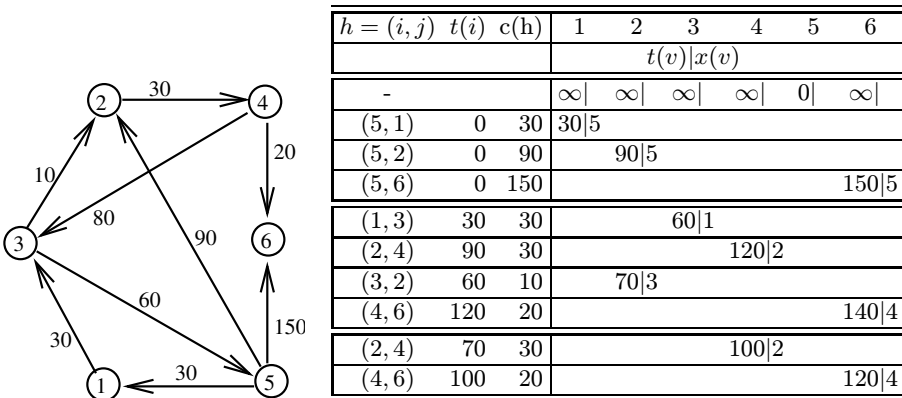
$h$	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 1)	(5, 2)	(5, 6)
$c(h)$	30	30	10	60	80	20	30	90	150

Orientované hrany pre kontrolu nerovnosti  $t(j) > t(i) + c(i, j)$  budeme postupne vyberať tak, ako sú usporiadané v tabuľke – teda v poradí (1, 3), (2, 4), (3, 2), atď., až kým sa nedostaneme k poslednej hrane (5, 6). Po spracovaní hrany (5, 6) začneme ďalšie kolo prehľadávania prvou hranou (1, 3). Ak v súčasnom kole došlo aspoň k jednému zlepšeniu značky  $t(v)$ , pokračujeme ďalším kolom.

Vývoj značiek  $t(v)$ ,  $x(v)$  budeme zapisovať do tabuľky, v ktorej venujeme každému zlepšeniu jeden riadok. V prvom stĺpci riadka zapíšeme hranu, ktorou sa dosiahlo zlepšenie a v stĺpci príslušného vrcholu zapíšeme zlepšenú značku

$t(v)|x(v)$ . Pre hrany, ktorými sa nedosiahlo zlepšenie, nebudeme vyčleňovať riadok. Ak začneme ďalšie kolo prehľadávania hrán, oddelíme ďalší riadok tabuľky dvojčiarou.

Značky tých vrcholov, ktoré sa nezmenili, nebudeme v novom riadku odpisovať. Pre každý vrchol platí posledná zapísaná hodnota značky v jeho stĺpci. Poznamenajme ešte, že v počítači pre realizáciu algoritmu netreba tabuľku – stačí uchovávať posledné platné hodnoty značiek. Nakoniec aj pri ručnom výpočte by sme vystačili s jedným riadkom tabuľky, ak by sme hodnoty značiek prepisovali. Zápis postupu riešenia do tabuľky má však tú výhodu, že vidíme postup výpočtu po jednotlivých krokoch algoritmu.



Obr. 3.7:  
Digraf  $\vec{G}$  a tabuľka s výpočtom najkratších ciest z vrchola 5 pomocou základného algoritmu.

Najkratšia cesta z vrchola 5 do vrchola 6 má dĺžku 120 a jej predposledný vrchol je 4, lebo vrchol 6 má značky  $t(6) = 120$  a  $x(6) = 4$ . Ďalej z tabuľky na obrázku 3.7 vidíme, že  $x(4) = 2$ ,  $x(2) = 3$ ,  $x(3) = 1$  a  $x(1) = 5$ , preto najkratšia 5–6 cesta bude

$$(5, (5, 1), 1, (1, 3), 3, (3, 2), 2, (2, 4), 4, (4, 6), 6).$$

Základný algoritmus je veľmi jednoduchý, čo do opisu. V kroku 2. je však ponechaná voľnosť výberu ďalšej hrany pre kontrolu vzťahu  $t(j) > t(i) + c(i, j)$ . Nevhodná implementácia hľadania vhodnej hrany  $(i, j)$  môže veľmi predĺžiť prácu

algoritmu. V ďalšom variante tohto algoritmu obmedzíme výber hrany tým, že v každom  $k$ -tom kole algoritmu budeme definovať množinu  $S_{k+1}$  začiatkov hrán  $(i, j)$ , ktorých konce  $j$  majú šancu na zmenšenie značky  $t(j)$  v nasledujúcom  $(k + 1)$ -vom kole. Množinu  $S_{k+1}$  tvoria práve tie vrcholy, ktorých značky sa v  $k$ -tom kole zmenili.

**Algoritmus 3.3. Fordov algoritmus** na hľadanie najkratších orientovaných  $u$ - $v$  ciest z pevného vrchola  $u \in V$  do všetkých ostatných vrcholov  $v \in V$  v hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$  s nezápornou cenou hrany  $c(h)$ .

- **Krok 1. Inicializácia.**

Pre každý vrchol  $i \in V$  priradiť dve značky  $t(i)$  a  $x(i)$ .

Polož  $t(u) = 0$ ,  $t(i) = \infty$  pre  $i \in V$ ,  $i \neq u$  a  $x(i) = 0$  pre každé  $i \in V$ .

Polož  $k := 1$ ,  $S_k := \{u\}$ ,  $n := |V|$ .

- **Krok 2.** Ak  $k = n$ , alebo ak  $S_k = \emptyset$ , STOP.

*{Pre každé  $i \in V$  platí: ak  $t(i) < \infty$ , potom značka  $t(i)$  predstavuje dĺžku najkratšej  $u$ - $i$  cesty, ktorá sa zostrojí spätne z  $i$  pomocou smerníkov  $x(i)$ .*

*Ak  $t(i) = \infty$ , vrchol  $i$  nie je dosiahnuteľný z vrchola  $u$ .}*

Inak pokračuj Krokom 3.

- **Krok 3.** Polož  $S_{k+1} := \emptyset$ .

- **Krok 4.** Pre všetky hrany typu  $(i, j)$ , kde  $i \in S_k$ , urob:

Ak  $t(j) > t(i) + c(i, j)$ , potom  $t(j) := t(i) + c(i, j)$ ,  $x(j) := i$ ,

$S_{k+1} := S_{k+1} \cup \{j\}$ .

- **Krok 5.** Polož  $k := k + 1$  a GOTO Krok 2.



Nech  $G = (V, H, c)$ , nech  $|V| = n$ ,  $|H| = m$ . Zložitosť Fordovho algoritmu odhadneme touto úvahou. Krok 1. sa dá uskutočniť  $O(n)$  operáciami. Krok 4. skontroluje najviac  $m$  hrán a opakuje sa najviac  $(n - 1)$ -krát. Celý algoritmus má teda zložitosť  $O(m \cdot n)$ . Ak chceme zložitosť vyjadriť len pomocou počtu vrcholov  $n$ , dostávame zložitosť  $O(n^3)$  (s využitím  $m \leq n(n - 1)$ ).

*Poznámka.*

1. Fordov algoritmus je plne použiteľný aj pre grafy, ak každú ich hranu považujeme za dvojicu proti sebe orientovaných hrán.
2. Fordov algoritmus dokáže pracovať i s takým hranovo ohodnoteným digrafom, v ktorom sú ohodnotenia hrán všeobecné – t. j. aj záporné čísla, ak v ňom neexistuje cyklus zápornej dĺžky.

**Algoritmus 3.4. Dijkstrov algoritmus** Algoritmus pre zostrojenie najkratšej orientovanej  $u-v$  cesty v hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$  s nezáporným ohodnotením hrán.

- **Krok 1.** Inicializácia. Pre každý vrchol  $i \in V$  priradi dve značky  $t(i)$  a  $x(i)$ . Značky  $t(i)$  budú dvojakého druhu, a to dočasné (ktoré sa ešte v priebehu výpočtu môžu zmeniť) a definitívne (ktoré sa už nemôžu zmeniť).

Polož  $t(u) = 0$ ,  $t(i) = \infty$  pre  $i \in V$ ,  $i \neq u$  a  $x(i) = 0$  pre každé  $i \in V$ . Zvoľ riadiaci vrchol  $r := u$  a značku  $t(\ )$  pri vrchole  $r = u$  prehlás za definitívnu, ostatné značky za dočasné.

- **Krok 2.** Ak je  $r = v$ , STOP. Ak  $t(v) < \infty$ , značka  $t(v)$  predstavuje dĺžku najkratšej  $u-v$  cesty, ktorú zostroj späťne z  $v$  pomocou smerníkov  $x(i)$ . Inak pre všetky hrany tvaru  $(r, j) \in H$ , kde  $j$  je vrchol s dočasnou značkou, urob:

Ak  $t(j) > t(r) + c(r, j)$ , potom  $t(j) := t(r) + c(r, j)$ ,  $x(j) := r$  a ponechaj zmenené značky ako dočasné.

- **Krok 3.** Zo všetkých dočasne označených vrcholov nájdí ten vrchol  $i$ , ktorý má značku  $t(i)$  minimálnu. Značku pri tomto vrchole  $i$  prehlás za definitívnu a zvoľ za nový riadiaci vrchol  $r := i$ .

*{ Pokiaľ existuje viac vrcholov s rovnakou minimálnou značkou, akú má vrchol  $i$ , za definitívnu značku môžeme prehlásiť len značku pri jednom z týchto vrcholov – ten potom berieme za riadiaci. Na tie ďalšie dôjde v nasledujúcich krokoch výpočtu. }*

GOTO Krok 2.



Zložitosť Dijkstrovho algoritmu: Vo všetkých krokoch 2. urobíme spolu maximálne  $m$  porovnaní, pretože pre žiadnu orientovanú hranu netreba robiť porovnanie  $t(j) > t(r) + c(r, j)$  viac než raz. Všetkých operácií, ktoré sa teda urobia



v rámci všetkých opakovaní kroku 2., je  $O(m)$ . V kroku 3., ktorý sa opakuje najviac  $n$ -krát, je treba nájsť minimum z maximálne  $n$  prvkov, čo sa dá urobiť  $O(n)$  operáciami. Zložitosť Dijkstrovho algoritmu je teda  $O(n^2)$ .

Dijkstrov algoritmus možno použiť aj pre hranovo ohodnotené grafy podobne, ako základný algoritmus alebo ako Fordov algoritmus.



Obr. 3.8: Digraf a graf so záporne ocenenými hranami, v ktorých Dijkstrov algoritmus zlyháva.

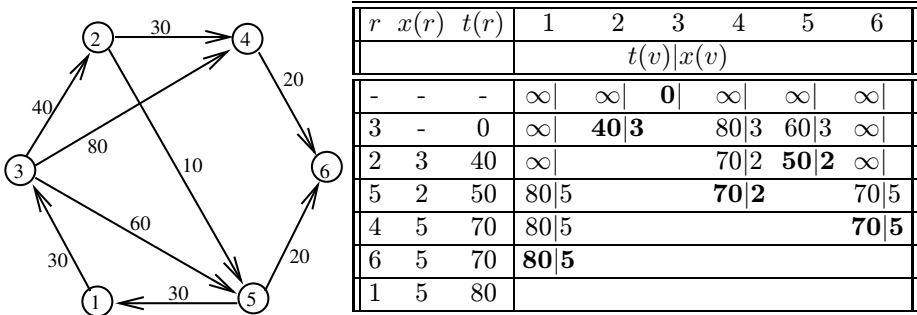
Pozor!! Dijkstrov algoritmus sa nehodí na digrafy so zápornou cenou hrany ani v prípade, že neobsahujú záporný cyklus. Jednoduché príklady zlyhania Dijkstrovho algoritmu pre digraf, resp. graf so všeobecne ohodnotenými hranami vidíme na obrázku 3.8.

*Poznámka.* Ak upravíme podmienku zastavenia v Kroku 2. na podmienku: Ak sú značky všetkých vrcholov definitívne, STOP, dostaneme verziu algoritmu, ktorá hľadá najkratšie cesty z vrchola  $u$  do všetkých dosiahnuteľných vrcholov.

**Príklad 3.3.** V digrafe  $\vec{G} = (V, H)$ , ktorý je daný diagramom na obrázku 3.9 zistíte všetky najkratšie cesty z vrchola 3 do ostatných vrcholov použitím Dijkstrovho algoritmu.

Prvý riadok tabuľky (po hlavičke oddelenej dvojčiarou) zodpovedá inicializácii. Všetkým vrcholom  $v \in V$  okrem vrcholu 3 je pridelená značka  $t(v) := \infty$ , vrcholu 3 značka  $t(3) := 0$ . Značky  $x(v)$  nemusia byť vôbec definované, určia sa v priebehu ďalšieho výpočtu. Značka vrcholu 3 je prehlásená za definitívnu (vytlačaná hrubo) a vrchol 3 je vybraný za riadiaci vrchol.

Ďalší riadok tabuľky zodpovedá značeniu z riadiaceho vrcholu 3. Pri tomto značení sa nám zmenili značky pri vrchoch 2, 4, 5. Hodnoty ostatných značiek opíšeme okrem definitívnych značiek, tie už na ďalší priebeh výpočtu nebudú mať žiaden vplyv.



Obr. 3.9: Tabuľka s výpočtom najkratších ciest z vrchola 3 pre digraf  $\vec{G}$ .

Minimálna dočasná značka  $t(v)$  v tomto riadku je značka  $t(2)$  vrcholu 2, preto značku  $t(2)$  prehlásime za definitívnu (vytlačená hrubo) a vrchol 2 bude ďalší riadiaci vrchol. Ďalší riadok budeme preto počítať s riadiacim vrcholom 2.

Za povšimnutie stojí riadok s riadiacim vrcholom 5. Po označení z riadiaceho vrchola 5 v riadku existujú dva vrcholy 4 a 6 s minimálnou značkou  $t(2) = t(6) = 70$ . V takomto prípade za definitívnu značku môžeme prehlásiť len jednu zo značiek  $t(2)$ ,  $t(6)$  (na druhú príde rad v ďalšom pokračovaní algoritmu). V našom prípade sme vybrali za nový riadiaci prvok vrchol 4, ale takisto by sme mohli zvoliť aj vrchol 6.

V riadkoch s riadiacimi vrcholmi 4 a 6 sa nám už žiadnu značku nepodarilo zlepšiť, čo však neznamená predčasný koniec algoritmu.

Ak by sme hľadali len jednu najkratšiu cestu – napríklad cestu z vrchola 3 do vrchola 5, skončíme vtedy, keď sa nám podarí definitívne označiť vrchol 5, tu konkrétne po vypočítaní riadku tabuľky s riadiacim vrcholom 2.

Po skončení algoritmu už ľahko nájdeme ľubovoľnú najkratšiu 3– $v$  cestu pomocou hodnôt  $x(v)$ . Tak napríklad  $x(1) = 5$ ,  $x(5) = 2$ ,  $x(2) = 3$ , a preto najkratšia 3–1 cesta je  $(3, (3, 2), 2, (2, 5), 5, (5, 1), 1)$  a jej dĺžka je  $t(1) = 80$ .

Poznamenajme ešte, že počítač môže všetky výpočty robiť v jednom riadku. To by sme mohli nakoniec robiť aj my prepisovaním čísel jedného riadku, avšak pre ručné počítanie je tabuľka výhodná, pretože v nej vidíme postup výpočtu po jednotlivých krokoch.

## 3.6 Výpočet matice vzdialeností

**Definícia 3.15.** Reálna funkcia  $d$  definovaná na kartézskom súčine  $V \times V$  sa nazýva **metrikou na množine  $V$** , ak platí:

1. Pre každé  $u, v \in V$  je  $d(u, v) \geq 0$  a rovnosť nastáva práve vtedy, keď  $u = v$ .
2. Pre každé  $u, v \in V$  platí  $d(u, v) = d(v, u)$ .
3. Pre každé  $u, v, w \in V$ , je  $d(u, w) \leq d(u, v) + d(v, w)$ .

**Definícia 3.16.** Nech  $G = (V, H, c)$  je súvislý hranovo ohodnotený graf alebo silne súvislý digraf,  $c(h) > 0$ . **Vzdialenosť vrcholov**  $u, v \in V$   $d(u, v)$  je dĺžka najkratšej  $u$ - $v$  cesty.

*Poznámka.* Keďže sme pripustili triviálnu  $u$ - $u$  cestu (obsahujúcu iba vrchol  $u$ ), ktorej dĺžka je nulová, z definície vzdialenosti vrcholov vyplýva, že pre každá  $u \in V$  platí  $d(u, u) = 0$ .

*Poznámka.* Ak graf  $G$  nie je súvislý, resp. digraf  $\vec{G}$  nie je silne súvislý, potom možno pre  $u \in V$ ,  $v \in V$  také, že  $v$  nie je dosiahnuteľný z  $u$  doplniť definíciu vzdialenosti tak, že položíme  $d(u, v) = \infty$ .

**Definícia 3.17.** Nech  $G = (V, H, c)$  je hranovo ohodnotený graf,  $c(h) > 0$ . Definujeme:

$$\begin{aligned} \text{excentricita vrchola } v \in V & \quad e(v) = \max\{d(u, v) \mid u \in V\} \\ \text{polomer – r\u00e1dius grafu } G & \quad r(G) = \min\{e(v) \mid v \in V\} \\ \text{priemer – diameter grafu } G & \quad d(G) = \max\{e(v) \mid v \in V\} \end{aligned}$$

Každý vrchol grafu  $G$  s minimálnou excentricitou  $e(v)$  nazveme **centr\u00e1lnym vrcholom** grafu  $G$ , množinu v\u00e9t\u00e7k\u00fdch centr\u00e1lnych vrcholov grafu nazveme **centrom** grafu  $G$ .

*Pozn\u00e1mka.* D\u00e1 sa \u013ahko uk\u00e1za\u0165, \u017ee pre priemer  $d(G)$  grafu  $G$  plat\u00ed

$$d(G) = \max\{d(u, v) \mid u \in V, v \in V\}.$$

**Veta 3.7.** Ak v s\u00falvislom grafe  $G = (V, H, c)$  je  $c(h) > 0$  pre ka\u017ed\u00fa hranu  $h \in H$ , potom je funkcia vzdialenosti  $d : V \times V \rightarrow \mathbb{R}$  metrikou na množine vrcholov  $V$ .

**D\u00d4KAZ.**

1. Ak  $u \neq v$ , potom najkrat\u00e1\u0161ia  $u$ - $v$  cesta obsahuje aspo\u0148 jednu hranu. Preto\u017ee

dĺžka hrany je kladné číslo, je pre  $u \neq v$   $d(u, v) > 0$ . Ak  $u = v$ , potom podľa definície  $d(u, v) = 0$ .

2. Nech  $\mu(u, v)$  je najkratšia  $u$ - $v$  cesta pre  $u \neq v$ . Ak napíšeme vrcholy a hrany postupnosti  $\mu(u, v)$  v opačnom poradí, dostaneme najkratšiu  $(v - u)$  – cestu s rovnakou dĺžkou. Je teda  $d(u, v) = d(v, u)$ .

3. Nech  $\mu(u, v)$  je najkratšia  $u$ - $v$  cesta,  $\mu(u, w)$  je najkratšia  $u$ - $w$  cesta,  $\mu(w, v)$  je najkratšia  $w$ - $v$  cesta. Zreťazenie  $\mu(u, w) \oplus \mu(w, v)$  ciest  $\mu(u, w)$ ,  $\mu(w, v)$  je  $u$ - $v$ -sled s dĺžkou  $d(\mu(u, w)) + d(\mu(w, v))$ , ktorá musí byť väčšia alebo rovná dĺžke najkratšej  $u$ - $v$  cesty, čo sa dá zapísať ako:

$$d(u, v) \leq d(u, w) + d(w, v).$$

*Poznámka.* V digrafoch nemusí platiť  $d(u, v) = d(v, u)$ , preto tu funkcia  $d$  vo všeobecnom prípade nie je metrikou. Toto budeme mať na mysli, keď budeme používať pre číslo  $d(u, v)$  termín „vzdialenosť vrcholov  $u, v$ “.

**Algoritmus 3.5. Floydov algoritmus** na výpočet matice vzdialeností vrcholov v hranovo ohodnotenom grafe, resp. digrafe  $G = (V, H, c)$ , kde  $c(h) \geq 0$ .

- **Krok 1.** Zostroj maticu  $\mathbf{C} = (c_{ij})$ , ktorej prvky sú definované nasledovne:

$$c_{ii} = 0 \quad \text{pre všetky } i \in V$$

a pre všetky  $i, j$ , také, že  $i \neq j$

$$c_{ij} = \begin{cases} c(i, j), & \text{ak } \{i, j\} \in H, \text{ resp. } (i, j) \in H \\ \infty, & \text{ak } \{i, j\} \notin H, \text{ resp. } (i, j) \notin H \end{cases}$$

Zostroj aj maticu  $\mathbf{X} = (x_{ij})$ , kde

$$x_{ii} = i \quad \text{pre všetky } i \in V$$

a pre všetky  $i, j$ , také, že  $i \neq j$

$$x_{ij} = \begin{cases} i, & \text{ak } \{i, j\} \in H, \text{ resp. } (i, j) \in H \\ \infty, & \text{ak } \{i, j\} \notin H, \text{ resp. } (i, j) \notin H \end{cases}$$

- **Krok 2.** Urob pre všetky  $k = 1, 2, \dots, n = |V|$ :  
Pre všetky  $i \neq k$  také, že  $c_{ik} \neq \infty$ , a pre všetky  $j \neq k$  také, že  $c_{kj} \neq \infty$ ,

urob:

Ak  $c_{ij} > c_{ik} + c_{kj}$ , potom polož:

$$c_{ij} := c_{ik} + c_{kj}$$

$$x_{ij} := x_{kj}$$



Po skončení Floydovho algoritmu je matica  $\mathbf{C}$  maticou vzdialeností vrcholov grafu, resp. digrafu  $G$  a matica  $\mathbf{X}$  obsahuje pre každú dvojicu vrcholov  $i, j$  takú, že  $j$  je dosiahnuteľný z  $i$ , predposledný vrchol najkratšej  $i$ - $j$  cesty (toto dokážeme v nasledujúcej vete). Ak potrebujeme nájsť najkratšiu  $i$ - $j$  cestu, využijeme maticu smerníkov  $\mathbf{X}$  nasledovne: Pre predposledný vrchol  $j_1$  najkratšej  $i$ - $j$  cesty je  $j_1 = x_{ij}$ . Ďalší vrchol tejto cesty (odzadu) je  $j_2 = x_{ij_1}$ , ďalší  $j_3 = x_{ij_2}$  atď., pokiaľ nedôjdeme do vrchola  $i$ .

Zložitosť Floydovho algoritmu. Nech  $n$  je počet vrcholov grafu  $G$ . Vytvorenie matic  $\mathbf{C}$  a  $\mathbf{X}$  v Kroku 1. má zložitosť  $O(n^2)$ . V Kroku 2. sa  $n$ -krát skontrolujú a prípadne aj zmenia skoro všetky prvky matice  $\mathbf{C}$  a matice  $\mathbf{X}$ , ktoré obe majú po  $n^2$  prvkov, z čoho vyplýva, že zložitosť Kroku 3. a teda aj celého algoritmu je  $O(n^3)$ .

**Veta 3.8.** *Nech  $G = (V, H, c)$  je hranovo ohodnotený graf alebo digraf, kde  $c(h) \geq 0$  pre každú hranu  $h \in H$ . Po skončení Floydovho algoritmu je matica  $\mathbf{C}$  maticou vzdialeností vrcholov grafu, resp. digrafu  $G$  a matica  $\mathbf{X}$  obsahuje pre každú dvojicu vrcholov  $i, j$  predposledný vrchol najkratšej  $i$ - $j$  cesty, pokiaľ je vrchol  $j$  dosiahnuteľný z vrchola  $i$ .*

DÔKAZ.

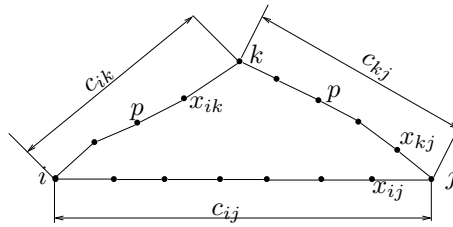
Pred iteráciou Kroku 2 pre  $k = 1$  matica  $\mathbf{C}$  obsahuje dĺžky všetkých jednohranových  $i$ - $j$  ciest pokiaľ také existujú, alebo  $\infty$ , ak vrcholy  $i, j$  nie sú susedné. Matica  $\mathbf{X}$  obsahuje v prvku  $x_{ij}$  predposledný (a v tomto prípade aj prvý) vrchol najkratšej jednohranovej  $i$ - $j$  cesty alebo  $\infty$ , ak takáto cesta neexistuje.

Po iterácii Kroku 2 pre  $k = 1$  matica  $\mathbf{C}$  obsahuje v prvku  $c_{ij}$  minimum dĺžok jednohranových ciest typu  $(i, \{i, j\}, j)$  a ciest typu  $(i, \{i, 1\}, 1, \{1, j\}, j)$  a matica  $\mathbf{X}$  obsahuje v prvku  $x_{ij}$  predposledný vrchol kratšej z týchto dvoch ciest.

Ukážeme, že matica  $\mathbf{C}$  obsahuje po iterácii Kroku 2 pre  $k$  v prvku  $c_{ij}$  dĺžku najkratšej  $i$ - $j$  cesty spomedzi cesty  $(i, \{i, j\}, j)$  (pokiaľ vôbec taká existuje) a všetkých  $i$ - $j$  ciest, ktoré prechádzajú (okrem vrcholov  $i, j$ ) iba (niektorými) vrcholmi z množiny  $\{1, 2, \dots, k\}$ . Matica  $\mathbf{X}$  obsahuje v prvku  $x_{ij}$  predposledný

vrchol tejto cesty. Označme toto tvrdenie  $T(k)$ . Budeme postupovať matematickou indukciou.

$T(1)$  sme už dokázali. Nech platí  $T(k-1)$ . Podľa indukčného predpokladu tesne po iterácii Kroku 2 pre  $k-1$  (a pred iteráciou pre  $k$ ) boli  $c_{ij}$ ,  $c_{ik}$ ,  $c_{kj}$  po rade dĺžkami najkratšej  $i$ - $j$  cesty  $\mu(i, j)$ ,  $i$ - $k$  cesty  $\mu(i, k)$ ,  $k$ - $j$  cesty  $\mu(k, j)$ , v ktorých sa okrem začiatočného a koncového vrchola nesmeli vyskytnúť iné vrcholy ako  $1, 2, \dots, k-1$ .



Obr. 3.10: K dôkazu správnosti Floydovho algoritmu.

Najkratší  $i$ - $j$  sled obsahujúci vrchol  $k$  a okrem neho a vrcholov  $i$ ,  $j$  iba vrcholy z množiny  $\{1, 2, \dots, k-1\}$  má dĺžku  $c_{ik} + c_{kj}$  (z  $i$  do  $k$  sa nevieme lepšie dostať ako po ceste  $\mu(i, k)$  a z  $k$  do  $j$  je najkratšia cesta  $\mu(k, j)$ ). Ak  $c_{ij} \leq c_{ik} + c_{kj}$ , potom najkratšia cesta  $\mu(i, j)$  idúca len cez niektoré z vrcholov  $1, 2, \dots, k-1$  je kratšia ako najkratší sled, ktorý ide cez vrchol  $k$  a okrem neho iba cez niektoré z vrcholov  $1, 2, \dots, k-1$ . V tomto prípade sa po  $k$ -tom kroku  $c_{ij}$  ani  $x_{ij}$  nezmenia.

Ak  $c_{ij} > c_{ik} + c_{kj}$ , potom najkratší  $i$ - $j$  sled idúci cez vrchol  $k$  a okrem neho len niektorými z vrcholov  $1, 2, \dots, k-1$  je kratší, ako cesta  $\mu(i, j)$ . Ukážeme, že v tomto prípade zreťazenie ciest  $\mu(i, k) \oplus \mu(k, j)$  je cestou. Predpokladajme, že  $\mu(i, k) \oplus \mu(k, j)$  nie je cestou. Potom obsahuje aspoň jeden vrchol  $p \neq k$  dvakrát. Vrchol  $p$  sa môže vyskytovať iba raz na ceste  $\mu(i, k)$  a iba raz na ceste  $\mu(k, j)$ . Vylúčením cyklu začínajúceho a končiaceho vrcholom  $p$  zo sledu  $\mu(i, k) \oplus \mu(k, j)$  dostaneme  $i$ - $j$  sled  $m'(i, j)$  neobsahujúci vrchol  $k$  (a teda prechádzajúci nanajvyš vrcholmi  $1, 2, \dots, k-1$ ) s dĺžkou menšou ako  $c_{ij}$ , čo je v spore s indukčným predpokladom. Zreťazenie  $\mu(i, k) \oplus \mu(k, j)$  je v tomto prípade najkratšou  $i$ - $j$  cestou prechádzajúcou nanajvyš cez niektoré z vrcholov  $1, 2, \dots, k$ . Pretože  $x_{ij}$  sa po tejto iterácii zmenilo na  $x_{ij} = x_{kj}$ , čo bol podľa indukčného predpokladu predposledný vrchol cesty  $\mu(k, j)$  a teda je predposledným vrcholom zreťazenia  $\mu(i, k) \oplus \mu(k, j)$ . ■

Floydov algoritmus je aplikovateľný i na digrafy so všeobecným (teda i záporným) ohodnotením hrán za predpokladu, že v ňom neexistuje cyklus zápornej ceny.

*Poznámka.* Floydov algoritmus je asi jedným z najľahšie naprogramovateľných algoritmov. Treba si však dať pozor na počítačovú reprezentáciu  $\infty$ . Niektorí by mohli navrhnúť použiť namiesto neho najväčšie zobraziteľné celé číslo (pri 16-bitovom type integer 32767). Nie je to však najlepšia voľba, pretože pri prípadnom sčítaní takéhoto nekonečna s iným celým číslom dostaneme buď záporný výsledok, alebo chybu z pretečenia. Lepšie je zvoliť za  $\infty$  také veľké číslo, ktorého dvojnásobok ešte nepresiahne najväčšie zobraziteľné číslo. Potom možno kontroly na nekonečno z Floydovho algoritmu vynechať a výsledný pascalovský program by mohol byť nasledovný:

```

for  $k := 1$  to  $n$  do
  for  $i := 1$  to  $n$  do
    for  $j := 1$  to  $n$  do
      if  $c[i, j] > c[i, k] + c[k, j]$  then
        begin
           $c[i, j] := c[i, k] + c[k, j];$ 
           $x[i, j] := x[k, j];$ 
        end

```

Na záver tejto časti uvediem ešte dve implementácie algoritmu na hľadanie najkratšej cesty v digrafe.

**Algoritmus 3.6. Label-set a Label-correct implementácia algoritmu** na hľadanie najkratších orientovaných  $u$ - $v$  ciest z pevného vrchola  $u \in V$  do všetkých ostatných vrcholov  $v \in V$  v hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$  s nezápornou cenou orientovanej hrany  $c(h)$ .

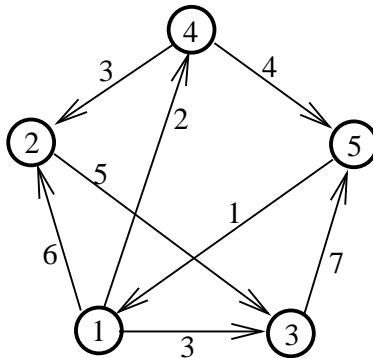
- **Krok 1: Inicializácia.**  
 Polož  $t(u) := 0$ ,  $t(i) := \infty$  pre  $i \in V$ ,  $i \neq u$  a  $x(i) := 0$  pre každé  $i \in V$ .  
 Polož  $\mathcal{E} := \{u\}$ .
- **Krok 2:** Vyber  $r \in \mathcal{E}$ , polož  $\mathcal{E} := \mathcal{E} - \{r\}$ .  
 Pre všetky orientované hrany tvaru  $(r, j) \in H$  urob:  
 Ak  $t(j) > t(r) + c(r, j)$ , potom  $t(j) := t(r) + c(r, j)$ ,  $x(j) := r$ ,  $\mathcal{E} := \mathcal{E} \cup \{j\}$ .

- **Krok 3:** Ak  $\mathcal{E} \neq \emptyset$ , choď na Krok 2.  
Ak  $\mathcal{E} = \emptyset$ , potom  $t(i)$  predstavuje dĺžku najkratšej orientovanej  $u-i$  cesty pre každý vrchol  $i$ . Najkratšiu orientovanú  $u-i$  cestu zostroj potom spätne pomocou značiek  $x(i)$  ako v predchádzajúcich dvoch algoritmoch.



Ak v druhom kroku algoritmu 3.6 vyberáme  $r \in \mathcal{E}$  ľubovoľne, dostávame implementáciu základného algoritmu, ktorú voláme **label correct algoritmus**. Ak za prvok  $r \in \mathcal{E}$  vyberáme prvok z najmenšou značkou  $t()$ , potom dostaneme implementáciu Dijkstrovho algoritmu, ktorú voláme **label set algoritmus**. Pre label correct algoritmus je výhodné organizovať  $\mathcal{E}$  ako zásobník, pre label set algoritmus sa  $\mathcal{E}$  organizuje ako prioritný front, prípadne ako halda. Aby sme do zásobníka, resp. do prioritného frontu  $\mathcal{E}$  nekladali ten istý vrchol viackrát, je vhodné ku každému vrcholu  $v \in V$  udržiavať indikátor hovoriaci, či vrchol  $v$  je v množine  $\mathcal{E}$ .

**Príklad 3.4.** Je daný digraf  $\vec{G} = (V, H, c)$  diagramom z obrázku 3.11. Vypočítame maticu vzdialeností pomocou Floydovho algoritmu.



Obr. 3.11: Digraf k príkladu 3.4.

Maticy  $\mathbf{C}$  a  $\mathbf{X}$  sú uvedené v dvojiciach pod sebou začínajúc inicializačnými maticami. Prvky  $k$ -teho riadku a stĺpca oboch matíc sú vyznačené tučným fontom, prvky  $c_{ij}$  matice  $\mathbf{C}$ , pre ktoré je  $c_{ij} > c_{ik} + c_{kj}$  sú v rámečkoch. Príslušné prvky  $x_{ij}$  v matici  $\mathbf{X}$  sú tiež zarámované – tie sa totiž v nasledujúcej matici zmenia, ostatné prvky ostanú bez zmeny.



Matica **C**

	1	2	3	4	5
1	0	6	3	2	$\infty$
2	$\infty$	0	5	$\infty$	$\infty$
3	$\infty$	$\infty$	0	$\infty$	7
4	$\infty$	3	$\infty$	0	4
5	1	$\infty$	$\infty$	$\infty$	0

Matica **C**  
po kroku 2 s  $k = 1$ 

	1	2	3	4	5
1	0	6	3	2	$\infty$
2	$\infty$	0	5	$\infty$	$\infty$
3	$\infty$	$\infty$	0	$\infty$	7
4	$\infty$	3	$\infty$	0	4
5	1	7	4	3	0

Matica **C**  
po kroku 2 s  $k = 2$ 

	1	2	3	4	5
1	0	6	3	2	$\infty$
2	$\infty$	0	5	$\infty$	$\infty$
3	$\infty$	$\infty$	0	$\infty$	7
4	$\infty$	3	8	0	4
5	1	7	4	3	0

Matica **X**

	1	2	3	4	5
1	1	1	1	1	$\infty$
2	$\infty$	2	2	$\infty$	$\infty$
3	$\infty$	$\infty$	3	$\infty$	3
4	$\infty$	4	$\infty$	4	4
5	5	$\infty$	$\infty$	$\infty$	5

Matica **X**  
po kroku 2 s  $k = 1$ 

	1	2	3	4	5
1	1	1	1	1	$\infty$
2	$\infty$	2	2	$\infty$	$\infty$
3	$\infty$	$\infty$	3	$\infty$	3
4	$\infty$	4	$\infty$	4	4
5	5	1	1	1	5

Matica **X**  
po kroku 2 s  $k = 2$ 

	1	2	3	4	5
1	1	1	1	1	$\infty$
2	$\infty$	2	2	$\infty$	$\infty$
3	$\infty$	$\infty$	3	$\infty$	3
4	$\infty$	4	2	4	4
5	5	1	1	1	5

Matica **C**  
po kroku 2 s  $k = 3$ 

	1	2	3	4	5
1	0	6	3	2	10
2	$\infty$	0	5	$\infty$	12
3	$\infty$	$\infty$	0	$\infty$	7
4	$\infty$	3	8	0	4
5	1	7	4	3	0

Matica **C**  
po kroku 2 s  $k = 4$ 

	1	2	3	4	5
1	0	5	3	2	6
2	$\infty$	0	5	$\infty$	12
3	$\infty$	$\infty$	0	$\infty$	7
4	$\infty$	3	8	0	4
5	1	6	4	3	0

Matica **C**  
po kroku 2 s  $k = 5$ 

	1	2	3	4	5
1	0	5	3	2	6
2	13	0	5	15	12
3	6	13	0	10	7
4	5	3	8	0	4
5	1	6	4	3	0

Matica **X**  
po kroku 2 s  $k = 3$ 

	1	2	3	4	5
1	1	1	1	1	3
2	$\infty$	2	2	$\infty$	3
3	$\infty$	$\infty$	3	$\infty$	3
4	$\infty$	4	2	4	4
5	5	1	1	1	5

Matica **X**  
po kroku 2 s  $k = 4$ 

	1	2	3	4	5
1	1	4	1	1	4
2	$\infty$	2	2	$\infty$	3
3	$\infty$	$\infty$	3	$\infty$	3
4	$\infty$	4	4	4	4
5	5	4	1	1	5

Matica **X**  
po kroku 2 s  $k = 5$ 

	1	2	3	4	5
1	1	4	1	1	4
2	5	2	2	1	3
3	5	4	3	1	3
4	5	4	2	4	4
5	5	4	1	1	5

### 3.7 Hľadanie cyklov zápornej ceny v digrafe

V predchádzajúcej časti sme uviedli niekoľko algoritmov na hľadanie najkratšej  $u-v$  cesty v digrafoch a grafoch s nezápornou cenou hrán. Pokiaľ sme sa zaoberali ich zložitou, sme ukázali, že všetky tieto algoritmy majú polynomiálnu zložitnosť. Ak budeme skúmať použiteľnosť týchto algoritmov na grafy a digrafy so všeobecne ohodnotenými hranami (t. j. aj hranami so zápornou cenou) zistíme, že pre každý z týchto algoritmov sa dá nájsť jednoduchý príklad grafu alebo digrafu, na ktorom tento algoritmus zlyhá.

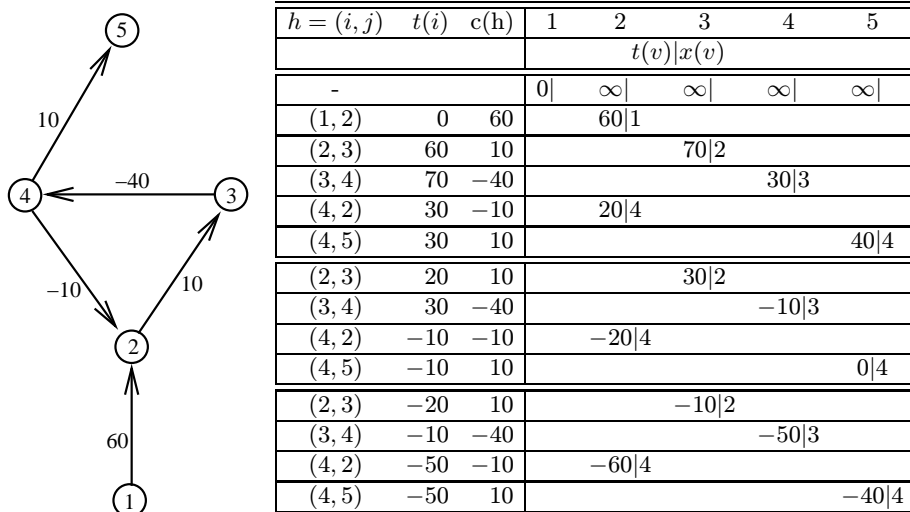
Ukazuje sa, že úloha hľadania najkratšej  $u-v$  cesty vo všeobecne ohodnotenom digrafe alebo grafe je NP-ťažká. Podstata zložitosti tejto úlohy spočíva v existencii cyklov so zápornou cenou. V acyklických digrafoch so všeobecnou cenou je úloha hľadania najkratšej cesty polynomiálnou aj v prípade záporných ohodnotení hrán a môžeme ju riešiť základným algoritmom 3.2, Fordovým algoritmom 3.3 alebo Floydovým algoritmom 3.5. Dijkstrov algoritmus 3.4 pri záporných ohodnoteniach hrán zlyháva.

Ak je už úloha hľadania najkratšej cesty vo všeobecne ohodnotenom digrafe ťažká, vzniká otázka, či je možné polynomiálnym algoritmom aspoň identifikovať cyklus zápornej ceny. Hľadanie cyklov zápornej ceny v digrafoch má veľký význam pri riešení mnohých iných optimalizačných problémov teórie grafov. Tak napríklad hľadanie cyklu zápornej ceny v digrafe je súčasťou algoritmu 7.4 na hľadanie toku v sieti s minimálnou cenou.

**Príklad 3.5.** Príklad aplikácie základného algoritmu 3.2 (str. 72) na digraf so záporným cyklom. Digraf  $G$  a príslušnú tabuľku s výpočtom vidíme na obrázku 3.12.

Základný algoritmus sa tu nikdy nezastaví, pretože hranami  $(2, 3)$ ,  $(3, 4)$ ,  $(4, 2)$  možno donekonečna znižovať ohodnotenia  $t(2)$ ,  $t(3)$ ,  $t(4)$ . Všimnime si, aké sú značky  $x()$ :  $x(4) = 3$ ,  $x(3) = 2$  a  $x(2) = 4$ , čo znamená, že do vrchola 4 sme prišli z vrchola 3 do vrchola 3 z vrchola 2 a do vrchola 2 z vrchola 4. Značky  $x()$  nás teraz nedovedia do začiatočného vrchola 1, ale definujú cyklus, ktorého súčet ohodnotení hrán je záporný.

Ak chceme modifikovať základný algoritmus 3.2 alebo Fordov algoritmus 3.3 na to, aby identifikoval existenciu cyklu zápornej ceny, musíme po každej zmene značiek  $t(j)$ ,  $x(j)$  skontrolovať, či značky  $x()$  definujú  $u-j$  cestu, alebo či zmenou značky  $x(j)$  vznikol cyklus obsahujúci vrchol  $j$ . To zistíme tak, že preskúmame



Obr. 3.12: Digraf so záporným cyklom a tabuľka s výpočtom podľa základného algoritmu.

postupnosť vrcholov

$$x(j), x(x(j)), x(x(x(j))), \dots,$$

ktorá môže mať najviac  $n - 1$  členov. Ak táto postupnosť obsahuje vrchol  $j$ , našli sme cyklus zápornej ceny obsahujúci vrchol  $j$ .

V našom príklade by sme identifikovali vznik cyklu zápornej ceny už v piatom riadku tabuľky po použití hrany  $(4, 2)$  po preznačovaní vrcholu 2.

Na nájdenie cyklu zápornej ceny v digrafe však nestačí nasadiť modifikovaný základný alebo Fordov algoritmus s jedným začiatočným vrcholom  $u$ . Ak by sme v našom príklade 3.5 štartovali z vrchola 5, neobjavili by sme žiaden cyklus zápornej ceny. Nieкто by mohol navrhnúť spustiť modifikovaný algoritmus z každého vrchola, čo by však nebolo veľmi ekonomické. Stačí však vykonať algoritmus s takou podmnožinou štartovacích vrcholov, že každý vrchol digrafu je dosiahnuteľný z niektorého štartovacieho vrchola.

Inou možnosťou je hľadať záporný cyklus v pomocnom digrafe  $\vec{G}'$ , ktorý vznikne z pôvodného digrafu  $\vec{G}$  pridaním fiktívneho vrcholu  $z \notin V$ , a všetkých

orientovaných hrán typu  $(z, v)$   $v \in V$  s nulovou cenou. Ako štartovací vrchol sa použije fiktívny vrchol  $z$ .

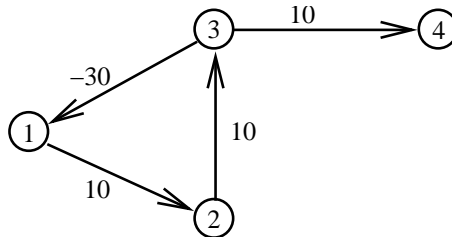
Floydov algoritmus 3.5 (str. 84) možno tiež modifikovať tak, že v prípade všeobecného digrafu nájde záporný cyklus. Stačí v kroku 1. definovať začiatočnú maticu  $\mathbf{C}$  nasledovne

$$c_{ij} = \begin{cases} c(i, j), & \text{ak } \{i, j\} \in H, \text{ resp. } (i, j) \in H \\ \infty, & \text{ak } \{i, j\} \notin H, \text{ resp. } (i, j) \notin H \end{cases}$$

Matica  $\mathbf{C}$  má na rozdiel od štandardného Floydovho algoritmu na hlavnej diagonále  $\infty$ . Matica  $\mathbf{X}$  je bez zmeny. Krok 2. je rovnaký. Treba si pri ňom však dať pozor na to, že pri štandardnom Floydovom algoritme sa prvky diagonály nemenili, pri tejto verzii sa však meniť budú. Po ukončení práce tohto algoritmu budú prvky  $c_{ii}$  na diagonále rovné dĺžke najkratšieho  $i$ - $i$  cyklu.

Ak sa na hlavnej diagonále matice  $\mathbf{C}$  v priebehu výpočtu Floydovým algoritmom objaví záporné číslo  $c_{jj}$ , objavili sme tým cyklus zápornej ceny obsahujúci vrchol  $j$ . Tento cyklus určíme pomocou matice smerníkov  $\mathbf{X}$ .

**Príklad 3.6.** Treba zistiť, či digraf na obrázku 3.13 obsahuje cyklus zápornej ceny. Zostavíme matice  $\mathbf{C}$  a  $\mathbf{X}$  a postupne urobíme Krok 2. Floydovho algoritmu 3.5 (str. 84) pre  $k = 1, 2, 3$ . Vývoj matíc  $\mathbf{C}$  a  $\mathbf{X}$  vidíme postupne v tabuľkách za obrázkom 3.13. Keďže v poslednom riadku matice  $\mathbf{C}$  pre  $k = 3$  sú samé  $\infty$ , pre  $k = 4$  sa už matice  $\mathbf{C}$  a  $\mathbf{X}$  nezmenia.



Obr. 3.13: Digraf k príkladu 3.6.

<b>C</b>	<b>C po <math>k = 1</math></b>	<b>C po <math>k = 2</math></b>	<b>C po <math>k = 3</math></b>
$\begin{matrix} \infty & 10 & \infty & \infty \\ \infty & \infty & 10 & \infty \\ -30 & \infty & \infty & 10 \\ \infty & \infty & \infty & \infty \end{matrix}$	$\begin{matrix} \infty & 10 & \infty & \infty \\ \infty & \infty & 10 & \infty \\ -30 & -20 & \infty & 10 \\ \infty & \infty & \infty & \infty \end{matrix}$	$\begin{matrix} \infty & 10 & 20 & \infty \\ \infty & \infty & 10 & \infty \\ -30 & -20 & -10 & 10 \\ \infty & \infty & \infty & \infty \end{matrix}$	$\begin{matrix} -10 & 0 & 20 & 30 \\ -20 & -10 & 10 & 20 \\ -30 & -20 & -10 & 10 \\ \infty & \infty & \infty & \infty \end{matrix}$
<b>X</b>	<b>X po <math>k = 1</math></b>	<b>X po <math>k = 2</math></b>	<b>X po <math>k = 3</math></b>
$\begin{matrix} - & 1 & - & - \\ - & - & 2 & - \\ 3 & - & - & 3 \\ - & - & - & - \end{matrix}$	$\begin{matrix} - & 1 & - & - \\ - & - & 2 & - \\ 3 & 1 & - & 3 \\ - & - & - & - \end{matrix}$	$\begin{matrix} - & 1 & 2 & - \\ - & - & 2 & - \\ 3 & 1 & 2 & 3 \\ - & - & - & - \end{matrix}$	$\begin{matrix} 3 & 1 & 2 & 3 \\ 3 & 1 & 2 & 3 \\ 3 & 1 & 2 & 3 \\ - & - & - & - \end{matrix}$

Už po vypočítaní tretej dvojice tabuliek sa na diagonále matice **C** objavilo na mieste  $c_{33}$  záporné číslo -10, čo stačí na konštatovanie, že skúmaný digraf obsahuje cyklus so zápornou cenou, ktorý obsahuje vrchol 3. Tretí riadok matice smerníkov **X** hovorí, že predposledný vrchol tohto cyklu je  $x_{33} = 2$ , bezprostredne pred vrcholom 2 leží na hľadanom cykle vrchol  $x_{32} = 1$  a pre ním je vrchol  $x_{31} = 3$ , v ktorom sa cyklus uzaviera. Hľadaný cyklus so zápornou cenou je teda

$$3, (3, 1), 1, (1, 2), 2, (2, 3), 3.$$

Ak chceme zistiť všetky vrcholy ležiace na zápornom cykle, treba druhý krok Floydovho algoritmu dopočítať pre všetky  $k = 1, 2, \dots, n$ , kde  $n = |V|$  je počet vrcholov skúmaného digrafu.

### 3.8 Hľadanie cyklov zápornej ceny v grafe

Najkratšiu cestu v grafe  $G = (V, H, c)$  sme hľadali doteraz tak, že sme graf  $G$  nahradili digrafom  $G' = (V, H', c')$  s rovnakou vrcholovou množinou  $V$  a hranovou množinou  $H'$  obsahujúcou ku každej hrane  $h \in H$  dvojicu opačne orientovaných hrán, obe s rovnakou cenou  $c(h)$ . Ak v digrafe  $G'$  nájdeme najkratšiu orientovanú  $u-v$  cestu  $m'(u, v)$ , potom tejto ceste jednoznačne prislúcha najkratšia  $u-v$  cesta  $\mu(u, v)$  v pôvodnom grafe  $G$ , ktorú z orientovanej cesty  $m'(u, v)$  dostaneme tak, že „zabudneme“ na orientáciu jej hrán.

Pri použití tohto postupu pre grafy  $G$  so všeobecným ohodnotením hrán však pre každú hranu  $h \in H$ ,  $h = \{u, v\}$  so zápornou cenou  $c(h) < 0$  máme v digrafe  $G'$  dve hrany  $(u, v)$ ,  $(v, u)$  obe so záporným ohodnotením, ktoré tvoria cyklus zápornej ceny  $u, (u, v), v, (v, u), u$ . Tomuto cyklu po „zabudnutí“ orientácie hrán prislúcha sled  $u, \{u, v\}, v, \{v, u\}, u$ , ktorý však nie je cyklom v

pôvodnom grafe  $G$ , pretože dvakrát obsahuje tú istú hranu. Pri orientovaných cykloch s viacerými než dvoma hranami sa táto situácia už nemôže stať. Graf  $G$  obsahuje záporný cyklus práve vtedy, keď digraf  $G'$  obsahuje záporný cyklus s aspoň tromi hranami.

Základný algoritmus 3.2 a tiež Fordov algoritmus 3.3 možno ľahko upraviť tak, aby hľadali v digrafe  $G'$  len cykly s viac ako dvoma hranami takto: Ak nájdeme orientovanú hranu  $(i, j)$  takú, že  $t(j) > t(i) + c(i, j)$  a takú, že  $x(i) = j$ , po vykonaní  $t(j) := t(i) + c(i, j)$  a  $x(j) := i$  by vznikol dvojhranový neželaný cyklus  $j, (j, i), i, (i, j), j$ . Preto takéto preznačkovanie zakážeme.

Rozhodovanie o zmene značiek  $t()$  a  $x()$  bude teda nasledovné:

Ak  $t(j) > t(i) + c(i, j)$  a  $x(i) \neq j$ , potom  $t(j) := t(i) + c(i, j)$ ,  $x(j) := i$ .

Teraz stačí doplniť takto zmenený základný resp. Fordov algoritmus o kontrolu vzniku cyklu po každej zmene značiek  $t()$  a  $x()$ , čím dostaneme algoritmus na zistenie existencie záporného orientovaného cyklu s viac než dvoma hranami v digrafe  $G'$  a teda aj na odhalenie záporného cyklu v pôvodnom grafe  $G$ .

### 3.9 Cesta maximálnej spoľahlivosti

**Definícia 3.18.** Majme hranovo ohodnotený graf  $G = (V, H, c)$  kde ohodnotenie  $c$  predstavuje spoľahlivosť hrany (pravdepodobnosť úspešného prechodu hranou), t. j.  $0 \leq c(h) \leq 1$ . Nech  $\mu(u, v)$  je  $u$ - $v$  cesta. **Spoľahlivosť**  $s(\mu(u, v))$  cesty  $\mu(u, v)$  definujeme:

$$s(\mu(u, v)) = \prod_{h \in \mu(u, v)} c(h).$$

$u$ - $v$  **cesta maximálnej spoľahlivosti** je tá  $u$ - $v$  cesta  $\mu(u, v)$ , ktorá má zo všetkých  $u$ - $v$  ciest najväčšiu spoľahlivosť.

**Veta 3.9.** Nech  $G = (V, H, c)$ , kde  $c(h) > 0$  je spoľahlivosť hrany  $h \in H$ .  $u$ - $v$  cesta  $\mu(u, v)$  je cestou maximálnej spoľahlivosti v grafe  $G = (V, H, c)$  práve vtedy, ak  $\mu(u, v)$  je najkratšou cestou v grafe  $\bar{G} = (V, H, \bar{c})$ , kde pre cenu hrany  $\bar{c}$  platí  $\bar{c}(i, j) = -\log_z(c(i, j))$ , (kde  $z > 1$ ).

**DÔKAZ.**

Pretože je funkcia  $\log_z(x)$  rastúcou funkciou (pre základ  $z > 1$ ), je spoľahlivosť  $s(\mu(u, v)) = \prod_{h \in \mu(u, v)} c(h)$  cesty  $\mu(u, v)$  maximálna práve vtedy, keď je

maximálny výraz:

$$\log (s(\boldsymbol{\mu}(u, v))) = \log \left( \prod_{h \in \boldsymbol{\mu}(u, v)} c(h) \right) = \sum_{h \in \boldsymbol{\mu}(u, v)} \log (c(h)).$$

Posledný výraz je maximálny práve vtedy, keď je súčet  $\sum_{h \in \boldsymbol{\mu}(u, v)} -\log (c(h))$  minimálny. Pretože  $0 < c(h) \leq 1$ , je  $-\log (c(h))$  nezáporné číslo. Pre cestu  $\boldsymbol{\mu}(u, v)$  je  $\sum_{h \in \boldsymbol{\mu}(u, v)} -\log (c(h))$  vlastne dĺžkou cesty v hranovo ohodnotenom grafe  $G = (V, H, c)$  s cenou hrany  $\bar{c}(h) = -\log (c(h))$ . ■

*Poznámka.* Pri predchádzajúcej metóde je jedno, pri akom základe  $z$  je braný logaritmus ceny hrany, pokiaľ je funkcia  $\log_z(x)$  rastúca, k čomu stačí, aby  $z > 1$ .

*Poznámka.* V prípade, že v grafe  $G = (V, H, c)$  existujú hrany so spoľahlivosťou 0, dopredu ich z hranovej množiny vylúčime, pretože sú nepoužiteľné, lebo ich možno úspešne absolvovať s pravdepodobnosťou 0.

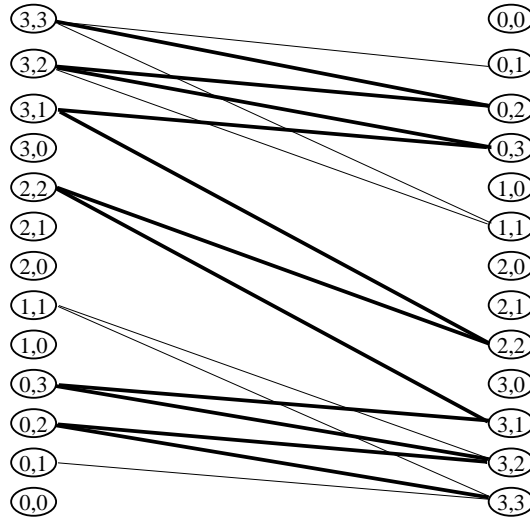
## 3.10 Aplikácie

### 3.10.1 Misionári a kanibali

Traja misionári a traja kanibali sedia na jednom brehu rieky a rozmýšľajú, ako sa dostanú na druhý breh. Majú so sebou dvojmiestnu loďku. Ich problémom však je, že ak kanibali prečísli misionárov, bude hostina na účet misionárov. Ako sa majú všetci šiesti prepraviť bez úhony na druhý breh?

Na riešenie tohto problému zostrojíme bipartitný graf možných konfigurácií misionárov a kanibalov na oboch brehoch rieky. Jednej konfigurácii zodpovedá vždy dvojica vrcholov na rovnakej úrovni. Vrcholy grafu sú usporiadané dvojice  $(m, k)$ , kde  $m$  je počet misionárov a  $k$  počet kanibalov. Dvojice  $(m, k)$ , kde  $0 < m < k$  sú zakázané, pretože v tom prípade by kanibali prečísli misionárov. Medzi vrcholmi na ľavej a pravej strane bipartitného grafu vedieme hranu, ak sa z jednej konfigurácie možno dostať do druhej. Riešenie problému budeme hľadať ako cestu z vrchola  $(3, 3)$  na ľavej strane grafu do vrchola  $(3, 3)$  na pravej strane grafu. Riešenie je na obrázku 3.14 vyznačené hrubými čiarami.

Čitateľ by tu mohol namietať: Stojí mi to zato na riešenie zábavnej úlohy konštruovať tak zložitý graf? A keď dôjde k skutočne závažnej úlohe hľadať optimálny spôsob ako sa dostať v zložitom systéme zo začiatočného stavu do niektorého žiadaného stavu, graf bude tak zložitý, že nebude ručne riešiteľný.



Obr. 3.14: Graf na riešenie problému misionárov a kanibalov.

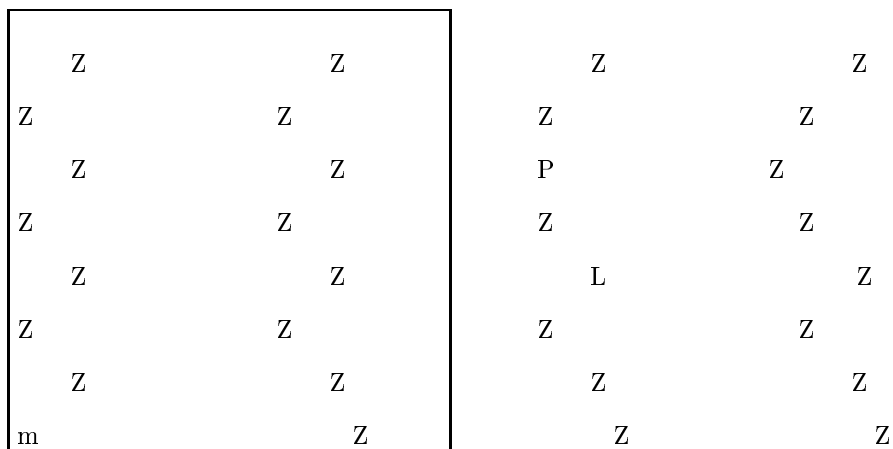
Tu je potrebné povedať, že metódy teórie grafov sú určené hlavne pre počítače. Pamäťová kapacita a rýchlosť počítačov umožnia ku každému stavu vypočítať a uložiť množinu susedných stavov, ktoré sú dostupné z daného stavu. To zodpovedá uloženiu grafu v tvare okolí alebo výstupných hviezd vrcholov. A takáto reprezentácia je mimoriadne vhodná pre grafové algoritmy hľadajúce optimálne cesty.

### 3.10.2 Jazdec na šachovnici

Programátor programuje počítačovú hru, ktorá sa hrá na štandardnej šachovnici so 64 políčkami. Súčasťou akcií počítača je zistiť, či je možné dostať sa jazdcom zo súčasného políčka na iné políčko, a ak áno, potom nájsť príslušnú cestu s najmenším počtom ťahov.

Ako model pre riešenie práve sformulovanej úlohy použijeme graf  $G$  so 64 vrcholmi, každý vrchol bude prislúchať jednému políčku šachovnice. Dva vrcholy  $i, j$  prehlásime za susedné, ak sa z políčka  $i$  na políčko  $j$  možno dostať jedným ťahom jazdca. Vrcholy obsadené inými figúrkami resp. vrcholy napadnuté nepriateľskými figúrkami z grafu  $G$  vylúčime. Dostaneme tak graf





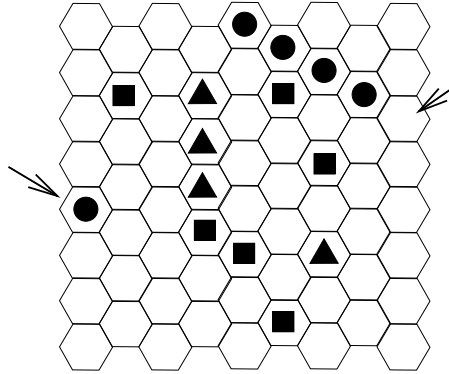
Obr. 3.15: Existuje spôsob, akým sa jazdec dostane po neobsadených a nenapadnutých políčkach do opačného rohu?

$\overline{G}$ , v ktorom hľadáme najkratšiu cestu z vrchola prislúchajúceho štartovaciemu políčku do vrchola cieľového políčka.

Podobným spôsobom by sme postupovali pri programovaní nasledujúcej počítačovej hry. Hrá sa na šachovnici zloženej zo šesťuholníkových buniek podobnej včeliemu plástu. Počítač v každom kroku vygeneruje náhodne do jedného políčka jeden z niekoľkých druhov symbolov (farebné krúžky, ovocie atď.). Hráč v tomto kroku určí, ktorý symbol z ktorého miesta kam presunúť. Hráč sa snaží dostať do jednej línie päť alebo viac symbolov. Akonáhle sa mu to podarí, symboly v línii uvoľnia svoje miesta (zmiznú) a hráčovi sa zvýši skóre. Presun symbolov je možný len do susednej bunky cez spoločnú hranu buniek.

Matematickým modelom problému bude graf  $G$ , ktorého vrcholy sú bunky. Vrcholy  $i, j$  grafu  $G$  budú tvoriť hranu práve vtedy, keď bunky  $i, j$  majú spoločnú hranu. V tomto prípade dostaneme dokonca rovinný graf. Ku grafu  $G$  zostrojíme graf  $\overline{G}$  tak, že z grafu  $G$  vynecháme všetky obsadené vrcholy a s nimi incidentné hrany okrem vrcholu s objektom, ktorý chceme presunúť. V grafe  $\overline{G}$  hľadáme najkratšiu cestu z vrchola príslušného k začiatkovej bunke do vrchola zodpovedajúceho koncovkej bunke.

V oboch prípadoch je vhodné použiť Dijkstrov algoritmus, ktorý zastavíme ihneď po definitívnom označení koncového vrchola hľadanej najkratšej cesty. Ak príslušná cesta neexistuje, algoritmus skončí bez označenia koncového vrchola.



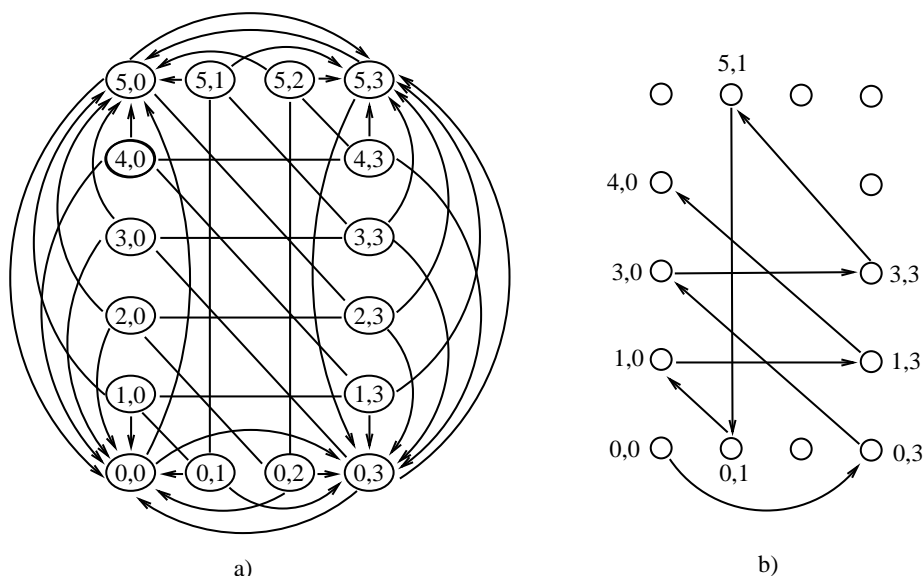
Obr. 3.16: Možno premiestniť označený čierny krúžok do označenej bunky?

### 3.10.3 Odmeriavanie vody

Sme na brehu jazera. Máme k dispozícii dve nádoby – jednu 5-litrovú a druhú 3-litrovú. Našou úlohou je odmerať presne 4 litre vody.

Stav nádob možno charakterizovať dvojicou celých čísel  $(p, q)$ , kde  $p$  je obsah vody v 5-litrovej nádobe a  $q$  obsah vody v 3-litrovej nádobe. Keď chceme vedieť, koľko vody sme preliali z nádoby do nádoby, môžeme to urobiť len tak, že jednu nádobu úplne vyprázdňujeme, alebo druhú úplne naplníme. Preto stavy, do ktorých sa môžeme dostať, sú charakterizované iba dvojicami  $(0, 0)$ ,  $(5, 0)$ ,  $(5, 1)$ ,  $(5, 2)$ ,  $(5, 3)$  a  $(0, 3)$ ,  $(1, 3)$  až  $(5, 3)$ . Z každého stavu  $(p, q)$  môžeme prejsť do stavu  $(0, q)$  alebo do stavu  $(p, 0)$  tak, že vodu z príslušnej nádoby vylejeme do jazera. Z ľubovoľného stavu  $(p, q)$  prejdeme do stavov  $(5, q)$  resp.  $(p, 3)$  tak, že prázdnu nádobu úplne doplníme vodou z jazera. Zo stavu  $(5, p)$  môžeme prejsť do stavu  $(5 - 3 + p, 3)$ , tak, že 3-litrovú nádobu doplníme vodou z 5-litrovej. Podobne možno prejsť zo stavu  $(q, 3)$  do stavu  $(q + 3, 0)$ , ak  $q \leq 2$  alebo do stavu  $(5, q + 3 - 5)$ , ak  $q \geq 3$ .

Zostrojíme digraf  $\vec{G}$ , ktorý bude mať za vrcholy všetky prípustné stavy nádob a ktorý bude mať za množinu hrán všetky usporiadané dvojice stavov



Obr. 3.17: Digraf modelujúci stavy pre problém odmeriavania vody.

$((p_1, q_1), (p_2, q_2))$ , také že zo stavu  $(p_1, q_1)$  možno prejsť do stavu  $(p_2, q_2)$  jediným preliatím. Digraf  $\vec{G}$  je na obrázku 3.17. Našu úlohu preformulujeme ako úlohu nájsť v digrafe  $\vec{G}$  cestu z vrchola  $(0, 0)$  do vrchola  $(4, 0)$ .

### 3.10.4 Zalamovanie odstavca v typografii

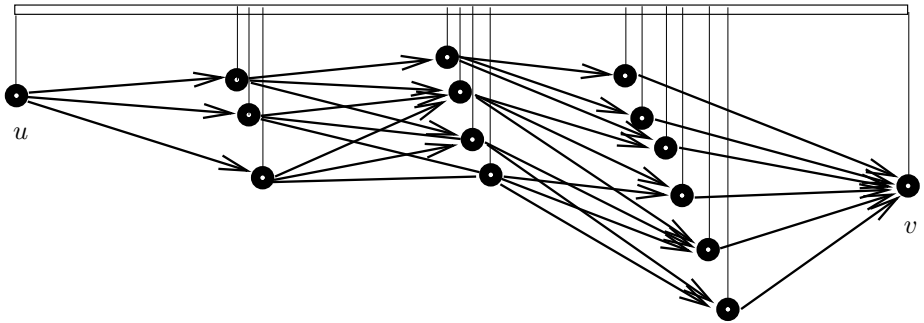
Väčšina súčasných textových editorov má aj činnosť, ktorá sa volá zalamovanie odstavcov. Používateľ sa nemusí starať, kedy skončí riadok – editor na vhodnom mieste ukončí riadok a pokračuje v ďalšom riadku. Ak používateľ žiada zarovnávanie pravého okraja textu, editor to urobí tak, že rovnomerne rozťahne medzislovné medzery. Väčšina textových editorov preruší riadok v okamihu, keď sa ďalší materiál do riadku nezmestí.

Typografický systém  $\text{T}_{\text{E}}\text{X}$  určený hlavne na písanie matematického textu, ktorým je vysádzaná aj táto publikácia, zalamuje riadky inak.

Práve vstupovaný riadok je možné zalomiť na viacerých miestach. Sú to medzery medzi slovami, a v prípade, keď je dovolené deliť slová, aj miesta, v ktorých

sa dajú slová rozdeliť. Prvé možné zalomenie by dávalo príliš veľké medzislovné medzery a riadok by bol neestetický. Pri poslednom možnom zalomení by zas riadok mal príliš stesnané medzery medzi slovami a zas by nebol pekný. Odchýlku od ideálneho stavu možno matematicky vyjadriť ako číslo a chápať ako „škaredosť“ riadku. Čím vyššia hodnota „škaredosti“, tým menej pekný riadok.

Nám ide nielen o pekný riadok, ale aj o pekný celý odstavec. Tu sa môže stať, že keď zvolíme ideálny prvý riadok, druhý riadok sa nám nepodarí pekne zalomiť. Malým zľavnením krásy prvého riadku môžeme dostať podstatné zlepšenie ďalších riadkov.



Obr. 3.18: Digraf pre hľadanie optimálneho zalomenia odstavca.

Na toto používa  $\text{\TeX}$  nasledujúci model teórie grafov. Z textu celého odstavca urobí jeden veľmi dlhý pomyselný riadok. Začiatok a koniec riadka označí ako vrcholy  $u, v$  digrafu  $\vec{G}$ . K začiatku riadka nájde všetky možné miesta pre zalomenie prvého riadku. Každému takémuto miestu priradí vrchol digrafu a hranu vedúcu zo začiatku do tohto vrchola. Hranu ohodnotí „škaredosťou“ riadku, ktorý vznikne zalomením textu na mieste. Ďalej ku každému zalomeniu odpovedajúcemu existujúcim vrcholom nájde všetky možné zalomenia, pre ne definuje ďalšie vrcholy digrafu, a medzi vrcholmi zavedie orientovanú hranu s ohodnotením „škaredosti“ riadku, ktorý by vznikol medzi zalomeniami príslušnými k týmto vrcholom. Takto postupne vytvorí digraf, kým sa nevyčerpajú všetky možnosti zalomení a kým sa nedosiahne vrchol  $v$  – pozri obrázok 3.18.

Teraz si povieme, že najkrajší odstavec dostaneme, keď súčet „škaredostí“ jednotlivých riadkov bude minimálny. Každému zalomeniu riadkov odstavca zodpovedá v digrafe  $\vec{G}$  niektorá orientovaná  $u-v$  cesta. Najkrajší odstavec bude

teda určený najkratšou orientovanou  $u-v$  cestou. Určiť optimálne zalomenie odstavca teda znamená nájsť v digrafe  $\vec{G}$  najkratšiu  $u-v$  cestu.

Vidíme, že teóriu grafov sa dá riešiť aj estetický problém, ktorý je zdanlivo veľmi ďaleko od matematiky.

### 3.10.5 Elektronický cestovný poriadok

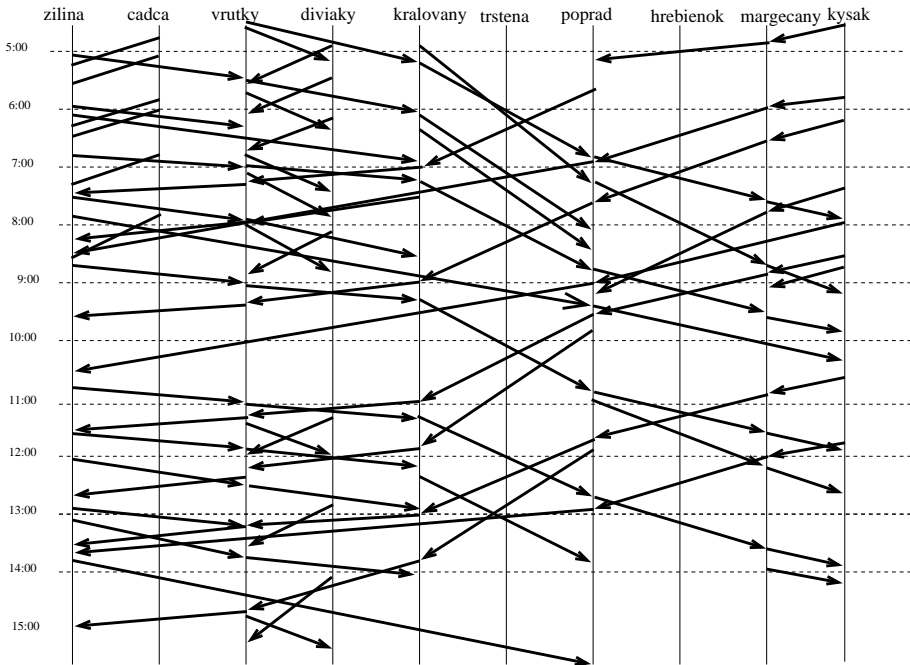
Často sa stretávame s veľmi praktickou úlohou nájsť vlakové, autobusové alebo i kombinované spojenie medzi dvoma miestami. Úloha má niekoľko variantov:

- Kedy najneskôr odísť z miesta A aby som bol v mieste B do hodiny H?
- Kedy sa najskôr dostanem z miesta A do miesta B ak cestu začnem v hodine H?
- Ktoré spojenie z miesta A do miesta B trvá najmenej?
- Ktoré spojenie z miesta A do miesta B je najlacnejšie?

Pre jednoduchosť sa najprv obmedzíme iba na železničnú dopravu. Zostrojíme digraf, v ktorom sú vrcholmi všetky usporiadané dvojice typu  $(s, t)$ , kde  $s$  je železničná stanica a  $t \in \langle 0, 1440 \rangle$  je celé číslo označujúce minútu dňa. V digrafe zavedieme orientované hrany dvojakým spôsobom. Hrany prvého druhu budú všetky hrany typu  $((s, t_1), (s, t_2))$ , kde  $t_1 < t_2$ . Tieto hrany vyjadrujú skutočnosť, že ak niekto je na stanici  $s$  v čase  $t_1$  (t. j. je v stave  $(s, t_1)$ ) môže byť na tej istej stanici aj v každom čase  $t_2 > t_1$  (t. j. môže sa dostať do stavu  $(s, t_2)$ ). Hranu druhého druhu  $((s_1, t_1), (s_2, t_2))$  zavedieme práve vtedy, keď zo stanice  $s_1$  v čase  $t_1$  odchádza priamy spoj do stanice  $s_2$ , ktorý tam prichádza v čase  $t_2$ . Hľadanie spojenia medzi stanicami  $s_1, s_2$  sa takto prevedie na hľadanie cesty z niektorého vrchola  $(s_1, t_1)$  do niektorého vrchola  $(s_2, t_2)$ . V prípade, že hľadáme najlacnejšie spojenie, stačí ohodnotiť hrany nášho digrafu príslušným cestovným, a potom hľadať najkratšiu cestu v takto ohodnotenom digrafe.

Ak by sme konštruovali podobný digraf pre autobusovú (resp. leteckú či vodnú dopravu), zostavujeme digraf úplne rovnako, len namiesto železničných staníc budeme mať autobusové zastávky (resp. letiská alebo prístavy).

Ak by sme nakoniec chceli vytvoriť digraf pre modelovanie kombinovaných spojení, zjednotíme vrcholové a hranové množiny digrafov pre jednotlivé druhy



Obr. 3.19: Digraf k úlohe hľadania spojenia v cestovnom poriadku.

dopráv. Musíme však pridať hrany tretieho druhu, ktoré budú modelovať možnosť dostať sa (peši, mestskou hromadnou dopravou, taxíkom, alebo iným spôsobom) zo železničnej stanice na blízku autobusovú zastávku, letisko, či prístav. Takéto hrany musíme dodať medzi všetkými terminálovými miestami všetkých druhov dopráv.

Každý, kto sa pokúsil hľadať najmä autobusové spojenie v neznámom regióne, bude isto súhlasiť, že problém dopravného informačného systému je už vážnym problémom. Všimnime si, že spôsob jeho riešenia je úplne rovnaký, ako problém misionárov a kanibalov – zostaviť graf alebo digraf, ktorého vrcholy sú stavy a ktorého hrany sú možné prechody z jedného stavu do susedného stavu, a kde sa pôvodná úloha rieši hľadaním cesty z daného začiatočného do želaného koncového stavu.

### 3.11 Cvičenia

1. O koľko môže vzrásť počet komponentov grafu po vybratí hrany – mostu? O koľko po vybratí vrchola, ktorý je artikuláciou? Prečo?
2. Ukážte, že každý cyklus v bipartitnom grafe má párny počet vrcholov a hrán.
3. Navrhnite tabuľkový výpočtový postup pre hľadanie všetkých najkratších  $u$ - $v$  ciest Fordovým algoritmom 3.3, ktorý bude modifikáciou postupu z príkladu 3.2 (strana 77).
4. Nech  $G$  je ľubovoľný graf. Nech  $\mathcal{D}$  je množina všetkých pravidelných podgrafov grafu  $G$  so všetkými vrcholmi párneho stupňa spolu s dvojicou  $O = (\emptyset, \emptyset)$ . ( $O$  nie je grafom, pretože v definícii grafu sa požaduje, aby jeho vrcholová množina bola neprázdna.) Pre  $D_1 \in \mathcal{D}$ ,  $D_2 \in \mathcal{D}$ ,  $D_1 = (V_1, H_1)$ ,  $D_2 = (V_2, H_2)$  definujeme ich súčet  $D_1 \oplus D_2$  ako minimálny podgraf grafu  $G$  s množinou hrán  $H_1 \Delta H_2 = (H_1 - H_2) \cup (H_2 - H_1)$ , ak  $H_1 \Delta H_2 \neq \emptyset$ ,  $D_1 \oplus D_2 = O$ , ak  $H_1 \Delta H_2 = \emptyset$ . Ďalej definujeme  $0.D = O$ ,  $1.D = D$  pre každé  $D \in \mathcal{D}$ . Ukážte, že  $(\mathcal{D}, \oplus, \cdot)$  je lineárny priestor nad telesom  $\mathbb{Z}_2$ .
5. Nech  $G$  je ľubovoľný graf, nech  $\mathcal{P}$  je množina všetkých pravidelných podgrafov grafu  $G$  stupňa 1 spolu s prvkom  $O = (\emptyset, \emptyset)$ . Pre  $P_1 \in \mathcal{P}$ ,  $P_2 \in \mathcal{P}$  definujeme  $P_1 \oplus P_2$  analogicky, ako v príklade 4. Je vždy  $P_1 \oplus P_2 \in \mathcal{P}$ ? Ako vyzerajú komponenty  $P_1 \oplus P_2$  v prípade, že  $P_1 \oplus P_2 \neq O$ ?
6. Sedliak preváža cez rieku kozu, vlka, kapustu a seba. V lodke sú však iba dve miesta, takže okrem seba môže zobrať iba kozu, vlka alebo kapustu. Nesmie nechať bez dozoru kozu s vlkom ani kapustu s kozou, pretože vlk by v tom prípade zožral kozu, resp. koza kapustu. Ako má previezť všetko bez ujmy cez rieku?
7. Tri žiarlivé manželské páry sa chcú prepraviť cez rieku v dvojmiestnej lodke. Ako to majú urobiť tak, aby sa žiadna žena neocitla sama v prítomnosti cudzieho muža?
8. Ako vyriešime úlohu o prelievaní vody, ak chceme aby sme to urobili
  - a) s najmenším možným počtom prelievaní
  - b) tak, aby sme čo najmenej vody vyliali späť do jazera.

9. Ako zmodifikujeme graf k hľadaniu železničného spojenia ak sa chceme zabezpečiť proti  $m$  minútovému meškaniu vlakov?

### Počítačové cvičenia

10. Napíšte program na generovanie náhodného hranovo ohodnoteného grafu s  $n$  vrcholmi  $1, 2, \dots, n$  a  $m$  hranami ( $n$  a  $m$  sú vstupné parametre,  $n \approx 500$ ). Využite Tarryho algoritmus na zistenie komponentov vygenerovaného grafu. Napíšte program na zistenie všetkých mostov a všetkých artikulácií vygenerovaného grafu.
11. Implementujte vo vhodnom programovacom jazyku základný, Dijkstrov a Floydov algoritmus na hľadanie vzdialenosti dvoch vrcholov i na výpočet matice vzdialeností pre graf i digraf. Porovnajzte rýchlosť jednotlivých algoritmov. Zistite, akou činnosťou trávi najviac času Dijkstra algoritmus a skúste ho zrýchliť. Rozoberte vplyv rôznych reprezentácií grafu, resp. digrafu na rýchlosť algoritmov. Skúšobné príklady by mali mať aspoň 500 až 1000 vrcholov.
12. Modifikujte algoritmy z predchádzajúceho cvičenia na hľadanie záporných cyklov vo všeobecne ohodnotených grafoch a digrafoch.
13. Implementujte label set a label correct algoritmus.



# Kapitola 4

## Acyklické grafy, stromy a kostry

### 4.1 Stromy a ich vlastnosti

**Definícia 4.1.** Triviálny graf je taký graf  $G = (V, H)$ , ktorého množina vrcholov  $V$  pozostáva z jediného vrchola.

*Poznámka.* Ak je  $G = (V, H)$  triviálny graf, potom  $H = \emptyset$ .

**Definícia 4.2.** Acyklický graf je taký graf, ktorý neobsahuje ako podgraf kružnicu.

*Poznámka.* Podľa vety 3.3 (na str. 64) možno všetky vrcholy a hrany kružnice usporiadať do cyklu a naopak, všetky vrcholy a hrany cyklu tvoria graf – kružnicu. Preto platí: Graf  $G$  je acyklický práve vtedy, keď neobsahuje cyklus.

**Definícia 4.3.** Strom je súvislý acyklický graf.

*Poznámka.* Triviálny graf je stromom.

*Poznámka.* Pretože každý komponent acyklického grafu je stromom (je súvislý a neobsahuje kružnicu), možno sa na acyklický graf pozeráť ako na zjednotenie stromov. Od toho je odvodený pojem les, ktorý sa používa ako synonymum pre acyklické grafy.

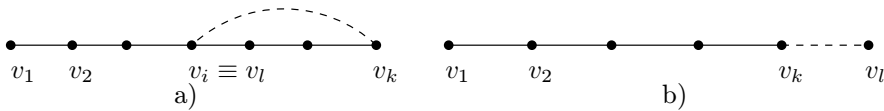
**Veta 4.1.** *Nech  $G = (V, H)$  je strom, ktorý má aspoň dva vrcholy. Potom  $V$  obsahuje aspoň dva vrcholy stupňa 1.*

DŔKAZ.

Nech

$$(v_1, \{v_1, v_2\}, v_2, \dots, \{v_{k-1}, v_k\}, v_k) \quad (4.1)$$

je cesta v strome  $G$  s najväčším počtom hrán. Ukážeme, že  $\deg(v_k) = 1$ .



Obr. 4.1: Keby  $\deg(v_k) \geq 1$ , existovala by aspoň jedna hrana (čiarkovane) incidentná s  $v_k$ , vytvárajúca jednu zo situácií a) alebo b).

Pretože  $\{v_{k-1}, v_k\} \in H$ , je  $\deg(v_k)$  aspoň 1. Keby  $\deg(v_k) > 1$ , musela by existovať hrana  $\{v_k, v_l\}$ , kde  $v_l \neq v_{k-1}$  a tiež  $v_l \neq v_k$ . Keby existovalo  $v_i = v_l$  pre  $i = 1, 2, \dots, v_{k-2}$ , mohli by sme vytvoriť cyklus

$$(v_i, \{v_i, v_{i+1}\}, \dots, \{v_{k-1}, v_k\}, v_k, \{v_k, v_l\}, v_l = v_i),$$

čo je v spore s acykličnosťou stromu  $G$  — pozri obrázok 4.1 a). Je teda  $v_l \neq v_i$  pre všetky  $i = 1, 2, \dots, k$ . Teraz už môžeme zostrojiť cestu

$$(v_1, \{v_1, v_2\}, v_2, \dots, \{v_{k-1}, v_k\}, v_k, \{v_k, v_l\}, v_l),$$

ktorá obsahuje o jednu hrana viac ako cesta (4.1) — pozri b) na obrázku 4.1. Z predpokladu  $\deg(v_k) > 1$  sme dostali spor s predpokladom, že cesta (4.1) má najväčší počet hrán. Je teda  $\deg(v_k) = 1$ .

Podobne dokážeme, že  $\deg(v_1) = 1$ . ■

**Veta 4.2.** *Nasledujúce tvrdenia sú ekvivalentné:*

- $G = (V, H)$  je strom.
- V grafe  $G = (V, H)$  existuje pre každé  $u, v \in V$  jediná  $u$ - $v$  cesta.
- Graf  $G = (V, H)$  je súvislý a každá hrana množiny  $H$  je mostom.
- Graf  $G = (V, H)$  je súvislý a  $|H| = |V| - 1$ .

e) V grafe  $G = (V, H)$  platí  $|H| = |V| - 1$  a  $G$  je acyklický.

DŔKAZ.

Dokazovanie si zjednodušíme, ak budeme dokazovať podľa schémy a)→b)→c)→d)→e)→a).

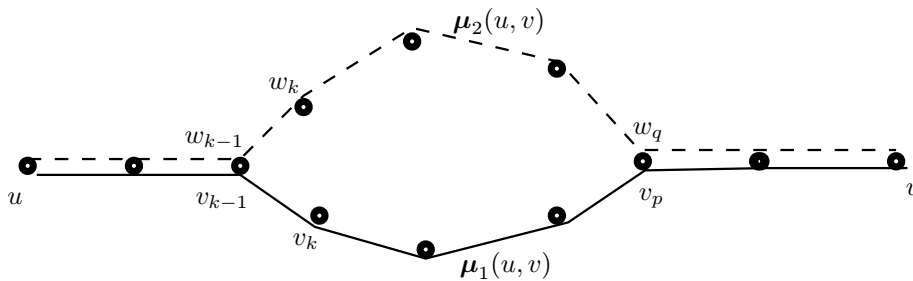
a)→b)

Nech  $G = (V, H)$  je strom. Dôkaz vykonáme sporom. Nech existujú dve rôzne  $u$ - $v$  cesty

$$\mu_1(u, v) = (u = v_1, \{v_1, v_2\}, \dots, \{v_{n-1}, v_n\}, v_n = v)$$

$$\mu_2(u, v) = (u = w_1, \{w_1, w_2\}, \dots, \{w_{m-1}, w_m\}, w_m = v)$$

.



Obr. 4.2: Z existencie dvoch ciest vyplýva existencia cyklu.

Nech  $k$  je najmenšie prirodzené číslo také, že  $v_k \neq w_k$  a  $v_1 = w_1, v_2 = w_2, \dots, v_{k-1} = w_{k-1}$ . (Po vrchol  $v_{k-1}$  resp.  $w_{k-1}$  obe cesty idú cez rovnaké vrcholy a hrany, ale vo vrchole  $v_{k-1} = w_{k-1}$  sa rozchádzajú.) Označme  $p$  prvý index väčší než  $k$ , kedy je vrchol  $v_p$  vrcholom cesty  $\mu_2(u, v)$ . Nech  $q$  prvý index väčší než  $k$ , kedy je vrchol  $w_q$  vrcholom cesty  $\mu_1(u, v)$ . Zrejme je  $v_p = w_q$ . Zreťazenie ciest

$$(w_{k-1} = v_{k-1}, \{v_{k-1}, v_k\}, \dots, \{v_{p-1}, v_p\}, v_p = w_q)$$

$$(v_{k-1} = w_{k-1}, \{w_{k-1}, w_k\}, \dots, \{w_{q-1}, w_q\}, w_q)$$

dáva cyklus, čo je spor s predpokladom, že  $G$  je acyklický graf.

b)→c)

Nech v grafe  $G = (V, H)$  existuje pre každé  $u, v \in V, u \neq v$ , jediná  $u$ - $v$  cesta.

Súvislosť grafu  $G$  je zrejmá z existencie  $u$ - $v$  cesty pre každé  $u, v \in V$ .

Ďalej pokračujeme dôkazom sporom. Nech existuje hrana  $\{u, v\} \in H$ , ktorá nie je mostom. Po odstránení hrany  $\{u, v\}$  graf ostáva súvislý - t. j. existuje  $u$ - $v$  cesta  $\mu(u, v)$  neobsahujúca hranu  $\{u, v\}$ , pričom cesta  $\mu(u, v)$  je rôzna od cesty  $u, \{u, v\}, v$ , čo je spor s predpokladom existencie jedinej  $u$ - $v$  cesty.

c)→d)

Nech  $G = (V, H)$  je súvislý graf a každá hrana množiny  $H$  je mostom. Chceme ukázať, že  $|H| = |V| - 1$ . Budeme postupovať matematickou indukciou podľa počtu vrcholov  $|V| = n$ .

Nech  $n = |V| = 1$ , potom ide o triviálny graf, ktorý je stromom a v ktorom je  $|H| = 0$  a tvrdenie platí.

Nech tvrdenie platí pre všetky stromy  $G' = (V', H')$ , kde  $|V'| < n + 1$ . Nech  $G = (V, H)$  je súvislý graf, v ktorom je každá hrana mostom a kde  $|V| = n + 1$ . Vezmime ľubovoľnú hranu  $h = \{u, v\} \in H$ . Po jej odstránení z množiny hrán sa graf  $G$  rozpadne na dva komponenty  $G_1 = (V_1, H_1)$ ,  $G_2 = (V_2, H_2)$  také, že  $u \in V_1$ ,  $v \in V_2$ . Pretože  $V_1$  neobsahuje vrchol  $v$  a  $V_2$  neobsahuje vrchol  $u$ , je  $|V_1| \leq n$ ,  $|V_2| \leq n$  a v oboch grafoch je každá hrana mostom. Oba teda spĺňajú požiadavky indukčného predpokladu, a preto je

$$|H_1| = |V_1| - 1$$

$$|H_2| = |V_2| - 1$$

$$\text{a preto } |H_1| + |H_2| = |V_1| + |V_2| - 2$$

$$\text{Pretože } |H| = |H_1| + |H_2| + 1$$

$$\text{je } |H| = |V_1| + |V_2| - 1 = |V| + 1$$

d)→e)

V tomto stave dôkazu už máme dokázané a) → b) → c) → d) a teda aj a) → d), čo možno formulovať ako tvrdenie: V strome  $G = (V, H)$  platí  $|H| = |V| - 1$ . Toto použijeme v tejto časti dôkazu.

Nech  $G = (V, H)$  je súvislý graf, v ktorom platí  $|H| = |V| - 1$ . Nepriamo ukážeme, že  $G$  je acyklický. Keby v grafe  $G$  existoval cyklus, odstránime z množiny hrán  $H$  jeho ľubovoľnú hranu  $h_1$ , čím neporušíme súvislosť. Položme  $H_1 = H - \{h_1\}$ ,  $G_1 = (V, H_1)$ .  $G_1$  je súvislý graf. Ak  $G_1$  neobsahuje cyklus, končíme. Inak z množiny hrán  $H_1$  odstránime niektorú hranu cyklu  $h_2$  a položíme  $H_2 = H_1 - \{h_2\}$ ,  $G_2 = (V, H_2)$ . Po odstránení konečného počtu  $k$  hrán

dostaneme súvislý acyklický graf  $G_k = (V, H_k)$ , v ktorom je  $|H_k| = |H| - k$  a ktorý je stromom, takže platí súčasne:

$$\begin{aligned} |H| - k &= |H_k| = |V| - 1 \\ |H| &= |V| - 1 \end{aligned}$$

z čoho už vyplýva, že  $k = 0$ , t. j. v grafe  $G$  neexistuje ani jeden cyklus.

e)→a)

Majme acyklický graf  $G = (V, H)$ , v ktorom platí  $|H| = |V| - 1$ . Stačí ukázať, že graf  $G$  je súvislý. Rozložme ho na komponenty

$$G_1 = (V_1, H_1), G_2 = (V_2, H_2), \dots, G_k = (V_k, H_k),$$

ktoré sú súvislé a acyklické, t. j. každý komponent  $G_i$  je stromom, a preto preň platí  $|H_i| = |V_i| - 1$ .

Platí:

$$|V| = \sum_{i=1}^k |V_i| \quad |H| = \sum_{i=1}^k |H_i|.$$

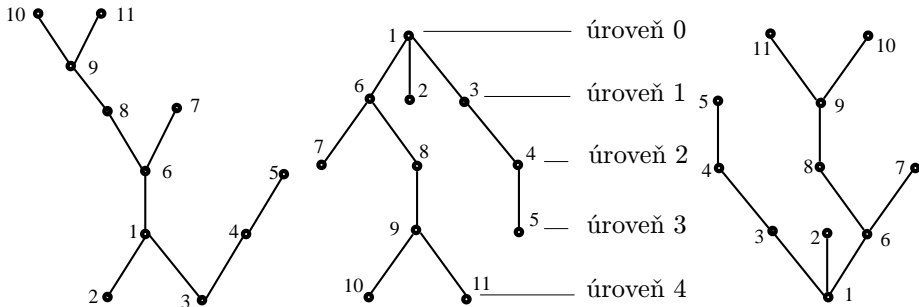
$$|V| - 1 = |H| = \sum_{i=1}^k |H_i| = \sum_{i=1}^k (|V_i| - 1) = \sum_{i=1}^k |V_i| - \sum_{i=1}^k 1 = |V| - k$$

Z posledného vzťahu vychádza  $k = 1$  - t. j. graf  $G$  má 1 komponent, t. j. je súvislý. ■

**Definícia 4.4. Koreňový strom** je strom  $G = (V, H)$  s pevne vybraným vrcholom  $k \in V$ , ktorý nazývame **koreň**. Koreňový strom budeme značiť  $G = (V, H, k)$ . **Úroveň vrchola**  $u$  v koreňovom strome  $G = (V, H, k)$  je dĺžka - počet hrán - (jedinej)  $k$ - $u$  cesty. **Výška koreňového stromu**  $G = (V, H, k)$  je maximum z úrovní všetkých vrcholov koreňového stromu  $G$ .

Diagramy koreňových stromov sa kreslia obvykle tak, že vrcholy nižšej úrovne sa umiestňujú nad vrcholy vyššej úrovne. Bežný je aj opačný spôsob - vrcholy nižšej úrovne pod vrcholy vyššej úrovne.

Majme strom s koreňom  $v_0$ . Nech  $(v_0, \{v_0, v_1\}, v_1, \dots, v_{n-1}, \{v_{n-1}, v_n\}, v_n)$  je (jediná)  $v_0$ - $v_n$  cesta. Potom vrcholy  $v_0, v_1, \dots, v_{n-1}$  sa volajú **predkovia** vrchola  $v_n$ , vrchol  $v_{n-1}$  sa volá **otec** vrchola  $v_n$ , vrchol  $v_n$  sa volá **syn** vrchola  $v_{n-1}$ . Ak je  $x$  predkom  $y$ , potom  $y$  sa volá **následník** alebo **potomok** vrchola  $x$ .



Obr. 4.3: Spôsoby kreslenia diagramu koreňového stromu s koreňom 1.

Vrchol bez následníka sa nazýva **koncový vrchol**, v niektorej literatúre tiež **list**.

Všimnime si ešte, že po vybratí niektorého vrchola za koreň môžeme jednoznačne určiť orientácie všetkých hrán požiadavkou, aby všetky hrany boli orientované tak, že vrchol nižšej úrovne je počiatkový vrchol a vrchol vyššej úrovne je koncový vrchol ľubovoľnej hrany.

Medzi koreňovými stromami majú význačné miesto tzv. **binárne koreňové stromy**, v ktorých má každý vrchol najviac dvoch synov. Často sa títo synovia označujú za ľavého a pravého.

## 4.2 Prehľadávanie grafu do hĺbky a do šírky

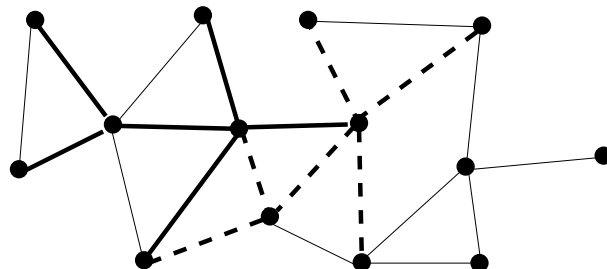
V tretej kapitole sme sa zaoberali Tarryho prieskumom grafov. Tarryho prieskum dáva algoritmický návod, ako systematicky prekontrolovať všetky vrcholy a hrany daného grafu, pričom nám stačí mať informáciu iba o okolí práve preskúmaného vrchola. Poradie vrcholov, v akom sa v Tarryho slede objavujú poprvýkrát, je poradím preskúmania vrcholov, poradie hrán v Tarryho slede určuje poradie preskúmania hrán. Zo striktného matematického hľadiska je Tarryho prieskum grafu algoritmom na zostrojenie uzavretého sledu obsahujúceho každú hranu súvislého grafu práve dvakrát. Tarryho prieskum je súčasťou viacerých zložitejších algoritmov – napríklad algoritmu na zistenie komponentov grafu, uvidíme ho v algoritme zdvojenia kostry pre riešenie úlohy obchodného

cestujúceho, jeho malá modifikácia bude základom pre labyrintový algoritmus na hľadanie uzavretého ľahu obsahujúceho všetky hrany grafu.

Pri mnohých algoritmoch treba systematicky prehľadať iba množinu vrcholov grafu  $G = (V, H)$ . Pre tento účel treba začať v niektorom vrchole  $v \in V$ , položiť  $V' = \{v\}$  a mať pravidlo určujúce, ktorý nepreskúmaný vrchol budeme prehľadať ako ďalší, ak už mám preskúmanú množinu vrcholov  $V' \subseteq V$ . Podobne ako pri Tarryho prieskume by nám pre výber ďalšieho vrchola mala stačiť iba informácia o okoliach vrcholov z množiny  $V'$ .

Existujú dva významné spôsoby prehľadávania grafov – **prehľadávanie do hĺbky** (Depth-First Search) a **prehľadávanie do šírky** (Breadth-First Search). Na rozdiel od Tarryho prieskumu, pri ktorom sa konštruuje Tarryho sled, prehľadávanie do hĺbky a prehľadávanie do šírky postupne konštruujú podstrom  $T$  grafu  $G$ . Tejto konštrukcii sa niekedy hovorí i **pestovanie stromu** (Tree-Growing). Poradie pridávania vrcholov do stromu  $T$  je poradím, v ktorom sa postupne preskúmajú vrcholy grafu  $G$ . Ak je prehľadávanie do hĺbky súčasťou niektorého algoritmu, potom sa spravidla preskúmanie vrchola robí hneď po rozhodnutí o jeho zaradení do stromu  $T$ . Pre opis pestovania stromu sa hodí nasledujúca definícia.

**Definícia 4.5.** Nech strom  $T = (V_T, H_T)$  je podgrafom grafu  $G = (V, H)$ . Hovoríme, že hrana  $h = \{u, v\} \in H$  je **hraničnou hranou**, ak  $u \in V_T$  a  $v \notin V_T$ . Nech  $h = \{u, v\}$  je hraničná hrana,  $u \in V_T$ ,  $v \notin V_T$ . Povieme, že  $u$  je **zaradený vrchol**,  $v$  je **voľný vrchol** hraničnej hrany  $h$ .



Obr. 4.4: Hruba – hrany stromu  $T$ , hruba čiarkovane – hraničné hrany, tenko – ostatné hrany grafu  $G$ .

Oba spôsoby – prehľadávanie do hĺbky i prehľadávanie do šírky – pestujú strom podľa rovnakého princípu. Začneme triviálnym stromom  $T$  obsahujúcim

jediný vrchol  $v \in V$ . Ak už máme nejaký strom  $T$ , rozšírime ho tak, že do jeho hranovej množiny pridáme niektorú hraničnú hranu  $h = \{u, v\}$ ,  $u \in T$ ,  $v \notin T$  a množinu vrcholov stromu  $T$  rozšírime o vrchol  $v$  hrany  $h$  (ten, ktorý neležal v  $T$ ). Pridanému vrcholu  $v$  určíme značku  $p(v)$  poradie, v ktorom sme ho objavili.

Jediným, ale podstatným rozdielom medzi oboma spôsobmi prehľadávania grafu je kritérium pre výber pridávanej hraničnej hrany. Prehľadávanie do hĺbky vyberá hraničnú hranu s najvyššou značkou  $p$  zaradeného konca hraničnej hrany, kým prehľadávanie do šírky vyberá hraničnú hranu s najnižšou značkou  $p$  zaradeného konca hraničnej hrany.

**Algoritmus 4.1. Prehľadávanie grafu  $G = (V, H)$  do hĺbky. (.)**

• **Krok 1. Inicializácia.**

Nech strom  $T$  je triviálny strom obsahujúci jediný vrchol  $v \in V$ .  
Polož  $p(v) := 1$ ,  $k := 1$ .

• **Krok 2.** Ak  $T$  ešte neobsahuje všetky vrcholy grafu, GOTO Krok 3.  
Inak STOP.

• **Krok 3.** V grafe  $G$  so stromom  $T$  nájdí hraničnú hranu  $h = \{u, v\}$  s **maximálnou značkou**  $p(u)$  zaradeného vrchola  $u$ .

• **Krok 4.** Polož  $T := T \cup \{h\} \cup \{v\}$ ,  $k := k + 1$ ,  $p(v) := k$ .  
GOTO Krok 2.



**Algoritmus 4.2. Prehľadávanie grafu  $G = (V, H)$  do šírky. (Breadth-First Search.)**

• **Krok 1.** Inicializácia. Nech strom  $T$  je triviálny strom obsahujúci jediný vrchol  $v \in V$ . Polož  $p(v) := 1$ ,  $k := 1$ .

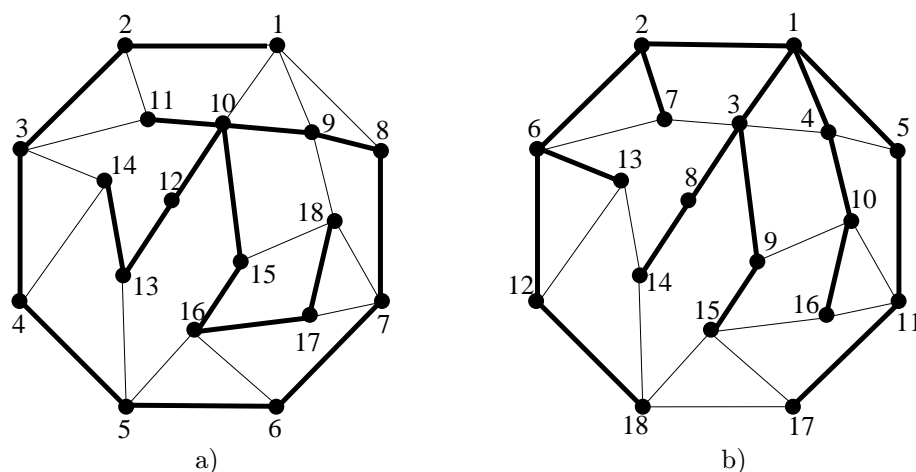
• **Krok 2.** Ak  $T$  ešte neobsahuje všetky vrcholy grafu, GOTO Krok 3.  
Inak STOP.

• **Krok 3.** V grafe  $G$  so stromom  $T$  nájdí hraničnú hranu  $h = \{u, v\}$  s **minimálnou značkou**  $p(u)$  zaradeného vrchola  $u$ .

• **Krok 4.** Polož  $T := T \cup \{h\} \cup \{v\}$ ,  $k := k + 1$ ,  $p(v) := k$ .  
GOTO Krok 2.







Obr. 4.5: a) Prehľadávanie do hĺbky. b) Prehľadávanie do šírky.  
Značky pri vrchoch predstavujú poradie, v akom boli objavené.

Prehľadávanie do hĺbky alebo do šírky sa môže prakticky použiť napríklad pri vyhľadávaní súboru daného mena v štruktúre adresárov na počítačovom disku (pozri aplikáciu 4.5.1 na strane 122). Veľmi často sa niektoré z prehľadávaní použije pri hľadaní optimálneho riešenia diskretných úloh, kde sa prehľadáva pomyselný graf, ktorého vrcholy sú prípustné riešenia a ktorého hrany tvoria dvojice susedných riešení.

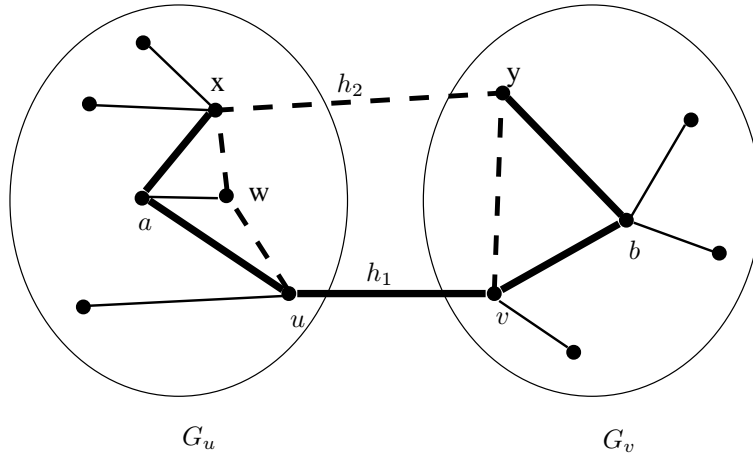
### 4.3 Najlacnejšia a najdrahšia kostra

**Definícia 4.6.** *Kostra* súvislého grafu  $G = (V, H)$  je taký jeho faktorový podgraf, ktorý je stromom. Nech  $G = (V, H, c)$  je hranovo ohodnotený graf,  $K$  kostra grafu  $G$ . **Cena**  $c(K)$  *kostry*  $K$  je súčet ohodnotení jej hrán.

**Najlacnejšia kostra** v grafe  $G$  je kostra s najmenšou cenou.

**Najdrahšia kostra** v grafe  $G$  je kostra s najväčšou cenou.

**Veta 4.3.** *Nech  $K_1, K_2$  sú dve kostry grafu  $G = (V, H)$ . Potom pre každú hranu  $h_1 \in (K_1 - K_2)$  existuje taká hrana  $h_2 \in (K_2 - K_1)$ , že obe hrany  $h_1, h_2$  ležia na (jedinej) kružnici grafu  $K_2 \cup \{h_1\}$  aj na (jedinej) kružnici grafu  $K_1 \cup \{h_2\}$ .*



Obr. 4.6: K dôkazu vety 4.3.

Hrany kostry  $K_1$  sú znázornené plnými čiarami.

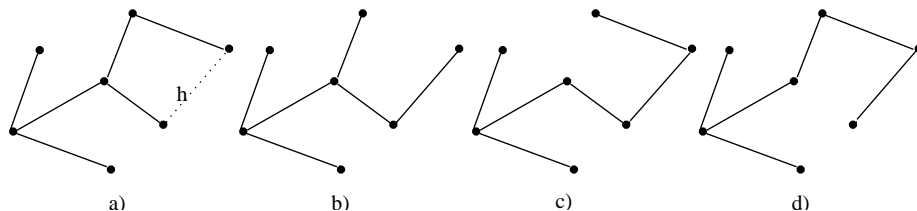
Cesta  $\mu_2(u, v) = (u, \{u, w\}, w, \{w, x\}, x, \{x, y\}, y, \{y, v\}, v)$  po hranách kostry  $K_2$  je znázornená čiarkovane.

## DŔKAZ.

Najprv ukážeme, že ak  $K$ , je kostra súvislého grafu  $G = (V, H)$  a hrana  $h \in H$  nepatrí do hranovej množiny kostry  $K$ , graf  $K \cup \{h\}$  obsahuje jediná kružnicu. Nech  $h = \{u, v\}$ . Ak má v grafe  $K \cup \{h\}$ , existovať kružnica, potom musí obsahovať hrana  $h$ , lebo kostra  $K$  je acyklický graf. Pretože  $K$  je strom, existuje v  $K$   $u-v$  jediná cesta  $\mu(u, v)$ . Zreťazenie cesty  $(v, \{v, u\}, u)$  s cestou  $\mu(u, v)$  je cyklus, graf ním indukovaný je kružnica. Existencia dvoch alebo viacerých kružníc obsahujúcich hrana  $\{u, v\}$  v  $K \cup \{h\}$  by viedla k existencii viacerých  $u-v$  ciest v kostre  $K$ , čo by bolo v spore s tým, že  $K$  je strom.

Nech  $h_1 = \{u, v\}$ . Graf  $K_1 - \{h_1\}$  má dva komponenty  $G_u, G_v$ ,  $u \in G_u$ ,  $v \in G_v$ . Nech  $\mu_2(u, v)$  je jediná  $u-v$  cesta v kostre  $K_2$  (na obrázku čiarkovaná čiara). Táto cesta musí obsahovať aspoň jednu hrana  $h_2 = \{x, y\}$  takú, že  $x \in G_u$  a  $y \in G_v$  (lebo začína v  $G_u$  a končí v  $G_v$ ). Zreťazenie  $(v, \{v, u\}, u) \oplus \mu_2(u, v)$  je cyklus v  $K_2 \cup \{h_1\}$  obsahujúci obe hrany  $h_1, h_2$ . V kostre  $K_1$  existuje jediná  $x-y$  cesta (na obrázku hrubá čiara). Pretože bolo  $x \in G_u$ ,  $y \in G_v$ , t. j.  $x, y$  bolo v rôznych komponentoch grafu  $K_1 - \{h_1\}$ , musí táto cesta obsahovať hrana

$h_1 = \{u, v\}$ . Cyklus  $\mu_1(x, y) \oplus (y, \{y, x\}, x)$  indukuje jediná kružnicu v  $K_1 \cup \{h_2\}$  a obsahuje obe hrany  $h_1, h_2$ . ■



Obr. 4.7: Susedné kostry.

Ak k hranám kostry  $K$  pridáme ľubovoľnú hranu  $h$  grafu  $G$ , ktorá neleží v hranej množine kostry  $K$  (na obrázku 4.7 a) je vyznačená bodkovanou čiarou), vznikne graf  $K \cup \{h\}$  obsahujúci cyklus. Vylúčením ktorejkoľvek hrany tohto cyklu z grafu  $K \cup \{h\}$  (okrem hrany  $h$ ) dostaneme inú kostru  $K'$ .

**Definícia 4.7.** Dve kostry sa nazývajú **susedné**, ak jednu možno získať z druhej výmenou jednej hrany.

**Veta 4.4.** Kostra  $K$  grafu  $G = (V, H, c)$  je najlacnejšia práve vtedy, keď k nej neexistuje lacnejšia susedná kostra.

DŮKAZ.

Keď je kostra  $K$  najlacnejšia, potom nemôže existovať žiadna lacnejšia kostra.

Nech  $K_0$  je najlacnejšia kostra a nech ku kostre  $K$  neexistuje žiadna lacnejšia susedná kostra. Ak by boli obe kostry totožné, bola by aj kostra  $K$  najlacnejšia. Nech teda existuje  $h_1 \in (K - K_0)$ . Podľa vety 4.3 existuje hrana  $h_2 \in (K_0 - K)$  tak, že obe hrany ležia na nejakom cykle v  $K \cup \{h_2\}$  aj na nejakom cykle  $K_0 \cup \{h_1\}$ .

Pretože hrana  $h_1$  leží na cykle v grafe  $K \cup \{h_2\}$ , musí byť

$$c(h_1) \leq c(h_2), \quad (4.2)$$

ináč by nahradením hrany  $h_1$  hranou  $h_2$  v tomto cykle vznikla lacnejšia susedná kostra kostry  $K$ , ale kostra  $K$  podľa predpokladov nemá lacnejšiu susednú kostru.

Pretože hrana  $h_2$  leží na cykle v grafe  $K_0 \cup \{h_1\}$ , musí byť

$$c(h_2) \leq c(h_1), \quad (4.3)$$

ináč by nahradením hrany  $h_2$  hranou  $h_1$  v tomto cykle vznikla lacnejšia susedná kostra kostry  $K_0$ , čo je v spore s predpokladom, že  $K_0$  je najlacnejšia kostra.

Spojením (4.2), (4.3) máme  $c(h_1) = c(h_2)$ . Pretože hrana  $h_2$  kostry  $K_0$  leží na cykle grafu  $K \cup \{h_2\}$  obsahujúcom aj hranu  $h_1$ , jej nahradením hranou  $h_1$  vytvoríme susednú kostru  $K^{(1)}$  kostry  $K_0$  s rovnakou cenou, avšak kostra  $K^{(1)}$  bude mať viac spoločných hrán s kostrou  $K$  ako kostra  $K_0$ . Po konečnom počte krokov dostaneme takto z kostry  $K_0$  postupnosť najlacnejších kostier

$$K_0, K^{(1)}, K^{(2)}, \dots, K^{(k)} = K,$$

takých, že

$$c(K_0) = c(K^{(1)}) = c(K^{(2)}) = \dots = c(K^{(k)}) = c(K),$$

– t. j.  $K$  je tiež najlacnejšia kostra. ■

**Algoritmus 4.3. Kruskalov algoritmus I.** na hľadanie najlacnejšej (najdrahšej) kostry súvislého hranovo ohodnoteného grafu  $G = (V, H, c)$ .

- **Krok 1.** Zoraď hrany podľa ich ohodnotenia vzostupne (zostupne) do postupnosti  $\mathcal{P}$ .
- **Krok 2.** Nech prvá hrana v postupnosti  $\mathcal{P}$  je hrana  $\{u, v\}$ . Vylúč hranu  $\{u, v\}$  z postupnosti  $\mathcal{P}$  a ak s už vybranými hranami nevytvára cyklus, zaraď ju do kostry.
- **Krok 3.** Ak je počet vybraných hrán rovný  $|V| - 1$  alebo ak je postupnosť  $\mathcal{P}$  prázdna, STOP. Inak GOTO Krok 2. ♣

*Poznámka.* Kruskalov algoritmus I. je formulovaný tak, že je použiteľný aj na nesúvislý hranovo ohodnotený graf  $G$ . V tom prípade nájde najlacnejšie, resp. najdrahšie kostry pre všetky komponenty grafu  $G$ . Túto istú vlastnosť bude mať aj Kruskalov algoritmus II.

Táto formulácia algoritmu je veľmi názorná, no programátorovi by sa mohlo právom zdať, že nedostatočne formuluje, akým spôsobom skontrolovať, či pridávaná hrana nevytvára s už vybranými hranami cyklus. Túto kontrolu si môžeme značne zjednodušiť, keď vrcholom pridáme značky vyjadrujúce, v ktorom komponente lesa tvoreného zo všetkých vrcholov a doteraz vybraných hrán príslušný vrchol leží. Ak má pridávaná hrana oba koncové vrcholy v tom istom komponente, jej pridanie by znamenalo vznik cyklu.

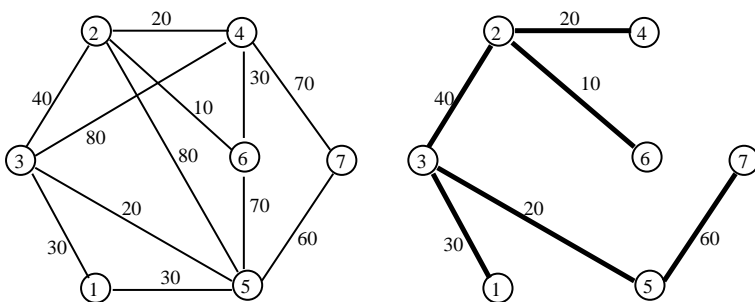
**Algoritmus 4.4. Kruskalov algoritmus II.** na hľadanie najlacnejšej (najdrahšej) kostry súvislého hranovo ohodnoteného grafu  $G = (V, H, c)$ .

- **Krok 1.** Zorad' hrany podľa ich ohodnotenia vzostupne (zostupne) do postupnosti  $\mathcal{P}$ .
- **Krok 2.** Pre každý vrchol  $i \in V$  polož  $k(i) = i$ .
- **Krok 3.** Nech prvá hrana v postupnosti  $\mathcal{P}$  je hrana  $\{u, v\}$ . Vylúč hrana  $\{u, v\}$  z postupnosti  $\mathcal{P}$ . Ak  $k(u) \neq k(v)$ , zaraď hrana  $\{u, v\}$  do kostry, a  $\forall i \in V$ , pre ktoré  $k(i) = k(v)$ , potom polož  $k(i) := k(u)$ .
- **Krok 4.** Ak je počet vybraných hrán rovný  $|V| - 1$  alebo ak je postupnosť  $\mathcal{P}$  prázdna, STOP. Inak GOTO krok 3.



Odhadneme zložitosť Kruskalovho algoritmu. Usporiadanie  $m$  hrán grafu podľa veľkosti vyžaduje  $O(m \cdot \log m)$  operácií. Krok 3 sa opakuje najviac  $m$ -krát. Ak sa prvá hrana postupnosti  $\mathcal{P}$  nezaraď do (budúcej) kostry, bola s ňou urobená len jedna operácia (porovnanie značiek jej koncov). Ak hrana bola zaraďená do kostry, čo nastane presne  $n - 1$  krát, bude treba preznačkováť značky vrcholov, čo vyžaduje  $O(n)$  operácií pri jednom zaradení,  $O(n \cdot (n - 1))$  operácií celkom. Zložitosť Kruskalovho algoritmu v tomto tvare odhadneme teda ako  $O(m \cdot \log m + n^2)$ . V literatúre sa uvádza, že použitím vhodných dátových štruktúr možno na celý Kruskalov algoritmus vystačiť s  $O(m \cdot \log n)$  operáciami.

**Príklad 4.1.** Nájdite najlacnejšiu kostru v hranovo ohodnotenom grafe, ktorého diagram je na obrázku 4.8 vľavo.



Obr. 4.8: Hranovo ohodnotený graf  $G$  a jeho najlacnejšia kostra.

Hrany grafu  $G$  usporiadame neklesajúco podľa ich ohodnotenia do nasledujúcej tabuľky:

{2,6}	{2,4}	{3,5}	{1,3}	{1,5}	{4,6}	{2,3}	{5,7}	{4,7}	{5,6}	{2,5}	{3,4}
10	20	20	30	30	30	40	60	70	70	80	80

Hrana do kostry	1	2	3	4	5	6	7
	$k(v)$						
-	1	2	3	4	5	6	7
{2,6}	1	2	3	4	5	2	7
{2,4}	1	2	3	2	5	2	7
{3,5}	1	2	3	2	3	2	7
{1,3}	1	2	1	2	1	2	7
{3,2}	1	1	1	1	1	1	7
{5,7}	1	1	1	1	1	1	1

Výpočet môžeme urobiť v tabuľke. Do prvého stĺpca tabuľky budeme zapisovať práve pridávanú hranu, ďalšie stĺpce tabuľky budú vyhradené pre značky vrcholov  $k(v)$ . Prvý riadok tabuľky zodpovedá inicializácii značiek  $k()$ . Po vložení prvej hrany {2, 6} do budúcej kostry sa zmení značka vrchola 6 na  $k(6) := k(2)$ , čo zapíšeme do druhého riadku tabuľky.

Postupne prechádzame hranami z tabuľky hrán a skúšame ich zaradiť do kostry, čo sa dá, ak konce pridávanej hrany majú rôzne značky  $k()$ . Ak hranu možno zaradiť do kostry, venujeme jej ďalší riadok tabuľky. Po zaradení hrany do kostry zmeníme značky  $k()$  podľa kroku 3. algoritmu 4.4 a ich zmenené hodnoty zapíšeme do riadku pridávanej hrany.

Prvá hrana, ktorú sme nemohli pridať do kostry je hrana {1, 5}, pretože v okamihu rozhodovania o jej zaradení bolo už  $k(1) = k(5) = 1$  (pozri riadok tabuľky prislúchajúci hrane {1, 3}).

Po skončení práce algoritmu sú hrany najlacnejšej kostry v prvom stĺpci tabuľky, jej diagram je na obrázku 4.8 vpravo.

Poznamenajme nakoniec, že v počítači stačí pre uloženie a spracovávanie vektora značiek  $k()$  jedno jednorozmerné  $n$ -prvkové pole. Aj pri ručnom výpočte by sme vystačili s jedným riadkom tabuľky, ak by sme hodnoty značiek  $k()$  prepisovali na mieste. V opisovanej tabuľke však možno sledovať postup výpočtu a vývoj značiek  $k()$ .

*Poznámka.* Kruskalov algoritmus II (algoritmus 4.4) môžeme s výhodou použiť na zistenie komponentov grafu. Ak totiž spustíme tento algoritmus na nesúvislý graf, po skončení práce algoritmu značky  $k(\cdot)$  vrcholov z  $V$  budú určovať komponenty nasledovne: Dva vrcholy  $u \in V$ ,  $v \in V$  sú v tom istom komponente grafu  $G$  práve vtedy, keď  $k(u) = k(v)$ . Pre zisťovanie komponentov grafu nie je potrebné usporiadať hrany podľa ich ohodnotenia a krok 1 algoritmu 4.4 možno preformulovať nasledovne: Zoraď hrany grafu  $G$  do postupnosti  $\mathcal{P}$  v ľubovoľnom poradí.

## 4.4 Cesta maximálnej priepustnosti

V tejto časti sa budeme zaoberať hranovo ohodnoteným grafmi  $G = (V, H, c)$ , v ktorých  $c(h) > 0$  znamená **priepustnosť** (kapacitu) hrany  $h$ . (Termíny kapacita a priepustnosť budeme považovať za synonymá, hoci v praxi cestného dopravného inžinierstva je medzi týmito pojmi istý rozdiel). Priepustnosť  $c(h)$  hrany  $h$  môže znamenať:

- maximálny počet vozidiel, ktoré prejdú hranou za jednotku času,
- únosnosť hrany – maximálnu hmotnosť vozidla, ktoré prejde hranou (napr. hrana predstavuje komunikáciu na ktorej je mostík s nosnosťou 5 ton, preto najťažšie vozidlo, ktoré smie prejsť hranou môže vážiť 5 ton a v tomto prípade  $c(h) = 5$  pre hranu, ktorá modeluje príslušnú komunikáciu),
- maximálnu výšku vozidla, ktoré prejde hranou
- maximálnu šírku vozidla, ktoré prejde hranou
- maximálny počet bitov za sekundu prenesiteľných úsekom počítačovej siete
- veľkosť maximálneho prúdu, ktorý môže tiecť segmentom elektrickej siete

Iste nájdete aj ďalšie praktické príklady.

V praxi priepustnosť cesty prechádzajúcej niekoľkými úsekmi je určená priepustnosťou najmenej priepustného úseku. Tým je motivovaná nasledujúca definícia.

**Definícia 4.8.** Nech  $G = (V, H, c)$  je hranovo ohodnotený graf, v ktorom cena hrany  $h \in H$   $c(h) > 0$  znamená jej priepustnosť. **Priepustnosť**  $c(\mu(u, v))$   $u-v$

cesty (sledu, polosledu, atď.)  $\mu(u, v)$  definujeme ako

$$c(\mu(u, v)) = \min\{c(h) \mid h \in \mu(u, v)\}.$$

**Veta 4.5.** *Nech  $K$  je najdrahšia kostra v súvislom hranovo ohodnotenom grafe  $G = (V, H, c)$ , nech  $\{u, v\} \in H$  je taká hrana grafu  $G$ , ktorá nepatrí k hranovej množine kostry  $K$ . Nech  $\mu(u, v)$  je (jediná)  $u$ - $v$  cesta v kostre  $K$ . Potom je priepustnosť cesty  $\mu(u, v)$  väčšia alebo rovná ako priepustnosť hrany  $\{u, v\}$ , t. j.*

$$c(\mu(u, v)) \geq c(u, v).$$

DŔKAZ.

Stačí dokázať, že pre ľubovoľnú hranu  $h \in \mu(u, v)$  je  $c(h) \geq c(u, v)$ . Dôkaz vykonáme sporom.

Nech pre nejakú hranu  $h \in \mu(u, v)$  platí  $c(h) < c(u, v)$ . Zreťazením ciest  $\mu(u, v)$  a  $u, \{u, v\}, v$  dostaneme cyklus. Pridaním hrany  $\{u, v\}$  k množine hrán kostry  $K$  a vylúčením hrany  $h$  z cesty  $\mu(u, v)$  a teda aj z hranovej množiny kostry  $K$  dostaneme graf  $K' = K \cup \{\{u, v\}\} - \{h\}$ , ktorý je súvislý a má  $|V| - 1$  hrán.  $K'$  je teda strom – kostra grafu  $G$ , ktorej cena je

$$c(K') = c(K) + \underbrace{c(u, v) - c(h)}_{>0} > c(K).$$

Za predpokladu, že existuje hrana  $h \in \mu(u, v)$  taká, že  $c(h) < c(u, v)$ , sme zostrojili kostru  $K'$ , ktorej cena je väčšia než cena kostry  $K$ , čo je v spore s maximalitou  $K$ . ■

**Veta 4.6.** *Nech  $K$  je najdrahšia kostra v súvislom hranovo ohodnotenom grafe  $G = (V, H, c)$ . Potom pre ľubovoľné dva vrcholy  $u, v \in V$  je (jediná)  $u$ - $v$  cesta v  $K$   $u$ - $v$  cestou maximálnej priepustnosti v  $G$ .*

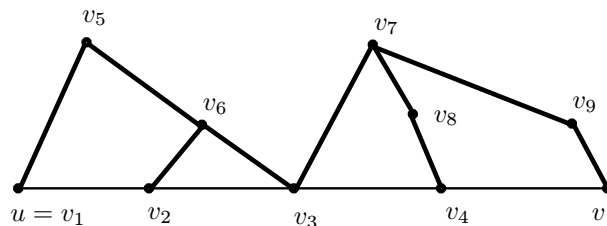
DŔKAZ.

Nech

$$M(u, v) = (v_1, \{v_1, v_2\}, v_2, \dots, \{v_{k-1}, v_k\}, v_k),$$

kde  $v_1 = u, v_k = v$  je  $u$ - $v$  cesta maximálnej priepustnosti. Nech pre  $i = 1, 2, \dots, k - 1$   $\mu(v_i, v_{i+1})$  je  $v_i$ - $v_{i+1}$  cesta po hranách najdrahšej kostry  $K$  – táto má podľa predchádzajúcej vety priepustnosť väčšiu alebo rovnú ako  $c(v_i, v_{i+1})$ . Zreťazením ciest  $\mu(v_i, v_{i+1})$  pre  $i = 1, 2, \dots, k - 1$  dostaneme  $u$ - $v$  sled  $S$  po hranách kostry  $K$ , ktorého priepustnosť je väčšia alebo rovná než priepustnosť cesty





Obr. 4.9: K dôkazu vety 4.6.

$$\begin{aligned}
 M(u, v) &= (u, \{u, v_2\}, v_2, \{v_2, v_3\}, v_3, \{v_3, v_4\}, v_4, \{v_4, v\}, v), \\
 \mu(u, v_2) &= (u, \{u, v_5\}, v_5, \{v_5, v_6\}, v_6, \{v_6, v_2\}, v_2), \\
 \mu(v_2, v_3) &= (v_2, \{v_2, v_6\}, v_6, \{v_6, v_3\}, v_3), \\
 \mu(v_3, v_4) &= (v_3, \{v_3, v_7\}, v_7, \{v_7, v_8\}, v_8, \{v_8, v_4\}, v_4), \\
 \mu(v_4, v) &= (v_4, \{v_4, v_8\}, v_8, \{v_8, v_7\}, v_7, \{v_7, v_9\}, v_9, \{v_9, v\}, v).
 \end{aligned}$$

Cesta max. priepustnosti po hranách kostry  
prechádza vrcholmi  $u, v_5, v_6, v_3, v_7, v_9, v$ .

$M(u, v)$ . Vynechaním viackrát prejedných hrán a vrcholov sledu  $S$  dostaneme  $u-v$  cestu po hranách kostry  $K$ , ktorej priepustnosť je väčšia alebo rovná priepustnosti cesty  $M(u, v)$ . Keďže cesta  $M(u, v)$  bola cestou maximálnej priepustnosti, je aj  $u-v$  cesta po hranách kostry  $K$  cestou maximálnej priepustnosti. ■

**Algoritmus 4.5. Algoritmus na hľadanie  $u-v$  cesty maximálnej priepustnosti v súvislom hranovo ohodnotenom grafe  $G = (V, H, c)$ .**

- **Krok 1.** V grafe  $G$  zostroj najdrahšiu kostru  $K$ .
- **Krok 2.** V kostre  $K$  nájdite (jedinú)  $u-v$  cestu. Táto (jediná)  $u-v$  cesta v kostre  $K$  je  $u-v$  cestou maximálnej priepustnosti v grafe  $G$ .



Uvedený algoritmus síce nájde  $u-v$  cestu maximálnej priepustnosti, no táto nemusí byť a spravidla ani nebýva optimálnou z hľadiska prejedenej vzdialenosti. Ak by sme chceli nájsť najkratšiu  $u-v$  cestu s maximálnou priepustnosťou, potrebujeme mať v príslušnom grafe okrem kapacitného ohodnotenia hrán aj ohodnotenie vyjadrujúce ich dĺžku.

**Algoritmus 4.6. Algoritmus na hľadanie najkratšej  $u-v$  cesty s maximálnou priepustnosťou v súvislom hranovo ohodnotenom grafe  $G = (V, H, c, d)$ , kde  $c(h)$  je priepustnosť a  $d(h)$  je dĺžka hrany  $h \in H$ .**

- **Krok 1.** V grafe  $G$  nájdí cestu  $\mu(u, v)$  maximálnej priepustnosti vzhľadom na ohodnotenie hrán  $c$ . Nech  $C$  je priepustnosť cesty  $\mu(u, v)$ .
- **Krok 2.** Vytvor graf  $G' = (V, H', d)$ , kde  $H' = \{h \mid h \in H, c(h) \geq C\}$ .  $\{H' \text{ obsahuje len tie hrany pôvodného grafu, ktoré majú priepustnosť väčšiu alebo rovnú než } C.\}$
- **Krok 3.** V grafe  $G'$  nájdí najkratšiu  $u-v$  cestu vzhľadom na ohodnotenie hrán  $d$ .



## 4.5 Aplikácie

### 4.5.1 Štruktúra adresárov na disku

Pevný disk počítača slúži na uloženie dát, ktoré sú usporiadané do súborov. Najmenšou logickou jednotkou, s ktorou diskový systém pracuje, je súbor. Sú médiá (napríklad magnetopáskové jednotky), ktoré súbory ukladajú do jednej postupnosti bez možnosti vytvárania nejakej ich štruktúry. Takáto jednoduchá organizácia dát súvisí často s tým, že takéto médium nemá priamy, ale iba sekvenčný prístup ku svojim údajom.

Pevný disk však má možnosť ľubovoľného prístupu k svojim dátam, čo sa dá využiť na lepšie usporiadanie súborov ako jednoduché sekvenčné usporiadanie. Na to aby sa súbory na disku dali lepšie organizovať sú súbory rozdelené do adresárov. Existuje hlavný adresár disku, ktorý nie je súčasťou žiadneho iného adresára. Každý adresár môže obsahovať súbory a ďalšie podadresáre.

Takáto disková štruktúra sa modeluje koreňovým stromom, v ktorom je koreňom hlavný adresár. Synmi každého adresára sú súbory ktoré obsahuje a jeho podadresáre. Otcom každého adresára (okrem hlavného adresára) je jeho nadradený adresár. Meno súboru alebo adresára môžeme interpretovať ako značku príslušného vrchola adresárového stromu. V rámci jedného adresára žiadne dva rôzne súbory alebo podadresáre nemôžu mať rovnakú značku. Úplná špecifikácia súboru alebo adresára na pevnom disku sa získa zrefazovaním všetkých značiek (jedinej) cesty z koreňa do príslušného vrchola.

Pretože v strome existuje jediná cesta z koreňa do ľubovoľného vrchola a pretože značky rôznych synov jedného vrchola adresárového stromu sú rôzne, úplné

špecifikácie rôznych súborov sú rôzne napriek tomu, že v odlišných adresároch môžu byť súbory s rovnakým menom.

Ak potrebujeme vyhľadať súbor daných vlastností v adresárovom strome, stačí nám upotrebiť niektorý z algoritmov na prehľadávanie grafov – napr. Tarryho algoritmus, ktorý pri nájdení hľadaného súboru môžeme predčasne ukončiť, pri hľadaní všetkých súborov daných vlastností ho však musíme nechať dobehnúť do konca. Efektívnejšie je však v tomto prípade prehľadávanie do hĺbky alebo do šírky.

### 4.5.2 Prefixové kódovanie

Kódovanie je zobrazenie, ktoré každému znaku abecedy  $A = \{a_1, a_2, \dots, a_n\}$  priradí slovo (t. j. konečnú postupnosť znakov) abecedy  $B = \{b_1, b_2, \dots, b_m\}$ . Slová abecedy  $B$ , ktoré sú obrazom nejakého znaku abecedy  $A$ , sa nazývajú **kódové slová** – je ich práve  $n$ , ostatné slová abecedy  $B$  sú **nekódové slová**. Množina kódových slov sa volá **kód**.

Účelom kódovania je prispôsobiť správu napísanú v abecede  $A$  možnostiam prenosového kanálu alebo archivačného média, skomprimovať správu, alebo vytvoriť také kódovanie, ktoré bude odolné proti chybám pri prenose. Čitateľ by sa tu mohol pozastaviť nad otázkou, prečo nepriraďujeme znakom abecedy  $A$  znaky, ale slová abecedy  $B$ . Je to preto, lebo obyčajne má abeceda  $B$  menej znakov ako abeceda  $A$ .

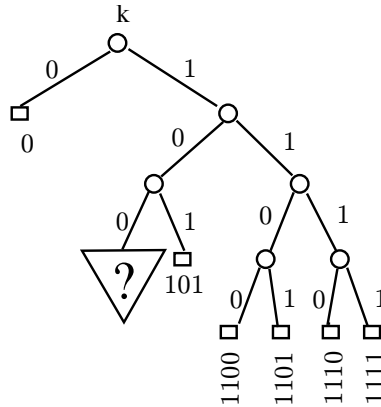
Abecedou  $B$  je najčastejšie dvojprvková množina  $B = \{0, 1\}$  – vtedy hovoríme o binárnom kódovaní. Príkladom binárneho kódovania je všeobecne známe priradenie 8-bitových binárnych čísel znakom latinskej abecedy, cifram 0 až 9 a ďalším špeciálnym znakom nazývané ASCII kód. Tu majú všetky kódové slová rovnakú dĺžku 8 znakov.

V teórii kódovania sa však používajú aj také kódovania, pri ktorých nemajú kódové slová rovnakú dĺžku. Medzi nimi majú význačné postavenie tzv. prefixové kódy. Kód je **prefixový**, keď žiadne kódové slovo nie je začiatkom iného kódového slova.

Každému prefixovému kódu prislúcha strom. Napríklad kódu

$$K = \{0, 101, 1100, 1101, 1110, 1111\}$$

zodpovedá strom na obrázku 4.10.



Obr. 4.10: Strom prefixového kódu.

Strom má koreň  $k$ , vrcholy stupňa 3 vyznačené kolieskami a vrcholy stupňa 1 vyznačené štvorčkami. Tieto zodpovedajú kódovým slovám. Na začiatku dekódovania nastavíme pomyselný ukazovateľ do vrchola  $k$ . Podľa prijatého znaku 0 alebo 1 ideme z aktuálneho vrchola hranou označenou prijatým znakom. Keď prídeme do koncového vrchola grafu, konštatujeme, že sme prijali kódové slovo. Nastavíme ukazovateľ do vrchola  $k$  a prijímame ďalšie znaky.

Jeden vrchol stromu stupňa 1 je označený trojuholníkom a nezodpovedá žiadnemu kódovému slovu. Pri bezchybnom prenose sa do takéhoto vrchola nemôžeme dostať. Vrchol označený trojuholníkom totiž prislúcha prijatému slovu 100, ktoré nie je kódovým slovom a nie je ani prefixom žiadneho iného kódového slova. Z toho vyplýva, že kód  $K$  nie je optimálny a že ho môžeme skrátiť, keď namiesto kódového slova 101 použijeme slovo 10. Dospejeme tak ku kratším zakódovaným správam.

Pri komprimácii dát a súborov sa využíva skutočnosť, že nie všetky znaky zdrojovej abecedy  $A$  sa vyskytujú s rovnakou relatívnou početnosťou. Túto nerovnomernosť môžeme využiť tak, že znaky vyskytujúce sa často zakódujeme kratším kódovým slovom, kým zriedkavým znakom pridelíme dlhšie kódové slová. Optimálne to robí konštrukcia, ktorú navrhol Huffman (čítaj hafmen) a výsledný kód sa volá Huffmanov kód.

**Algoritmus 4.7. Algoritmus na zostrojenie Huffmanovho kódu.**

Budeme postupne budovať binárny koreňový strom, ktorého všetky vrcholy stupňa 1 budú znaky zdrojovej abecedy  $A$ . Každý vrchol stromu bude mať priradenú pravdepodobnosť a každej hrane stromu priradíme binárny znak 0 alebo 1.

- **Krok 1.** Zostroj graf  $G = (V, H, p)$ , kde  $V = A$  a kde  $p(v)$  je pravdepodobnosť znaku  $v$ . Inicializačne polož  $H = \emptyset$ . Všetky vrcholy z  $V$  inicializačne prehlás za neoznačené.
- **Krok 2.** Nájdi dva neoznačené vrcholy  $u, w$  z množiny  $V$  s najmenšími pravdepodobnosťami  $p(u), p(w)$ . Označuj vrcholy  $u, v$  ako spracované. Množinu vrcholov  $V$  rozšír o vrchol  $x$ , t. j. polož  $V := V \cup \{x\}$  pre nejaké  $x \notin V$ , polož  $p(x) := p(u) + p(w)$ ,  $H := H \cup \{\{x, u\}, \{x, w\}\}$  a nový vrchol  $x$  prehlás za neoznačený. Hrane  $\{x, u\}$  priradi značku 0, hrane  $\{x, w\}$  značku 1.
- **Krok 3.** Ak je graf  $G$  súvislý, prehlás posledne pridaný vrchol  $x$  za koreň  $k$  a GOTO Krok 4. Inak GOTO Krok 2.
- **Krok 4.** { Teraz je graf  $G$  stromom so všetkými vrcholmi stupňa 1 zodpovedajúcimi znakom zdrojovej abecedy  $A$ . Všetky hrany stromu  $G$  sú označené binárnymi značkami 0 alebo 1. }

Z koreňa stromu do každého koncového vrchola vedie jediná cesta, binárne značky hrán na tejto ceste určujú prefixový kód každého znaku abecedy  $A$ .

**4.5.3 Riešenie zložitých elektrických obvodov**

Majme elektrický obvod zložený z odporov a napäťových zdrojov (zvaných tiež dvojpóly). Spoločná svorka dvoch alebo viacerých dvojpólov sa nazýva uzol, počet dvojpólov zapojených na uzol si nazveme stupeň uzla. Vetva je sériové spojenie niekoľkých dvojpólov medzi dvoma uzlami, ktoré majú aspoň stupeň 3. Vyriešiť problém daného elektrického obvodu znamená určiť pre každú vetvu prúd, ktorý ňou tečie.

Ku každému elektrickému obvodu  $\mathcal{O}$ , ktorý má aspoň dva uzly vyššieho stupňa než 2, si môžeme zostrojiť graf  $G_{\mathcal{O}} = (V, H)$ , ktorého vrcholmi budú všetky uzly príslušného obvodu so stupňom väčším ako 2 a ktorého hranami budú všetky vetvy obvodu  $\mathcal{O}$ . Teraz už môžeme definovať slučku obvodu  $\mathcal{O}$  ako ľubovoľný cyklus v grafe  $G_{\mathcal{O}}$ .

Pri hľadaní prúdov vetvami používame tri fyzikálne zákony – Ohmov zákon, ktorý hovorí, že ak odporom veľkosti  $R$  tečie prúd  $I$ , potom na ňom vznikne úbytok napätia  $U = R \cdot I$ , prvý Kirchhoffov zákon hovoriaci, že algebraický súčet prúdov v uzle sa rovná nule a druhý Kirchhoffov zákon, ktorý hovorí, že algebraický súčet napätí v ľubovoľnej slučke sa rovná nule. Napätia v slučke sú dvojakého typu – známe napätia zdrojov a napätia objavujúce sa na odporoch, ktoré vypočítame z prúdov tečúcich odporami a známých veľkostiach odporov podľa Ohmovho zákona. Druhý Kirchhoffov zákon pre ľubovoľnú slučku  $S$  má tvar

$$\sum_{\substack{\text{zdroj } U_i \\ \text{leží na slučke } S}} U_i + \sum_{\substack{\text{odpor } R_j \\ \text{leží na slučke } S}} I_{R_j} \cdot R_j = 0, \quad (4.4)$$

kde  $I_{R_j}$  je prúd tečúci odporom  $R_j$ . Poznamenajme, že pre všetky odpory  $R_j$  v  $k$ -tej vetve je  $I_{R_j}$  rovné vetvovému prúdu  $I_k$ .

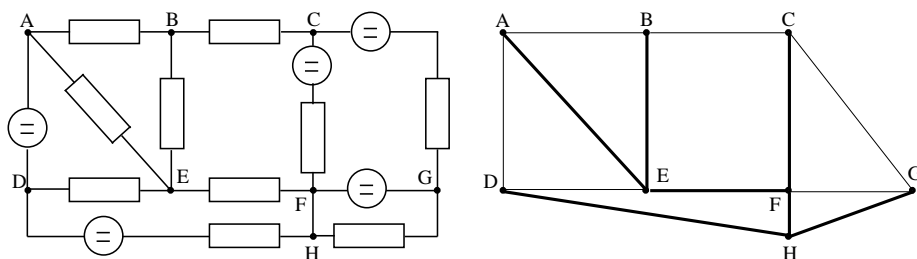
Nech  $n$  je počet uzlov a  $m$  počet vetiev obvodu  $\mathcal{O}$ . Označme  $I_1, I_2, \dots, I_m$  prúdy tečúce vetvami  $h_1, h_2, \dots, h_m$  obvodu  $\mathcal{O}$ . Rovnice pre prvý Kirchhoffov zákon zostavíme jednoducho

$$\sum_{\{j|h_j \in H_u\}} I_j = 0 \quad \forall \text{ uzol } u = 1, 2, \dots, n, \quad (4.5)$$

kde  $H_u$  je množina vetví incidentných s uzlom  $u$ .

Rovnic typu (4.5) je  $n$ , ale len  $n - 1$  z nich je lineárne nezávislých. Keďže  $m > n$ , musíme ich doplniť  $m - n + 1$  rovnicami typu (4.4) vyjadrujúcimi druhý Kirchhoffov zákon. Počet slučiek v obvode  $\mathcal{O}$  môže byť veľmi veľký, vybrať ľubovoľných  $(m - n + 1)$  z nich nie je dobrá myšlienka, lebo príslušné rovnice by mohli byť lineárne závislé a riešenie výslednej sústavy by v takom prípade nebolo jednoznačne určené. Stojíme teda pred problémom, ako nájsť  $m - n + 1$  lineárne nezávislých slučiek obvodu  $\mathcal{O}$ .

Zostrojíme ľubovoľnú kosť  $K = (V, H_K)$  v grafe  $G$ . Táto kosť má podľa vety 4.2  $n - 1$  hrán. Ostalo teda  $m - (n - 1) = m - n + 1$  hrán, ktoré neležia v kostre  $K$ . Ak ku hranám kostry  $K$  pridáme ľubovoľnú hranu  $h \in (H - H_K)$ , dostaneme graf  $K \cup \{h\}$ , v ktorom existuje práve jeden cyklus definujúci slučku  $S_h$  v obvode  $\mathcal{O}$ . Dá sa ukázať, že systém  $m - n + 1$  slučiek  $\{S_h | h \in (H - H_K)\}$  je systém lineárne nezávislých slučiek v tom zmysle, že systém príslušných lineárnych rovníc vyjadrujúcich druhý Kirchhoffov zákon je lineárne nezávislý. Na obrázku 4.11 vidíme obvod  $\mathcal{O}$  vľavo, vpravo k nemu príslušný graf  $G_{\mathcal{O}}$ , hrany jeho kostry sú vyznačené hrubými čiarami. Slučkové rovnice pre slučky  $(A, B, E, A)$ ,  $(A, E, F, H, D, A)$ ,  $(B, C, F, E, B)$ ,  $(C, G, H, F, C)$ ,  $(D, E, F, H, D)$



Obr. 4.11: Elektrický obvod  $\mathcal{O}$ , k nemu príslušný graf  $G_{\mathcal{O}}$  a jeho kostra  $K$ .

a  $(F, G, H, F)$  budú lineárne nezávislé a možno ich použiť pre riešenie príslušného obvodu  $\mathcal{O}$ .

Podobne sa dá využiť kostra grafu  $G_{\mathcal{O}}$  na identifikáciu nezávislých slučiek pri riešení obvodov so striedavým prúdom a so všeobecnými impedanciami (odporní, kapacitami a indukčnosťami).

Na tomto príklade vidíme dva spôsoby aplikácie matematiky (v tomto prípade teórie grafov) v iných odboroch (tu konkrétne v elektrotechnike). Prvý z nich je ten, že pomáha presne definovať pojmy – videli sme to na prípade definície slučky v elektrickom obvode. Druhý spôsob je, že dáva exaktný nástroj na riešenie konkrétneho problému – tu zostavenia sústavy lineárne nezávislých rovníc opisujúcich elektrický obvod.

#### 4.5.4 Úloha o elektrifikácii

Do  $n$  miest v istom regióne máme zaviesť elektrický prúd. Možné spôsoby vybudovania vedení medzi jednotlivými miestami možno modelovať hranovo ohodnoteným grafom  $G = (V, H, c)$ , v ktorom je  $V$  množinou všetkých miest, množinu hrán  $H$  tvoria všetky neusporiadané dvojice miest  $\{u, v\}$  také, že medzi miestami  $u$  a  $v$  možno viesť priame elektrické vedenie a cena hrany  $c(\{u, v\}) > 0$  predstavuje náklady na vybudovanie priameho vedenia medzi miestami  $u$  a  $v$ . Chceme vybudovať elektrickú sieť s najmenšími nákladmi avšak tak, aby v každom mieste bola zavedená elektrina.

Na riešenie vyššie spomínanej úlohy stačí určiť, medzi ktorými vrcholmi grafu  $G$  postavíme elektrické vedenie, čo je to isté ako keď určíme, ktoré hrany grafu  $G$  vyberieme do riešenia. Nech  $G_1$  je taký faktorový podgraf grafu  $G$ ,

t. j. podgraf obsahujúci všetky vrcholy grafu  $G$ , ktorý obsahuje práve všetky hrany vybrané pre stavbu vedenia. Ak má byť elektrina zavedená do všetkých miest, graf  $G_1$  musí byť súvislý. Úloha o elektrifikácii sa dá teda naformulovať ako hľadanie najlacnejšieho súvislého faktorového podgrafu grafu  $G$  (najlacnejší v zmysle súčtu ocenení všetkých jeho hrán).

Ak však má byť  $G_1$  najlacnejší súvislý faktorový podgraf, nesmie obsahovať cyklus – ináč by sme vylúčením ľubovoľnej hrany takéhoto cyklu dostali lacnejší súvislý faktorový podgraf grafu  $G$ . Preto  $G_1$  musí byť kostrou grafu  $G$ . Úloha o elektrifikácii sa teda dá formulovať ako hľadanie najlacnejšej kostry v súvislom hranovo ohodnotenom grafe  $G$ .

Tu popísaný princíp možno bez zmeny použiť aj pri plánovaní telekomunikačných, dátových a počítačových sietí.

#### 4.5.5 Plánovanie prepravy nadrozmerných nákladov

Existujú dopravné podniky, ktoré sa špecializujú na cestnú prepravu mimo-riadne veľkých nákladov ako sú riečne lode, veľké chemické reaktory, žeriavy a iné veľké konštrukcie, ktorých veľkosť, váha, výška alebo dĺžka presahujú hodnoty bežne prepravovaných predmetov.<sup>1</sup> Ako technické prostriedky sa na takúto prepravu používajú veľké ťahače, mnohonápravové prívesy, dlhé predmety sa prepravujú aj na dvoch prívesoch ovládaných vpredu i vzadu. Prepravovaný náklad sprevádza konvoj doprovdných vozidiel pripravených vyriešiť na mieste menšie problémy (ako odstrániť a znovu na pôvodné miesto privariť prekážajúci kovový stĺpik plotu, zábradlie či dopravnú značku) i náročnejšie úlohy (zosilnenie mostíka, či dokonca nadvihnutie železničného nadjazdu nad cestou).

Každá takáto mimoriadna preprava sa musí dopredu veľmi starostlivo naplá-novať. Existujú počítačové modely cestnej siete, ktoré pre každý cestný úsek obsahujú jeho šírku, typ povrchu vozovky, únosnosť prípadných mostov na úseku, polomer zatáčania, svetlú výšku prípadných podjazdov, ktorými úsek prechádza, typ krajnice a mnoho ďalších údajov. Algoritmus pre hľadanie cesty maximálnej priepustnosti tu dáva návod, ako nájsť vhodnú cestu pri zohľadnení všetkých možných požiadaviek.

Bez ohľadu na to, či bola plánovaná trasa nadmerného nákladu určená ručne alebo počítačom, krátko pred plánovanou prepravou skupina odborníkov prejde

---

<sup>1</sup>Autor získal uvádzané skúsenosti pri spolupráci s podnikom Závod ťažkej dopravy, ČSAD, n.p. Ostrava



plánovanou trasou a pozorne skontroluje, či sa na nej niečo oproti evidovanému stavu nezmenilo a či je pre plánovanú prepravu skutočne vhodná.

## 4.6 Cvičenia

1. Iným spôsobom ako vo vete 4.2 dokážte, že v strome  $G = (V, H)$  je  $|H| = |V| - 1$ . Postupujte matematickou indukciou s využitím tvrdenia vety 4.1, podľa ktorého v každom strome s aspoň dvoma vrcholmi existuje vrchol stupňa 1.
2. Implikácia  $c \rightarrow d$  z vety 4.2 sa dá dokázať aj s využitím nasledujúcej úvahy: Keďže je v  $G$  každá hrana mostom, po odobratí ľubovoľnej hrany  $h \in H$  bude mať graf  $G - \{h\}$  o jeden komponent viac ako graf  $G$ . Nech  $H = \{h_1, h_2, \dots, h_m\}$ , vytvoríme postupnosť grafov  $G_1 = G - \{h_1\}$ ,  $G_2 = G - \{h_1, h_2\}, \dots, G_m = G - \{h_1, h_2, \dots, h_m\}$ , v ktorej má každý graf o jeden komponent viac ako jeho bezprostredný predchodca. Z počtu komponentov grafu  $G_m$  usúďte, čomu sa rovná  $m$ .
3. Po skončení práce ľubovoľného algoritmu na hľadanie najkratšej cesty z vrchola  $u$  do ostatných vrcholov súvislého hranovo ohodnoteného grafu  $G = (V, H, c)$  dostávame v značkách  $x(v)$  vrcholov grafu  $G$  predposledný vrchol najkratšej  $u-v$  cesty. Priradením  $x : V - \{u\} \rightarrow V$  vlastne definujeme pre každý vrchol  $v \in V$ ,  $v \neq u$  predchodcu vrchola  $v$  v nejakom koreňovom strome  $S$  s koreňom  $u$ . Strom  $S$  je faktorovým podgrafom grafu  $G$  – je kostrou grafu  $G$ . Je  $S$  najlacnejšou kostrou grafu  $G$ ?
4. Nech  $K$  je najlacnejšia kostra súvislého hranovo ohodnoteného grafu  $G = (V, H, c)$ . Pre každé  $u, v \in V$  je kostrou  $K$  jednoznačne určená  $u-v$  cesta  $m_K(u, v)$  obsahujúca len hrany kostry  $K$ . Je  $m_K(u, v)$  najlacnejšou cestou v grafe  $G$ ?
5. Prispôbte Kruskalov algoritmus na hľadanie komponentov ľubovoľného grafu  $G$ . Porovnajte tento spôsob hľadania komponentov so spôsobom založeným na Tarryho algoritme.
6. Nech  $\vec{G} = (V, H, c)$  je hranovo ohodnotený digraf, kde  $c(h) > 0$  je priepustnosť hrany  $h \in H$ . Tu nemožno použiť algoritmus na hľadanie cesty maximálnej priepustnosti založený na najdrahšej kostre. Navrhnite značkovací algoritmus pre hľadanie  $u-v$  cesty maximálnej priepustnosti

podobný základnému algoritmu na hľadanie najkratšej  $u-v$  cesty. Značka  $t(v)$  vrchola  $v$  nech značí priepustnosť doteraz nájdenej najpriepustnejšej  $u-v$  cesty a  $x(v)$  predposledný vrchol tejto cesty. Odhadnite zložitosť navrhnutého algoritmu.

### Počítačové cvičenia

7. Naprogramujte Kruskalov algoritmus. Aká reprezentácia grafu je preň najvhodnejšia?
8. Na základe Kruskalovho algoritmu napíšte program na generovanie náhodných stromov so zadaným počtom vrcholov  $n$ . Návod: Začnite lesom obsahujúcim jednu náhodne vybratú hranu. Náhodne vygenerujte ďalšiu hranu a pokiaľ nevytvára s existujúcim lesom cyklus, dodajte ju k lesu.
9. Napíšte program pre generovanie náhodných súvislých grafov so zadaným počtom vrcholov  $n$  a hrán  $m$ ,  $m \geq n - 1$ .
10. Na základe Kruskalovho algoritmu napíšte program, ktorý zistí všetky komponenty zadaného grafu.

# Kapitola 5

## Acyklické digrafy

### 5.1 Vlastnosti acyklických digrafov

V predchádzajúcej kapitole sme sa zaoberali acyklickými súvislými grafmi. Zistili sme, že tieto grafy majú pomerne jednoduchú štruktúru, ktorú charakterizujú prvé dve vety predchádzajúcej kapitoly. Vďaka tejto štruktúre sú mnohé úlohy v stromoch ľahko riešiteľné, dokonca aj také, ktoré sú vo všeobecných grafoch ťažké. Napríklad: pretože v strome existuje pre  $u \neq v$  jediná  $u$ - $v$  cesta, nájsť najdlhšiu  $u$ - $v$  cestu v strome je úloha jednoduchá.

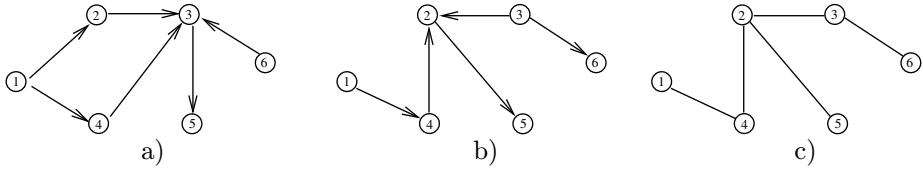
Analógiou acyklických grafov sú v orientovanom prípade acyklické digrafy, analógiou stromov orientované stromy.

**Definícia 5.1.** **Acyklický digraf** je taký digraf, ktorý neobsahuje cyklus. **Orientovaný strom** je neorientovane súvislý digraf, ktorý neobsahuje polocyklus.

Ak  $\vec{G} = (V, H)$  je acyklický digraf, potom nemôže obsahovať hrany  $(u, v)$  a  $(v, u)$  súčasne, lebo by v tomto prípade obsahoval i cyklus  $(u, (u, v), v, (v, u), u)$ .<sup>1</sup>

---

<sup>1</sup>Tu by si niekto mohol položiť otázku, či  $(u, (u, v), v, (v, u), u)$  nie je polocyklus, čo však nie je, lebo obsahuje tú istú orientovanú hranu  $(u, v)$  dvakrát.



Obr. 5.1: a) Neorientovane súvislý acyklický digraf, ktorý nie je orientovaným stromom.  
 b) Orientovaný strom  $G$ . c) Neorientovaný strom príslušný k  $\vec{G}$ .

Ku každému acyklickému digrafu  $\vec{G} = (V, H)$  možno zostrojiť graf  $G' = (V, H')$  s tou istou množinou vrcholov  $V$  a s množinou hrán  $H'$  definovanou

$$H' = \{\{u, v\} \mid (u, v) \in H\}. \quad (5.1)$$

$H'$  je vlastne množina hrán  $H$ , v ktorej „zabudneme“ na orientáciu.<sup>2</sup> Pretože acyklický digraf môže pre každú dvojicu vrcholov  $u \in V$ ,  $v \in V$ , kde  $u \neq v$ , obsahovať najviac jednu z orientovaných hrán  $(u, v)$ ,  $(v, u)$ , platí

$$|H'| = |H|.$$

K ľubovoľnej  $u-v$  polocene v digrafe  $\vec{G}$  je jednoznačne priradená  $u-v$  cesta v grafe  $G'$  prechádzajúca tými istými vrcholmi. Podobne ľubovoľnému polocyklu v digrafe  $\vec{G}$  je jednoznačne priradený cyklus v grafe  $G'$  určený tou istou postupnosťou vrcholov.

Z neorientovanej súvislosti digrafu  $\vec{G}$  vyplýva teda súvislosť grafu  $G'$ . Z acykličnosti digrafu  $\vec{G}$  však nevyplýva acykličnosť príslušného grafu  $G'$  – pozri obrázok 5.1 a). Ak však v digrafe  $\vec{G}$  neexistuje polocyklus, potom je k nemu príslušný graf  $G'$  acyklický. Ak je digraf  $\vec{G}$  orientovaným stromom, potom je graf  $G'$  stromom a platí preň veta 4.2 predchádzajúcej kapitoly, ktorá sa dá pre digraf  $\vec{G}$  preformulovať do nasledujúcej vety.

<sup>2</sup>Graf  $G' = (V, H')$  s množinou hrán  $H'$  definovanou vzťahom (5.1) možno zostrojiť pre ľubovoľný digraf  $\vec{G} = (V, H)$ , avšak vzťah  $|H| = |H'|$ , ktorý budeme ďalej využívať, platí len vtedy, ak v digrafe  $\vec{G}$  pre ľubovoľné  $u \in V$ ,  $v \in V$  existuje najviac jedna z hrán  $(u, v)$ ,  $(v, u)$ , čo je v acyklickom digrafe zaručené.

**Veta 5.1.** *Nasledujúce tvrdenia sú ekvivalentné:*

- a) Digraf  $\vec{G} = (V, H)$  je orientovaný strom.
- b) V digrafe  $\vec{G} = (V, H)$  existuje pre každé  $u, v \in V$  jediná  $u$ - $v$  plocesta.
- c) Digraf  $\vec{G} = (V, H)$  je neorientovane súvislý a každá orientovaná hrana množiny  $H$  je mostom. (Mostom v orientovanom digrafe rozumieme takú orientovanú hranu, po vybratí ktorej stúpne počet komponentov digrafu).
- d) Digraf  $\vec{G} = (V, H)$  je neorientovane súvislý a  $|H| = |V| - 1$ .
- e) V digrafe  $\vec{G} = (V, H)$  platí  $|H| = |V| - 1$  a  $G$  neobsahuje polocyklus.

V stromoch s aspoň dvoma vrcholmi existujú aspoň dva vrcholy stupňa 1. Analógiou tejto vety je nasledujúce tvrdenie.

**Veta 5.2.** *Nech  $\vec{G} = (V, H)$  je acyklický digraf. Potom  $V$  obsahuje aspoň jeden vrchol  $z$  taký, že  $\text{iddeg}(z) = 0$  a aspoň jeden vrchol  $u$  taký, že  $\text{odeg}(u) = 0$ .*

DÔKAZ.

Nech

$$v_1, (v_1, v_2), v_2, \dots, (v_{k-1}, v_k), v_k \quad (5.2)$$

je orientovaná cesta v digrafe  $\vec{G}$  s najväčším počtom hrán. Ukážeme, že  $\text{odeg}(v_k) = 0$ .

Keby  $\text{odeg}(v_k) > 0$ , musela by existovať orientovaná hrana  $(v_k, v_l)$ , kde  $v_l \neq v_k$ . Keby existovalo  $v_i = v_l$  pre  $i = 1, 2, \dots, v_{k-1}$ , mohli by sme vytvoriť orientovaný cyklus

$$v_i, (v_i, v_{i+1}), \dots, (v_{k-1}, v_k), v_k, (v_k, v_l), v_l = v_i,$$

čo je v spore s acykličnosťou digrafu  $\vec{G}$ . Je teda  $v_l \neq v_i$  pre všetky  $i = 1, 2, \dots, k$ . Teraz už môžeme zostrojiť orientovanú cestu

$$v_1, (v_1, v_2), v_2, \dots, (v_{k-1}, v_k), v_k, (v_k, v_l), v_l,$$

ktorá obsahuje o jednu hranu viac ako pôvodná orientovaná cesta, čo je v spore s predpokladom, že (5.2) je cesta s najväčším počtom hrán.

Z predpokladu  $\text{odeg}(v_k) > 0$  sme dostali spor. Je teda  $\text{odeg}(v_k) = 0$ .

Analogicky sa ukáže, že  $\text{ideg}(z) = 0$ . ■

Podľa definície 3.6 na str. 62 vrchol  $v$  je dosiahnuteľný z vrchola  $u$  v digrafe  $\vec{G}$ , ak v digrafe  $\vec{G}$  existuje orientovaná  $u$ - $v$  cesta.

**Definícia 5.2.** Nech  $\vec{G} = (V, H)$  je neorientovane súvislý digraf. Nech  $z \in V$  je taký jeho vrchol, z ktorého sú dosiahnuteľné všetky vrcholy digrafu  $\vec{G}$ . Potom hovoríme, že vrchol  $z$  je **prameň**. Ďalej nech  $u \in V$  je taký vrchol digrafu  $\vec{G}$ , ktorý je dosiahnuteľný zo všetkých ostatných vrcholov digrafu  $\vec{G}$ . Potom hovoríme, že vrchol  $u$  je **stok**<sup>3</sup>.

*Poznámka.* Keďže pripúšťame triviálny  $v$ - $v$  sled, každý vrchol  $v$  je dosiahnuteľný zo samého seba.

**Definícia 5.3.** Nech  $\vec{G} = (V, H)$  je orientovaný strom. Ak v strome  $\vec{G}$  existuje prameň  $z$ , potom hovoríme, že  $G$  je **koreňový strom** a vrchol  $z$  je koreň stromu  $\vec{G}$ . **Binárny koreňový strom** je koreňový strom, v ktorom má každý vrchol najviac dvoch bezprostredných následníkov.

**Veta 5.3.** Ak acyklický digraf  $\vec{G} = (V, H)$  obsahuje prameň  $z$ , potom  $\text{ideg}(z) = 0$ . Ak  $\vec{G}$  obsahuje stok  $u$ , potom  $\text{odeg}(u) = 0$ .

DŮKAZ.

Nech  $\text{ideg}(z) > 0$ , potom existuje aspoň jedna orientovaná hrana typu  $(x, z)$ . Pretože  $x$  je dosiahnuteľný zo  $z$ , existuje orientovaná cesta

$$(z = v_1, (v_1, v_2), v_2, \dots, v_k, (v_{k-1}, v_k), v_k = x) .$$

Zreťazenie tejto cesty s cestou  $(x, (x, z), z)$  dáva orientovaný cyklus, čo je v spore s acykličnosťou digrafu  $\vec{G}$ .

Analogicky sa dôkaz vykoná aj pre stok  $u$ . ■

**Veta 5.4.** Digraf  $\vec{G} = (V, H)$  je acyklický práve vtedy, keď jeho vrcholy možno usporiadať do postupnosti

$$v_1, v_2, \dots, v_n \tag{5.3}$$

(t.j. prečíslovať) tak, že platí:

$$\text{Ak } (v_i, v_k) \in H \text{ potom } i < k. \tag{5.4}$$

---

<sup>3</sup>Upozorňujem čitateľa, že používanie termínov „zdroj“, „prameň“, „ústie“ a „stok“ je v literatúre značne nejednotné, a teda sa môže líšiť od definície 5.2.

DŔKAZ.

Nech sŔ vrcholy množiny  $V$  usporiadané do postupnosti (5.3) tak, že platí (5.4).  
Nech existuje cyklus

$$v_{k_1}, (v_{k_1}, v_{k_2}), v_{k_2}, \dots, v_{k_{r-1}}, (v_{k_{r-1}}, v_{k_r}), v_{k_r}, \quad \text{kde } v_{k_1} = v_{k_r}.$$

Potom z (5.4) vyplŔva  $k_1 < k_2, \dots, < k_r = k_1$ , čiže  $k_1 < k_1$ , čo je spor.

Nech  $\vec{G} = (V, H)$  je acyklický digraf. Chceme dokázať, že jeho vrcholy sa dajú usporiadať do postupnosti tvaru (5.3) tak, aby platilo (5.4). Budeme postupovať matematickou indukciou podľa počtu vrcholov  $n = |V|$ .

Ak  $|V| = 1$ , potom  $V = \{v\}$  a postupnosť (5.3) je  $v = v_1$ .

Nech tvrdenie platí pre všetky acyklické digrafy s  $|V| = n$ , majme acyklický digraf  $\vec{G} = (V, H)$ , kde  $|V| = n + 1$ . V digrafe  $\vec{G}$  existuje aspoň jeden vrchol, ktorý označíme  $v_{n+1}$  taký, že  $\text{odeg}(v_{n+1}) = 0$ . Zostrojme digraf  $\vec{G}' = (V', H')$ , v ktorom  $V' = V - \{v_{n+1}\}$  a  $H' = H - H^-(v_{n+1})$ , kde  $H^-(v_{n+1})$  je množina orientovaných hrán vchádzajúcich do vrchola  $v_{n+1}$ .

Digraf  $\vec{G}'$  je acyklický. Keďže  $|V'| = n$ , platí pre  $\vec{G}'$  indukčný predpoklad, t. j. jeho vrcholy možno usporiadať do postupnosti

$$v_1, v_2, \dots, v_n$$

tak, že platí (5.4). Postupnosť

$$v_1, v_2, \dots, v_n, v_{n+1}$$

je hľadanou postupnosťou pre digraf  $\vec{G}$ , pre ktorú platí (5.4). ■

**Definícia 5.4.** Očíslovanie vrcholov  $v_1, v_2, \dots, v_n$  acyklického digrafu  $\vec{G} = (V, H)$ , pre ktoré platí:

$$\text{ak } (v_i, v_k) \in H, \text{ potom } i < k,$$

nazveme **monotónnym očíslovaním** vrcholov acyklického digrafu.

**Algoritmus 5.1. Algoritmus I. na monotónne očíslovanie acyklického digrafu  $\vec{G} = (V, H)$ .**

- **Krok 1.** Polož  $i := 1$ .
- **Krok 2.** {Digraf  $\vec{G} = (V, H)$  obsahuje aspoň jeden taký vrchol  $v \in V$ , že  $\text{ideg}(v) = 0$ .}  
Vezmi taký vrchol  $v \in V$ , že  $\text{ideg}(v) = 0$  a polož  $v_i := v$ .
- **Krok 3.** Ak  $V - \{v\} = \emptyset$  STOP,  
inak  $\vec{G} := \vec{G} - \{v\}$ ,  $i := i + 1$  a Goto Krok 2.



Práve prezentovaný algoritmus vykoná  $n$ -krát kroky 2. a 3. Pri každom kroku 2. musí prezrieť najprv  $n$ , potom  $n - 1$ ,  $n - 2$  atď. vrcholov, aby našiel vrchol stupňa 0, v kroku 3. musí upraviť najprv maximálne  $n - 1$  potom  $n - 2$  ... stupňov vrcholov digrafu  $\vec{G}$ . Algoritmus má teda zložitosť  $O(n^2)$ . Existujú aj lepšie implementácie, najmä pre digrafy, v ktorých je počet hrán relatívne malý. Ako príklad môže slúžiť nasledujúca verzia algoritmu.

**Algoritmus 5.2. Algoritmus II. na monotónne očíslovanie vrcholov acyklického digrafu  $\vec{G} = (V, H)$ .**

- **Krok 1.** Pre každé  $v \in V$  priradi značku  $d(v) := \text{ideg}(v)$ . Urči podmnožinu  $V_0$  vrcholovej množiny  $V$  s nulovou značkou  $d$ , t. j.

$$V_0 = \{v \mid v \in V, d(v) = 0\}.$$

Polož  $k := |V_0|$  a prvky z množiny  $V_0$  zoradi do ľubovoľnej postupnosti  $\mathcal{P} = v_1, v_2, \dots, v_k$ . Polož  $i := 1$ .

- **Krok 2.** Postupne pre každý vrchol  $w$  výstupnej hviezdy vrchola  $v_i$  taký, že  $w \neq v_i$ , urob:  
 $d(w) := d(w) - 1$ . Ak  $d(w) = 0$  potom zaradi vrchol  $w$  na koniec postupnosti  $\mathcal{P}$ , t. j. polož  $k := k + 1$ ,  $v_k := w$ .
- **Krok 3.** Ak  $k = n = |V|$  STOP, inak polož  $i := i + 1$  a GOTO Krok 2.



Pri uložení digrafu vo forme výstupných hviezd jeho vrcholov výpočet značiek  $d(v)$  vyžaduje prekontrolovať  $m = |H|$  hrán, pre výber množiny  $V_0$  a stanovenie postupnosti  $v_1, v_2, \dots, v_k$  treba prekontrolovať  $n = |V|$  vrcholov. Krok 1.

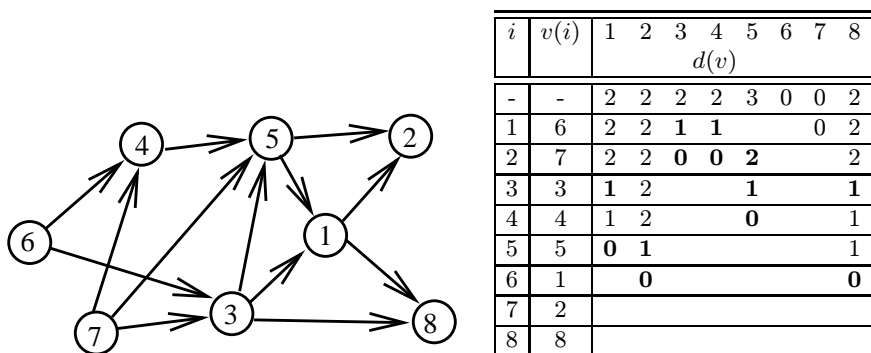


má teda zložitosť  $O(m + n)$ .

Krok 2. pre každý vrchol  $v_i$  zmení značky všetkých vrcholov jeho výstupnej hviezdy – má teda  $\text{odeg}(v_i)$  operácií. Počet všetkých operácií vykonaných v kroku 2. je  $\sum_{v \in V} \text{odeg}(v) = |H| = m$ . V kroku 3. sa vykoná  $n$  operácií, takže v oboch krokoch sa vykoná  $O(n + m)$  operácií.

Posledne prezentovaný algoritmus má zložitosť  $O(m + n)$ .

**Príklad 5.1.** Nájdite monotónne očíslovanie digrafu  $\vec{G} = (V, H)$  s diagramom na obrázku 5.2.



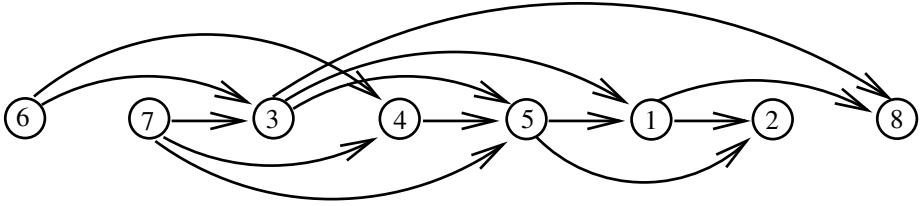
Obr. 5.2: Acyklický digraf a tabuľka s výpočtom monotónneho usporiadania.

Výpočet podľa algoritmu 5.2 urobíme v tabuľke (pozri obrázok 5.2), v ktorej v prvom stĺpci budú začínajúc druhým riadkom postupne čísla  $i = 1, 2, \dots, 8$ , v druhom stĺpci bude po skončení výpočtu poradie vrcholov digrafu  $\vec{G}$  v hľadanom monotónnom usporiadaní. V ďalších  $n = |V|$  stĺpcoch budú značky  $d(v)$  vrcholov z  $V$  inicializačne určené podľa kroku 1. algoritmu 5.2 ako  $d(v) := \text{ideg}(v)$ , čomu zodpovedá prvý riadok tabuľky. Z prvého riadku ihneď vidno, že  $V_0 = \{6, 7\}$ , takže do druhého stĺpca tabuľky zapíšeme v riadkoch s  $i = 1, 2$  vrcholy 6, 7.

V súlade s krokom 2. algoritmu 5.2 postupne v riadkoch s  $i = 1, 2, 3, \dots, n-1$  znižujeme značky  $d(w)$  všetkých vrcholov  $w \in V^+(v_i)$ . Znížené značky  $d()$  sú v tabuľke vyznačené tučným fontom. Vrcholy, ktorých značky sa vynulovali zapíšeme do ďalších riadkov 2. stĺpca tabuľky. Takto pokračujeme kým nezaradíme do 2. stĺpca tabuľky všetky vrcholy. Ak by sa stalo pre niektoré  $i$ , že jeho riadok tabuľky neobsahuje žiadnu nulovú značku  $d$ , znamenalo by to, že digraf  $\vec{G}$  nie je acyklický. Pre náš konkrétny príklad je monotónne očíslovanie digrafu  $\vec{G}$

nasledujúce:

$$v_1 = 6, v_2 = 7, v_3 = 3, v_4 = 4, v_5 = 5, v_6 = 1, v_7 = 2, v_8 = 8.$$



Obr. 5.3: Diagram digrafu z obrázku 5.2 nakreslený tak, že vrcholy digrafu sú zoradené podľa monotónneho očíslovania zľava doprava.

Všimnime si, že ak zobrazíme v rovine vrcholy acyklického digrafu v jednej línii usporiadané zľava doprava podľa monotónneho očíslovania, potom v takomto diagrame šípky predstavujúce orientované hrany smerujú len zľava doprava, čo dobre vidieť na obrázku 5.3.

Mnohé algoritmy možno veľmi zjednodušiť, ak môžeme predpokladať, že spracovávaný acyklický digraf je monotónne očíslovaný. Preto mnohé algoritmy pre acyklické digrafy začínajú monotónnym očíslovaním digrafu.

## 5.2 Najkratšia a najdlhšia cesta v acyklických digrafoch

Úloha hľadania najkratšej cesty v acyklických digrafoch je podstatne jednoduchšia než vo všeobecných digrafoch alebo grafoch. Na riešenie tejto úlohy môžeme samozrejme použiť ktorýkoľvek z algoritmov pre všeobecné digrafy (základný algoritmus, Fordov algoritmus, Dijkstrov algoritmus), keď nám však ide o rýchlosť, môžeme s výhodou využiť skutočnosť, že vrcholy acyklického digrafu možno monotónne očíslovať. Toto očíslovanie nie je výpočtovo náročné (ako sme ukázali, má zložitnosť  $O(m+n)$ ) a dá sa využiť pre hľadanie všetkých najkratších  $u-v$  ciest z pevného vrchola  $u$  nasledovne: Predpokladajme, že všetci predchodcovia  $v$  vrchola  $w$ , ktorí sú dosiahnuteľní z vrchola  $u$ , majú značky  $t(v)$  rovné dĺžke najkratšej  $u-v$  cesty, ostatní predchodcovia vrchola  $w$  majú značky  $\infty$ .

Nech  $t(w) = \infty$ . Ak pre všetkých predchodcov  $v$  vrchola  $w$  s konečnou značkou  $t(v)$  urobíme

$$\text{ak } t(w) > t(v) + c(v, w), \quad \text{potom } t(w) := t(v) + c(v, w),$$

potom aj značka  $t(w)$  je rovná dĺžke najkratšej  $u$ - $w$  cesty.

**Definícia 5.5.** Nech  $G = (V, H, c)$  ( $\vec{G} = (V, H, c)$ ) je hranovo ohodnotený graf (digraf),  $u \in V, v \in V$ . **Najdlhšia (orientovaná)  $u$ - $v$  cesta** v grafe  $G$  (digrafe  $\vec{G} = (V, H, c)$ )  $G$  je tá (orientovaná)  $u$ - $v$  cesta, ktorá má zo všetkých  $u$ - $v$  ciest najväčšiu dĺžku.

Acyklické digrafy majú ešte jednu výnimočnú vlastnosť. Kým úloha hľadania najdlhšej cesty je vo všeobecných grafoch a digrafoch NP-ťažká (nemáme pre ne polynomiálny algoritmus), v acyklických digrafoch možno na hľadanie najdlhšej cesty použiť základný alebo Fordov algoritmus - oba pracujúce so záporne vzatými pôvodnými cenami hrán. Pozor, Dijkstrov algoritmus pre hrany ohodnotené zápornými číslami zlyhá!

**Algoritmus 5.3. Algoritmus na výpočet najkratšej  $u$ - $v$  cesty v neorientovane súvislom acyklickom hranovo ohodnotenom digrafe  $\vec{G} = (V, H, c)$ .**

- **Krok 1.** Monotónne očísľuj vrcholy digrafu  $\vec{G}$ , nech  $\mathcal{P} = v_1, v_2, \dots, v_k$  postupnosť vrcholov digrafu  $\vec{G}$  zoradená podľa monotónneho očísľovania. Zisti index vrchola  $u$  v postupnosti  $\mathcal{P}$ . Nech  $i$  je index taký, že  $u = v_i$ .
- **Krok 2.** Pre každý vrchol  $v \in V$  priradi značky  $t(v), x(v)$ . Polož  $t(u) := 0, t(j) := \infty$  pre všetky  $j \neq u, j \in V$ . Polož  $x(j) := 0$  pre všetky  $j \in V$ .
- **Krok 3.** Pre všetky vrcholy  $w$  výstupnej hviezdy vrchola  $v_i$  také, že  $w \neq v_i$ , urob:  
Ak  $t(w) > t(v_i) + c(v_i, w)$ , potom  $t(w) = t(v_i) + c(v_i, w)$ , a  $x(w) := v_i$ .
- **Krok 4.**  $i := i + 1$ . Ak  $i = n$  STOP, inak GOTO Krok 3.



Po skončení práce algoritmu majú všetky vrcholy  $v$ , ktoré sú dostupné z vrchola  $u$  konečnú značku  $t(v)$  rovnú dĺžke najkratšej  $u$ - $v$  cesty, vrcholy, ktoré nie sú z  $u$  dosiahnuteľné majú značku  $\infty$ . Hľadanú najkratšiu  $u$ - $v$  cestu

zrekonštruujeme podľa značiek  $x()$  spätne od vrchola  $v$  – obsahuje vrcholy  $v, x(v), x(x(v)), \dots, u$ .

Je vidieť, že Krok 1. algoritmu vyžaduje  $O(n + m)$  operácií, Krok 2.  $O(n)$  operácií, Krok 3. počas celého algoritmu  $O(m)$  operácií a konečne Krok 4.  $O(n)$  operácií. Celý algoritmus má zložitosť  $O(n + m)$ .

Predchádzajúci algoritmus sa hodí i pre acyklické digrafy so všeobecným (teda aj záporným) ohodnotením hrán. Dá sa použiť na hľadanie najdlhšej  $u-v$  cesty tak, že k acyklickému digrafu  $\vec{G} = (V, H, c)$  vezmeme digraf  $\vec{G}' = (V, H, c')$  s tými istými množinami vrcholov a orientovaných hrán a s cenou  $c'$  definovanou predpisom  $c'(h) = -c(h)$  pre všetky  $h \in H$ . Najkratšia  $u-v$  cesta v digrafe  $\vec{G}'$  je potom najdlhšou cestou v digrafe  $\vec{G}$ .

Iná možnosť je pozmeniť tento algoritmus tak, aby priamo hľadal najdlhšie cesty. Na to stačí:

- inicializovať značky v kroku 2. nasledovne:  
 $t(u) := 0, t(j) := -\infty$  pre všetky  $j \neq u, j \in V$ .
- V kroku 3. obrátiť rozhodovacie nerovnosti:  
 Ak  $t(w) < t(v_i) + c(v_i, w)$ , potom  $t(w) = t(v_i) + c(v_i, w)$  a  $x(w) := v_i$ .

**Definícia 5.6.** Hovoríme, že acyklický digraf  $\vec{G} = (V, H)$  je **tranzitívny**, ak pre ľubovoľné dve hrany  $(u, v) \in H, (v, w) \in H$  existuje hrana  $(u, w) \in H$ .

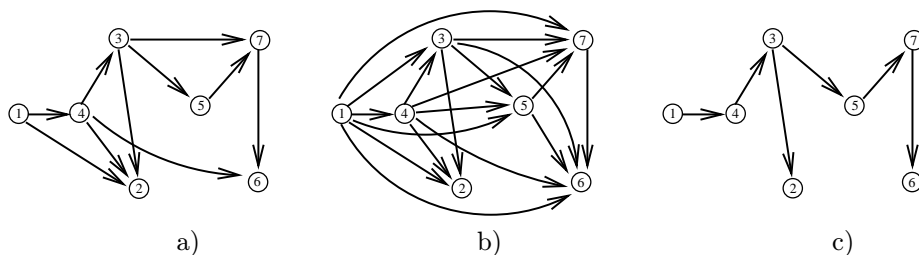
Lahko sa dá dokázať nasledujúca veta.

**Veta 5.5.** *Acyklický digraf  $\vec{G}$  je tranzitívny práve vtedy, keď ku každej orientovanej  $u-v$  ceste v  $\vec{G}$  existuje hrana  $(u, v) \in H$ .*

**Definícia 5.7.** Hovoríme, že digraf  $\vec{G}_T$  je **tranzitívny uzáver** digrafu  $\vec{G}$ , ak  $\vec{G}_T$  je minimálny tranzitívny digraf obsahujúci ako podgraf digraf  $\vec{G}$ .

Hovoríme, že digraf  $\vec{G}_R$  je **tranzitívna redukcia** digrafu  $\vec{G}$ , ak  $\vec{G}_R$  je minimálny faktorový podgraf digrafu  $\vec{G}$  s rovnakou dosiahnuteľnosťou vrcholov ako digraf  $\vec{G}$ .

Všimnime si, že tranzitívny uzáver  $\vec{G}_T$  digrafu  $\vec{G} = (V, H)$  má tú istú množinu vrcholov  $V$  ako digraf  $\vec{G}$  – vyplýva to z požiadavky  $\vec{G} \subseteq \vec{G}_T$  a z minimality digrafu  $\vec{G}_T$ . Pre tranzitívnu redukciu  $\vec{G}_R = (V_R, H_R)$  digrafu  $\vec{G}$  je



Obr. 5.4:

- a) Digraf  $\vec{G}$ . b) Transzitívny uzáver  $\vec{G}_T$  digrafu  $\vec{G}$ .  
 c) Transzitívna redukcia  $\vec{G}_R$  digrafu  $\vec{G}$ .

rovnosť  $V_R = V$  priamo požadovaná definíciou ( $\vec{G}_R$  má byť faktorový podgraf digrafu  $\vec{G}$ ).

Transzitívny uzáver  $\vec{G}_T = (V, H_T)$  digrafu  $\vec{G} = (V, H)$  zostrojíme tak, že pre každú usporiadanú dvojicu vrcholov  $u, v \in V, u \neq v$  skontrolujeme, či je  $v$  dosiahnuteľné z  $u$ , a ak áno, pridáme orientovanú hranu  $(u, v)$  do hranovej množiny  $H_T$  transzitívneho uzáveru.

Transzitívnu redukciu digrafu  $\vec{G}$  zostrojíme tak, že z hranovej množiny  $H$  digrafu  $\vec{G}$  postupne vylúčime všetky hrany  $(u, v)$  také, že existuje orientovaná  $u-v$  cesta obsahujúca viac než jednu hranu.

## 5.3 Metódy časového plánovania

Práce na veľkom projekte možno postupne rozčleniť na čiastočné projekty, tie pozostávajú z vykonania niekoľkých úloh, tie sa zase môžu skladať ešte z menších častí. Pri plánovaní projektu sa musíme na istej rozlišovacej úrovni zastaviť a definovať si tzv. **elementárne činnosti**, ktoré už z prijatého rozlišovacieho hľadiska považujeme za nedeliteľné.

Technologický postup budovania projektu dovoľuje niektoré elementárne činnosti vykonávať súčasne, ale medzi niektorými činnosťami môže technológia predpisovať precedenčný vzťah. Budeme hovoriť, že činnosť  $A$  **predchádza** činnosť  $B$  a píšat  $A \prec B$ , ak sa činnosť  $B$  môže začať vykonávať až po skončení vykonávania činnosti  $A$ . Ak  $A \prec B$  budeme tiež hovoriť že činnosť

$A$  je **predchodca** činnosti  $B$  alebo činnosť  $B$  je **následník** činnosti  $A$ . Vzťah  $A \prec B$  je binárnou reláciou na množine všetkých elementárnych činností. Budeme ju volať **precedenčná relácia** alebo **relácia precedencie**. (Pozri definíciu binárnej relácie 1.3 na strane 14.)

Tak napríklad základy domu sa môžu betónovať až po skončení výkopových prác a dokončení bednení, steny domu sa môžu začať stavať až potom, čo sú dokončené základy, stropy sa môžu začať budovať až po dokončení stien, strecha po dokončení stropov, vnútorné omietky po dokončení stropov súčasne s vonkajšími omietkami, ktoré čakajú na dokončenie stien.

Relácia precedencie  $\prec$  je **tranzitívna**, t. j. platí: ak  $A \prec B$ ,  $B \prec C$  potom aj  $A \prec C$ . Ak elementárna činnosť  $B$  musí čakať na skončenie činnosti  $A$  a činnosť  $C$  musí čakať na skončenie činnosti  $B$ , tým skôr musí činnosť  $C$  čakať na ukončenie činnosti  $A$ .

Ďalšou dôležitou vlastnosťou relácie precedencie  $\prec$  je, že pre žiadne  $A \in \mathcal{E}$  neplatí  $A \prec A$ , v opačnom prípade by začiatok činnosti  $A$  musel čakať na jej vlastný koniec, čo je technologický nezmysel. Z toho ďalej vyplýva, že neexistuje postupnosť činností  $A_1, A_2, \dots, A_n$  taká, že

$$A_1 \prec A_2 \prec \dots \prec A_n \prec A_1,$$

lebo potom by z tranzitivity vyplývalo  $A_1 \prec A_1$ . Relácia  $\prec$  je teda antireflexívna (pozri definíciu 1.4 na strane 14 v nulte kapitole.) Posledná vlastnosť je formou istej "acykličnosti" relácie  $\prec$ .

Pre modelovanie následností nie je potrebné zadať všetky usporiadané dvojice elementárnych činností, ktoré sú v relácii precedencie. Pre úplný popis technologického procesu plne postačuje také zadanie, v ktorom ku každej činnosti zadáme len jej bezprostredných následníkov.

Hovoríme, činnosť  $A$  **bezprostredne predchádza** činnosti  $B$  a píšeme  $A \prec\prec B$ , ak  $A \prec B$  a neexistuje činnosť  $C$  taká, že  $A \prec C$  a súčasne  $C \prec B$ . Ak  $A \prec\prec B$  budeme tiež hovoriť že činnosť  $A$  je **bezprostredný predchodca** činnosti  $B$  alebo činnosť  $B$  je **bezprostredný následník** činnosti  $A$ .

Vykonanie každej elementárnej činnosti vyžaduje istý čas. Pri metóde CPM, ktorú budeme ďalej popisovať, sa predpokladá, že čas na vykonanie každej elementárnej činnosti je nemenný, jednotlivé elementárne činnosti však majú vo všeobecnosti rôzne časy vykonávania.

**Úloha časového plánovania**  $\mathcal{U}$  je daná množinou elementárnych činností  $\mathcal{E}$ , precedenčnou reláciou  $\prec$  na množine  $\mathcal{E}$  a reálnou funkciou  $c : \mathcal{E} \rightarrow \mathbb{R}$  priradujúcou každej činnosti  $A \in \mathcal{E}$  jej trvanie  $c(A)$ .

Často býva vhodné modelovať trvanie a následnosť elementárnych operácií vo forme vrcholovo ohodnoteného digrafu  $\vec{\mathbb{G}}_{\prec} = (V, H, c)$ , ktorého množinou vrcholov je množina všetkých elementárnych činností, ohodnotenie  $c(v) > 0$  vrchola  $v \in V$  je čas spracovania príslušnej elementárnej činnosti a množinou orientovaných hrán je  $H = \{(A, B) \mid A, B \in V, A \prec B\}$ . Digraf  $\vec{\mathbb{G}}_{\prec}$  nazveme **precedenčným digrafom** alebo **digrafom precedencie**  $\prec$  pre príslušnú úlohu  $\mathcal{U}$  časového plánovania.

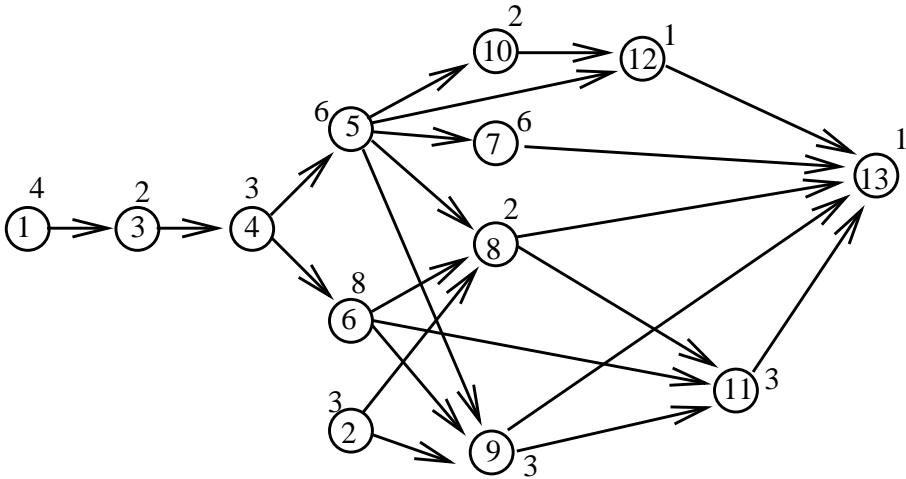
Vzhľadom na tranzitivitu relácie  $\prec$  je príslušný precedenčný digraf tranzitívny – má veľmi veľa hrán. Ak totiž existuje orientovaná  $A$ – $B$  cesta, potom musí existovať v hranovej množine  $H$  (vzhľadom na tranzitivitu) aj priama orientovaná hrana  $(A, B)$ . Z vyššie popísaných vlastností precedenčnej relácie  $\prec$  ďalej vyplýva veľmi dôležitá vlastnosť precedenčného digrafu – precedenčný digraf  $\vec{\mathbb{G}}_{\prec}$  je **acyklický**.

Jednoduchšou štruktúrou (vzhľadom na počet hrán) je **digraf bezprostrednej precedencie**  $\vec{\mathbb{G}}_{\prec\prec} = (V, H, c)$ , kde  $V$  je množina všetkých elementárnych činností,  $c(v) > 0$  čas vykonávania činnosti  $v$  a kde  $H = \{(A, B) \mid A, B \in V, A \prec\prec B\}$ . Všimnime si, že  $\vec{\mathbb{G}}_{\prec\prec}$  je tranzitívnou redukciou digrafu  $\vec{\mathbb{G}}_{\prec}$  a tiež  $\vec{\mathbb{G}}_{\prec\prec}$  je tranzitívnym uzáverom digrafu  $\vec{\mathbb{G}}_{\prec\prec}$ . Digraf bezprostrednej precedencie  $\vec{\mathbb{G}}_{\prec\prec}$  je tiež **acyklický**.

Technologický predpis zadáva úlohu časového plánovania – množinu elementárnych činností, ich trvanie a vzťah bezprostrednej precedencie v tabuľke, kde ku každej elementárnej činnosti okrem jej trvania udáva zoznam jej bezprostredných následníkov. Túto tabuľku budeme volať **technologická tabuľka projektu**. Nebolo by chybou, ak by tabuľka obsahovala aj iných ako bezprostredných následníkov, títo však v tabuľke nie sú potrební a zbytočne rozširujú rozsah vstupných dát.

Činnosť	Číslo	Trvanie činnosti	Následné činnosti
Výkopy	1	4	3
Inžinierske siete	2	3	8 9
Bednenie základov	3	2	4
Betónovanie základov	4	3	5 6
Obvodové múry	5	6	7 8 9 10 12
Priečky	6	8	8 9 11
Strecha	7	6	13
Elektrické inštalácie	8	2	11 13
Vodovodné inštalácie	9	3	11 13
Vonkajšie omietky	10	2	12
Vnútorne omietky	11	3	13
Okná, dvere	12	1	13
Kolaudácia	13	1	-

Tabuľka 5.1: Technologická tabuľka projektu s precedenčným digrafom na obrázku 5.5.



Obr. 5.5: Diagram digrafu bezprostrednej precedencie k technologickej tabuľke 5.1 projektu.



Vytvoriť **rozvrh** pre danú úlohu  $\mathcal{U}$  časového plánovania znamená každej elementárnej činnosti  $A$  priradiť časový interval  $\langle b_A, c_A \rangle$ ,  $b_A < c_A$  v ktorom sa bude činnosť  $A$  vykonávať. Tu  $b_A$  znamená začiatok vykonávania činnosti  $A$  ( $b$  – beginning time),  $c_A$  je koniec vykonávania činnosti  $A$  ( $c$  – completion time). **Prípustný rozvrh** úlohy  $\mathcal{U}$  je taký rozvrh pre úlohu  $\mathcal{U}$ , kde pre ľubovoľné elementárne činnosti  $A, B$  platí:

1.  $c_A - b_A = c(A)$
2. ak  $A \prec B$ , potom  $b_A < c_A \leq b_B < c_B$

Všimnime si, že na základe vlastnosti 1. prípustného rozvrhu stačí pre každú elementárnu činnosť  $A$  určiť jej začiatok  $b_A$ , čas ukončenia sa vypočíta ako  $c_A = b_A + c(A)$ .

Prípustných rozvrhov pre daný problém časového plánovania je veľa, z nich by sme chceli vybrať optimálny z nejakého hľadiska. Veľmi často sa ako kritérium optimality berie  $C_{\max}$  čas ukončenia poslednej činnosti, teda

$$C_{\max} = \max\{c_A \mid A \in \mathcal{E}\},$$

pričom sa predpokladá, že projekt začína v čase 0. Veličinu  $C_{\max}$  budeme nazývať **trvanie rozvrhu**, čo dobre vystihuje skutočnosť, že projekt sa odovzdáva odberateľovi až po ukončení poslednej elementárnej činnosti. V anglosaskej literatúre sa pre veličinu  $C_{\max}$  používa tiež termín **makespan**.

Základnou otázkou časového plánovania je pre danú úlohu  $\mathcal{U}$  určiť prípustný rozvrh taký, aby príslušné trvanie rozvrhu  $C_{\max}$  bolo minimálne. Označme  $T$  minimum zo všetkým možných trvaní rozvrhov. Veličinu  $T$  budeme nazývať **trvanie projektu**. Ďalšou otázkou je zistiť tie elementárne činnosti, ktorých každé oneskorenie má vplyv na predĺženie trvania  $T$  skúmaného projektu. Pre ostatné činnosti nás zaujíma, pokiaľ najneskôr sa ich ukončenie môže oneskoriť bez toho, aby sa predĺžil čas  $T$  trvania projektu.

Ľubovoľná orientovaná cesta v digrafe  $\vec{\mathbb{G}}_{\leftarrow}$  predstavuje technologicky prípustné poradie vykonania elementárnych činností a súčet čistých trvaní spracovania všetkých elementárnych činností na tejto ceste (t. j. bez časových rezerv) sa dá vypočítať ako súčet ohodnotení vrcholov na tejto ceste. Naopak ľubovoľnému technologicky prípustnému poradiu spracovania elementárnych činností možno priradiť orientovanú cestu v digrafe  $\vec{\mathbb{G}}_{\leftarrow}$  so súčtom ohodnotení vrcholov rovným čistému času (bez časových rezerv) potrebnému na spracovanie daného poradia činností. Trvanie projektu  $T$  musí byť teda rovné maximu zo súčtov ohodnotení vrcholov všetkých orientovaných ciest v digrafe  $\vec{\mathbb{G}}_{\leftarrow}$ .

Pre každú elementárnu činnosť určujeme dva významné časové okamihy. **Najskôr možný začiatok** elementárnej činnosti t. j. prvý časový okamih<sup>4</sup>, kedy možno začať činnosť vykonávať pri dodržaní precedenčnej relácie  $\prec$ . Ak už poznáme hodnotu trvania projektu  $T$ , chceme pre každú elementárnu činnosť poznať **najneskôr nutný koniec** tejto činnosti definovaný ako posledný časový okamih<sup>5</sup>, po ktorý sa ukončenie tejto činnosti môže oneskoriť bez predĺženia trvania projektu  $T$ . Znovu pripomíname, že predpokladáme pritom, že projekt sa začína v čase 0.

Nech  $z(v)$ ,  $k(v)$  sú najskôr možný začiatok, resp. najneskôr nutný koniec elementárnej činnosti  $v$ , ktorej trvanie je  $c(v)$ . **Časovú rezervu**  $R(v)$  činnosti  $v$  definujeme ako

$$R(v) = k(v) - z(v) - c(v).$$

**Kritická činnosť** je taká činnosť  $v$ , pre ktorú je  $R(v) = 0$ . **Kritická cesta** v digrafe  $\vec{G}_{\prec}$  je taká orientovaná  $u-v$  cesta  $\mu(u, v)$  v digrafe  $\vec{G}_{\prec}$ , kde  $\text{iddeg}(u) = 0$ ,  $\text{odeg}(v) = 0$ , ktorá obsahuje len kritické činnosti.

*Poznámka.* Dá sa ukázať, že súčet ohodnotení vrcholov každej kritickej cesty v  $\vec{G}_{\prec}$  sa rovná trvaniu projektu  $T$ .

Ako sme už spomenuli, trvanie  $T$  projektu možno určiť ako maximum súčtov ohodnotení vrcholov všetkých orientovaných ciest v digrafe  $\vec{G}_{\prec}$ , t. j. ako „dĺžku“ najdlhšej orientovanej cesty v digrafe  $\vec{G}_{\prec}$ , ak za „dĺžku“ orientovanej cesty berieme súčet ohodnotení jej vrcholov.

Podobne možno určiť  $z(v)$  – najskôr možný začiatok a  $k(v)$  – najneskôr nutný koniec elementárnej činnosti  $v \in V$  pomocou „dĺžok“ najdlhších orientovaných ciest nasledovne. Označme  $\text{idmax}(v)$  ako maximum „dĺžok“ všetkých orientovaných  $u-v$  ciest v  $\vec{G}_{\prec}$ ,  $\text{odmax}(v)$  ako maximum „dĺžok“ všetkých orientovaných  $v-u$  ciest v  $\vec{G}_{\prec}$ . Potom

$$z(v) = \text{idmax}(v) - c(v) \quad k(v) = T - \text{odmax}(v) + c(v)$$

Existujú dva možné prístupy k nájdeniu trvania projektu  $T$ , najskôr možných začiatkov  $z(v)$  a najneskôr nutných koncov  $k(v)$  elementárnych činností  $v \in V$ . Jedným z nich je klasický prístup využívajúci štandardné algoritmy na hľadanie najdlhších orientovaných ciest v digrafe. Tento prístup však vyžaduje taký model úlohy časového plánovania  $\mathcal{U}$ , v ktorom by boli činnosti modelované

<sup>4</sup>Najmenší čas meraný od času 0 – t. j. od začiatku vykonávania projektu.

<sup>5</sup>Najväčší čas meraný od času 0 – t. j. od začiatku vykonávania projektu.

orientovanými hranami s ohodnotením rovným trvaniu príslušnej elementárnej činnosti – digraf  $\vec{\mathbb{G}}_{\leftarrow}$  bezprostrednej precedencie sa pre tento účel nehodí. Klasický prístup k úlohe časového plánovania je rozobratý na strane 152 v časti 5.4 tejto kapitoly.

Druhý prístup na určenie veličín  $T$ ,  $z(v)$ ,  $k(v)$  využíva postupy bežné v teórii výrobných rozvrhov ([1], [12]). Nedokáže síce využiť štandardné algoritmy na hľadanie extrémálnych ciest, avšak algoritmy na výpočet optimálneho rozvrhu využívajú ako model rozvrhovacej úlohy  $\mathcal{U}$  digraf  $\vec{\mathbb{G}}_{\leftarrow}$  bezprostrednej precedencie a sú veľmi jednoduché. Algoritmus na určenie najskôr možných začiatkov elementárnych činností je založený na postupnom zaraďovaní takej činnosti  $v$ , ktorá nemá nezaraďeného predchodcu tak, že začiatok činnosti  $v$  položíme do okamihu, kedy sa skončí spracovanie posledného predchodcu vrchola  $v$ .

Nasledujúce dva algoritmy budú postupne zaraďovať činnosti – vrcholy digrafu  $\vec{\mathbb{G}}_{\leftarrow}$  – do rozvrhu a pri zaraďení súčasne im určovať ich najskôr možný začiatok, resp. najneskôr nutný koniec. Aktuálne nezaraďené vrcholy budeme držať v množine  $\mathcal{I}$ , podgraf digrafu  $\vec{\mathbb{G}}_{\leftarrow}$  indukovaný množinou<sup>6</sup>  $\mathcal{I}$  označíme  $\vec{\mathbb{G}}$ . Vstupný, resp. výstupný stupeň vrchola  $v$  v digrafe  $\vec{\mathbb{G}}$  budeme označovať symbolom  $\text{ideg}_{\vec{\mathbb{G}}}(v)$ , resp.  $\text{odeg}_{\vec{\mathbb{G}}}$ , analogický význam v digrafe  $\vec{\mathbb{G}}_{\leftarrow}$  budú mať  $\text{ideg}(v)$ , resp.  $\text{odeg}(v)$ . Symbol  $V^+(v)$ , resp.  $V^-(v)$ , podľa definície 1.18 na str. 23 označuje množinu koncových vrcholov všetkých hrán vychádzajúcich z vrchola  $v$ , resp. množinu začiatočných vrcholov všetkých hrán vchádzajúcich do vrchola  $v$ .

**Algoritmus 5.4. Algoritmus I. na určenie najskôr možných začiatkov  $z(v)$  elementárnych činností v digrafe  $\vec{\mathbb{G}}_{\leftarrow} = (V, H, c)$ .**

- **Krok 1.** Inicializácia. Polož  $\mathcal{I} := V$ ,  $\vec{\mathbb{G}} := \vec{\mathbb{G}}_{\leftarrow}$ .
- **Krok 2.** Vyber taký vrchol  $v \in \mathcal{I}$ , že  $\text{ideg}_{\vec{\mathbb{G}}}(v) = 0$ .  
Ak  $V^-(v) = \emptyset$ , potom  $z(v) := 0$ .  
Ak  $V^-(v) \neq \emptyset$ , potom  $z(v) := \max\{z(u) + c(u) \mid u \in V^-(v)\}$ .
- **Krok 3.** Polož  $\mathcal{I} := \mathcal{I} - \{v\}$ .  
Ak  $\mathcal{I} = \emptyset$ , polož  $T := \max\{z(w) + c(w) \mid w \in V, \text{odeg}(w) = 0\}$  a STOP.  
Inak polož  $\vec{\mathbb{G}} := \vec{\mathbb{G}} - \{v\}$  a GOTO Krok 2.



<sup>6</sup>Pozri definíciu 1.17 podgrafu indukovaného množinou na str. 23.

**Algoritmus 5.5. Algoritmus I. na určenie najneskôr nutných koncov  $k(v)$  elementárnych činností v digrafe  $\vec{\mathbb{G}}_{\leftarrow} = (V, H, c)$ .**

- **Krok 1.** Inicializácia. Polož  $\mathcal{I} := V$ ,  $\vec{G} := \vec{\mathbb{G}}_{\leftarrow}$ .
- **Krok 2.** Vyber taký vrchol  $v \in \mathcal{I}$ , že  $\text{odeg}_{\vec{G}}(v) = 0$ .  
Ak  $V^+(v) = \emptyset$ , potom  $k(v) := T$ .  
Ak  $V^+(v) \neq \emptyset$ , potom  $k(v) := \min\{(k(u) - c(u)) \mid u \in V^+(v)\}$ .
- **Krok 3.** Polož  $\mathcal{I} := \mathcal{I} - \{v\}$ . Ak  $\mathcal{I} = \emptyset$ , STOP.  
Inak polož  $\vec{G} := \vec{G} - \{v\}$  a GOTO Krok 2.



Odhadneme časovú zložitosť algoritmu 5.4. Predpokladajme, že  $n = |V|$ ,  $m = |H|$ . Krok 1. spočíva v skopírovaní digrafu  $\vec{\mathbb{G}}_{\leftarrow} = (V, H, c)$  do digrafu  $\vec{G}$ , čo sa dá urobiť v čase  $O(m + n)$ . Krok 2. vyžaduje v najhoršom prípade prehlídnutie  $|\mathcal{I}|$  vrcholov, kým nájdeme vrchol s nulovým vstupným stupňom. Prekontrolovanie stupňa vrchola sa dá urobiť v konštantnom čase  $O(1)$ , ak budeme stupne vrcholov udržiavať v značke  $d()$  podobne ako v algoritme 5.2 na strane 136. Nájdenie vrcholov vstupného stupňa 0 vyžaduje spolu najviac  $n + (n - 1) + \dots + 2 + 1 = \frac{n \cdot (n + 1)}{2}$  krokov – dá sa urobiť v čase  $O(n^2)$ . Nakoniec výpočet všetkých maxim vo všetkých  $n$  iteráciách kroku 2. sa dá urobiť v čase  $O(m)$ . Počas celého algoritmu vykonanie všetkých iterácií kroku 2. algoritmu 5.4 vyžiada čas  $O(m + n^2)$ . Nakoniec jedno vykonanie kroku 3. vyžaduje konštantný čas a  $n$  iterácií kroku 3. sa dá urobiť v čase  $O(n)$ . Celková zložitosť algoritmu je teda  $O(m + n^2)$ , resp. s využitím  $m < n^2$  môžeme stanoviť jeho zložitosť ako  $O(n^2)$ . Analogicky sa ukáže, že aj zložitosť algoritmu 5.5 je  $O(n^2)$ .

Použitím monotónneho očíslovania digrafu  $\vec{\mathbb{G}}_{\leftarrow}$  možno navrhnúť algoritmy s menšou zložitosťou. V nasledujúcich algoritmoch okrem značiek  $z(v)$ ,  $k(v)$  zavedieme aj značku  $x(v)$  – predposledný vrchol orientovanej  $u-v$  cesty s najväčším súčtom ohodnotení vrcholov, a značku  $y(v)$  – druhý vrchol orientovanej  $v-u$  cesty s najväčším súčtom ohodnotení vrcholov.

**Algoritmus 5.6. Algoritmus II. na určenie najskôr možných začiatkov  $z(v)$  elementárnych činností v digrafe  $\vec{\mathbb{G}}_{\leftarrow} = (V, H, c)$ .**

- **Krok 1.** Vytvor monotónne očíslovanie  $v_1, v_2, \dots, v_n$  vrcholov digrafu  $\vec{\mathbb{G}}_{\leftarrow}$ .

- **Krok 2.** Každému vrcholu  $v \in V$  priradiť dve značky  $z(v)$ ,  $x(v)$ .  
Pre každé  $v \in V$  inicializačne polož  $x(v) := 0$ ,  $z(v) := 0$ .
- **Krok 3.** Postupne pre  $k = 1, 2, \dots, n - 1$  urob:
 

Pre všetky také vrcholy  $w$  výstupnej hviezdy vrchola  $v_k$ , že  $w \neq v_k$ , urob:  
Ak  $z(w) < z(v_k) + c(v_k)$ , potom  $z(w) := z(v_k) + c(v_k)$  a  $x(w) := v_k$ .
- **Krok 4.** Vypočítaj trvanie projektu
 
$$T := \max\{z(w) + c(w) \mid w \in V, \text{odeg}(w) = 0\}$$



**Algoritmus 5.7. Algoritmus II. na určenie najneskôr nutných koncov  $k(v)$  elementárnych činností v digrafe  $\vec{G}_{\leftarrow} = (V, H, c)$ .**

- **Krok 1.** Vytvor monotónne očíslovanie  $v_1, v_2, \dots, v_n$  vrcholov digrafu  $\vec{G}_{\leftarrow}$ .
- **Krok 2.** Každému vrcholu  $v \in V$  priradiť dve značky  $k(v)$ ,  $y(v)$ . Nech  $T$  je trvanie projektu.  
Pre každé  $v \in V$  inicializačne polož  $k(v) := T$ ,  $y(v) := 0$ .
- **Krok 3.** Postupne pre  $i = n - 1, n - 2, \dots, 1$  urob:
 

Pre všetky vrcholy  $w$  výstupnej hviezdy vrchola  $v_i$  také, že  $w \neq v_i$ , urob:  
Ak  $k(v_i) > k(w) - c(w)$ ,  
potom  $k(v_i) := k(w) - c(w)$  a  $y(v_i) := w$ .



Výsledkom oboch algoritmov máme pre každý vrchol  $v \in V$  sú dve značky  $z(v)$  – najskôr možný začiatok elementárnej činnosti  $v$  a  $k(v)$  – najneskôr nutný koniec elementárnej činnosti  $v$ . Navyiac pre každú činnosť  $v$  máme značku  $x(v)$  – činnosť, ktorej čas ukončenia najviac ohraničuje začiatok činnosti  $v$  a značku  $y(v)$  – činnosť, ktorej začiatok najviac ohraničuje koniec činnosti  $v$ .

Aká je zložitosť algoritmu 5.6? Nech  $n = |V|$ ,  $m = |H|$ . Monotónne očíslovanie vrcholov v kroku 1. možno urobiť v čase  $O(m + n)$ . Inicializačný krok 2. vyžaduje čas  $O(n)$ . V kroku 3. treba vypočítať  $n$  značiek  $z(v)$ , a  $n$  značiek  $x(v)$ . Pritom treba urobiť v kroku 3. iba toľko porovnaní, koľko je hrán – t. j.  $m$  porovnaní. Krok 3. vyžaduje celkom čas  $O(m + n)$ . Nakoniec výpočet maxima v kroku 4. možno urobiť v čase  $O(n)$ . Celý algoritmus má teda zložitosť

$O(m+n)$ . Analogicky sa ukáže, že zložitosť algoritmu 5.7 je tiež  $O(m+n)$ . Monotónne usporiadanie digrafu  $\vec{G} \leftarrow$  pomohlo navrhnúť efektívnejší algoritmus (najmä v prípade keď  $m$  je podstatne menšie než  $n^2$ ).

**Príklad 5.2.** Urobte časovú analýzu projektu s precedenčným digrafom z obrázku 5.5 na strane 144.

Riešenie: Pre každú činnosť treba vypočítať najskôr možný začiatok a najneskôr nutný koniec. Algoritmy 5.6 a 5.7 využívajú monotónne očíslovanie vrcholov precedenčného digrafu. Ak si však prezrieme tabuľku s výstupnými hviezdami vrcholov, môžeme konštatovať, že náš precedenčný digraf už je monotónne očíslovaný.

Výstupné hviezdy			Tabuľka pre výpočet najskôr možných začiatkov činností															
$v$	$c(v)$	$V^+(v)$	$v$	$c(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(v)$												
			-		-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	4	3	1	4	0		4											
2	3	8 9	2	3	0								3	3				
3	2	4	3	2	4				6									
4	3	5 6	4	3	6				9	9								
5	6	7 8 9 10 12	5	6	9							15	15	15	15		15	
6	8	8 9 11	6	8	9								17	17		17		
7	6	13	7	6	15													21
8	2	11 13	8	2	17											19		
9	3	11 13	9	3	17											20		
10	2	12	10	2	15												17	
11	3	13	11	3	20													23
12	1	13	12	1	17													
13	1	-	13	1	23													

$$T = \max\{z(v) + c(v) \mid v \in V\} = 24.$$

Výpočet najskôr možných začiatkov počítame v tabuľke, v ktorej prvý riadok obsahuje pre každý vrchol  $v$  inicializačné hodnoty značiek  $z()$  určené podľa kroku 2. algoritmu 5.6. Ďalšie riadky tabuľky obsahujú vývoj značiek  $z()$  po preznačovaní vrcholov výstupnej hviezdy vrchola uvedeného v prvom stĺpci tabuľky – riadky sú tvorené podľa kroku 3. algoritmu 5.6. Pre uľahčenie počítania tabuľka obsahuje stĺpec  $c(v)$  s trvaniami príslušných činností a stĺpec  $z(v)$  výsledných najskôr možných začiatkov elementárnych činností.

Tabuľka pre výpočet najneskôr nutných koncov činností

$v$	$c(v)$	$k(v) - c(v)$	$k(v)$	$k(v)$												
				1	2	3	4	5	6	7	8	9	10	11	12	13
-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	1	23	24													24
12	1	22	23												23	
11	3	20	23											23		
10	2	20	22										22			
9	3	17	20									20				
8	2	18	20								20					
7	6	17	23							23						
6	8	9	17						17							
5	6	11	17					17								
4	3	6	9				9									
3	2	4	6			6										
2	3	14	17		17											
1	4	0	4	4												

Výpočet najneskôr nutných koncov činností použitím algoritmu 5.7 urobíme v podobnej tabuľke popisujúcej vývoj značiek  $k()$ . Podobne ako vo všetkých predchádzajúcich príkladoch, v počítači stačí na uloženie značiek  $z()$ ,  $k()$  po jednom  $n$ -rozmernom poli, čo by stačilo aj pre ručný výpočet, keby sme značky prepisovali na mieste. V tom prípade by však nebolo vidno ich vývoj počas behu algoritmu.

Všimnime si ešte, že výpočet najneskôr nutných koncov činností pomocou výstupných hviezd je jednoduchší ako výpočet najskôr možných začiatkov pomocou výstupných hviezd vrcholov. Bolo by možné algoritmus 5.6 preformulovať pre uloženie precedenčného digrafu formou vstupnej hviezdy, čím by sa trochu zjednodušil výpočet najskôr možných začiatkov činností.

Výsledok časovej analýzy daného projektu možno zhrnúť do nasledujúcej tabuľky:

Činnosť $v$	1	2	3	4	5	6	7	8	9	10	11	12	13
Trvanie činnosti $c(v)$	4	3	2	3	6	8	6	2	3	2	3	1	1
Najskôr možný začiatok $z(v)$	0	0	4	6	9	9	15	17	17	15	20	17	23
Najneskôr nutný koniec $k(v)$	4	17	6	9	17	17	23	20	20	22	23	23	24
Rezerva	0	14	0	0	2	0	2	1	0	5	0	5	0
Kritická činnosť	•		•	•		•			•		•		•

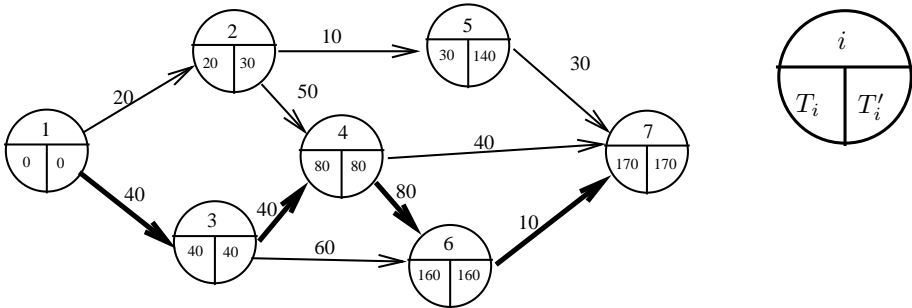
## 5.4 Klasická interpretácia metódy CPM a jej nedostatky

Majme úlohu časového plánovania  $\mathcal{U}$  danú množinou elementárnych činností  $\mathcal{E}$ , precedenčnou reláciou  $\prec$  na množine  $\mathcal{E}$  a reálnou funkciou  $c : \mathcal{E} \rightarrow \mathbb{R}$  priradujúcou každej činnosti  $A \in \mathcal{E}$  jej trvanie  $c(A)$ .

Klasická CPM metóda modeluje návaznosť úloh pre tento prípad tzv. sieťovým digrafom, v ktorom je každej úlohe  $A \in \mathcal{E}$  pridelená práve jedna hrana ohodnotená dĺžkou spracovania  $c(A)$  úlohy  $A$ , vrcholy – predstavujú časové začiatky a konce spracovania elementárnych činností (reprezentovaných s nimi incidentnými hranami).

Sieťový digraf je neorientovane súvislý acyklický digraf, obsahujúci práve jeden prameň  $z$ , ktorý nazveme **začiatok vykonávania projektu** a jediný stok  $k$ , ktorý nazveme **koniec vykonávania projektu**<sup>7</sup>.

Veľmi často sa predpokladá, že  $V = \{1, 2, \dots, n\}$  a že  $z = 1, k = n$ .



Obr. 5.6: Diagram sieťového digrafu, aký nájdete v mnohých učebniciach.

Ako ho zostrojíte z technologickej tabuľky bez fiktívnych činností s nulovým trvaním, väčšinou autori taktne zamlčia.

Predpokladáme začiatok vykonávania projektu v čase 0. **Najskôr možný začiatok  $T_i$  činností vychádzajúcich z vrchola  $i$**  je prvý časový okamih, kedy skončí vykonávanie poslednej z činností vchádzajúcich do vrchola  $i$ . **Trva-**

<sup>7</sup>Prameň v digrafe bol definovaný ako vrchol, z ktorého sú všetky ostatné vrcholy dosiahnuteľné. Stok v digrafe bol definovaný ako vrchol dosiahnuteľný zo všetkých ostatných vrcholov. V acyklickom digrafe môže existovať najviac jeden prameň a najviac jeden stok  $k$  a platí  $\text{iddeg}(z) = 0$   $\text{odeg}(k) = 0$ .



**nie projektu**  $T_n$  je najskôr možný začiatok činností vychádzajúcich z konca projektu  $n$ . **Najneskôr nutný koniec**  $T'_i$  **činností vchádzajúcich do vrchola**  $i$  je posledný časový okamih, po ktorý sa činnosti vchádzajúce do vrchola  $i$  môžu oneskoriť bez toho, aby sa zväčšilo trvanie projektu  $T_n$ . **Časová rezerva**  $R_i$  **vo vrchole**  $i$  je  $R_i = T'_i - T_i$ .

Hodnotu  $T_n$  trvania projektu určíme ako  $T_n = d_{\max}(1, n)$ , t. j. dĺžku najdlhšej orientovanej cesty od začiatku projektu 1 do konca projektu  $n$ . Každú orientovanú cestu dĺžky trvania projektu  $T_n$  v sieťovom digrafe nazveme **kritickou cestou** (kritických ciest môže existovať aj viac). Činnosti ležiace na kritickej ceste sa nazývajú **kritické činnosti**.

Pre každý vrchol  $i$  sieťového digrafu vypočítame  $T_i$ , t. j. najskôr možný začiatok činností vychádzajúcich z vrchola  $i$ , ako

$$T_i = d_{\max}(1, i)$$

a  $T'_i$  najneskôr nutný koniec činností vchádzajúcich do vrchola  $i$  ako

$$T'_i = T_n - d_{\max}(i, n)$$

kde  $d_{\max}(x, y)$  je dĺžka najdlhšej orientovanej  $x$ - $y$  cesty.

Pre vrchol  $i$  ležiaci na nejakej kritickej ceste je  $T_i = T'_i$ , pre ostatné vrcholy je  $T_i < T'_i$ .

Výhodou metódy CPM je, že sa hodnoty  $T_i$ ,  $T'_i$  dajú vypočítať pomocou dĺžok najdlhších orientovaných ciest (na rozdiel od predchádzajúcej časti, v ktorej by sme potrebovali predefinovať dĺžku cesty na súčet ohodnotení vrcholov). Pre metódu CPM sa tak dajú použiť štandardné algoritmy na hľadanie extrémálnych ciest v acyklickom digrafe. Touto výhodou sa však výpočet výhod končí.

Potiaže pri praktickej aplikácii CPM metódy začnú hneď pri pokuse vytvoriť k vstupnej technologickej tabuľke príslušný sieťový digraf  $\vec{G}_S$ . Bez dodania ďalších tzv. fiktívnych činností s nulovým trvaním je to vo väčšine prípadov neriešiteľný problém. Ak pripustíme dodanie fiktívnych činností, potom úloha zostrojíte k danej technologickej tabuľke príslušný sieťový digraf má viacero riešení.

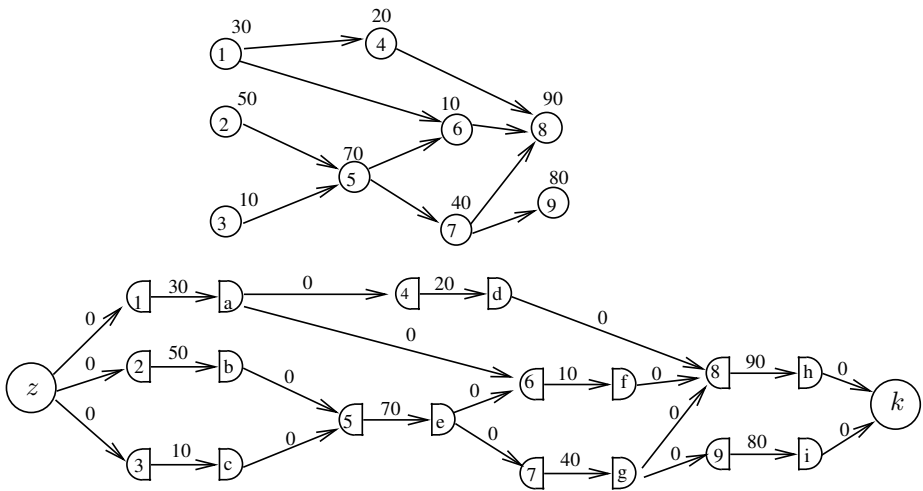
Jeden z postupov je vytvoriť najprv digraf  $\vec{G}_{\leftarrow} = (V, H)$  bezprostrednej precedencie a potom z neho vytvoriť príslušný sieťový digraf  $\vec{G}_S = (V_S, H_S)$  nasledovne: Každý vrchol množiny  $v \in V$  rozdeliť na vstupnú časť  $v_I$  a výstupnú

časť  $v_O$  a do množiny  $V_S$  dať dva špeciálne vrcholy  $z, k$  a všetky „polovičky“  $v_I, v_O$ :

$$V_S = \{z\} \cup \{k\} \cup \{v_I \mid v \in V\} \cup \{v_O \mid v \in V\},$$

do množiny  $H_S$  vložiť všetky orientované hrany typu  $(u_O, v_I)$  s nulovou cenou  $c()$ , ak existovala v  $\vec{G}_{\leftarrow}$  orientovaná hrana  $(u, v)$ , všetky orientované hrany typu  $(v_I, v_O)$  s cenou trvania činnosti  $v$  a nakoniec všetky orientované hrany typu  $(z, v_I)$  pre také  $v$ , ktoré nemá v  $\vec{G}_{\leftarrow}$  predchodcov a  $(u_O, k)$  pre také  $u$ , ktoré nemá v  $\vec{G}_{\leftarrow}$  následníkov, oba posledné typy hrán s nulovou cenou.

$$H_S = \{(u_O, v_I) \mid (u, v) \in H\} \cup \{(v_I, v_O) \mid v \in V\} \cup \{(z, v_I) \mid v \in V, \text{iddeg}(v) = 0\} \cup \{(u_O, k) \mid u \in V, \text{odeg}(u) = 0\}$$



Obr. 5.7:

Konštrukcia sieťového digrafu  $\vec{G}_S$  (dole) z precedenčného digrafu  $\vec{G}_{\leftarrow}$  (hore).

Výsledný digraf  $\vec{G}_S$  má veľké množstvo orientovaných hrán s nulovou cenou - sú to tzv. **fiktívne činnosti**, okrem toho, že zabezpečujú správnu návaznosť elementárnych činností, pre prax nič nehovoria a sú dodané len preto, aby sa učinilo zadosť požiadavkám na sieťový digraf. Sú síce známe postupy, ako znižovať počet fiktívnych hrán, avšak úloha formulovaná ako: „K danému digrafu bezprostrednej precedencie nájsť príslušný sieťový digraf s minimálnym počtom

*fiktívnych hrán*“ sa ukázala byť NP-ťažká, zatiaľ čo algoritmy na riešenie všetkých otázok sieťového plánovania prezentované v predchádzajúcej časti 5.3 majú zložitosť  $O(n + m)$ .

Pri metóde CPM definujeme najskôr možný začiatok činností vychádzajúcich do vrchola  $v$  a najneskôr nutný koniec činností vychádzajúcich z vrchola  $v$ . Čo to však znamená pre prax, že dve či viac činností vchádza či vychádza do či zo spoločného vrchola, teória okolo CPM nevysvetľuje. Navyše sieťový digraf možno skonštruovať tak, že žiadne dve reálne činnosti nemajú spoločný vstupný ani výstupný vrchol – takáto je napríklad aj konštrukcia popisovaná v tejto časti a ilustrovaná na obrázku 5.7.

Postup opísaný v predchádzajúcej časti 5.3 definoval len najskôr možný začiatok a najneskôr nutný koniec pre jednotlivú činnosť, čo sú jasné a zrozumiteľné pojmy. Predchádzajúca časť vychádza z postupov teórie rozvrhov – Sequencing and Scheduling (pozri napr. v [1], [12]), ktorá rieši mnohé problémy rozvrhovania zložitých úloh a operácií pre jeden, alebo aj viac strojov. Software, ktorý rieši praktické rozvrhovacie problémy, je jedným z najčastejšie ponúkaných programových vybavení v súčasnej dobe.

Na základe týchto poznatkov môžeme tvrdiť, že postupy prezentujúce najprv technologické tabuľky a potom príslušný sieťový digraf (pozri napr. [6]), (ktorý vraj vznikol z technologických tabuliek) bez fiktívnych hrán, nie sú celkom korektné, „z pedagogických dôvodov“ tam vznikol najprv sieťový digraf a až potom tabuľka, čo však vo všeobecnosti bez fiktívnych hrán nie je možné.

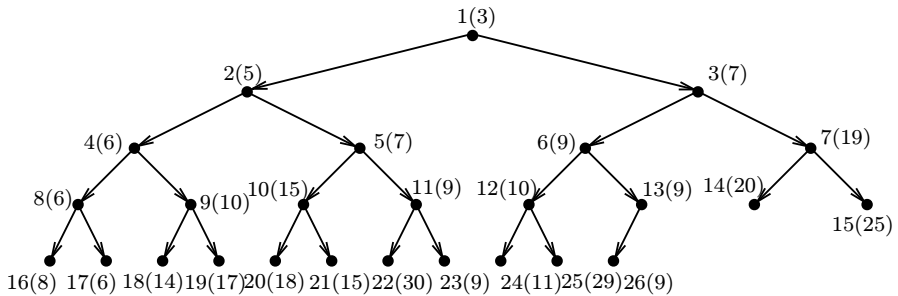
Iná literatúra (napr. [17]) síce nutnosť zavedenia fiktívnych hrán nezamľčuje, avšak ňou navrhovaný postup vytvorenia sieťového digrafu popísaný na štyroch stranách textu je nealgoritmický – je formulovaný do niekoľkých formálnych pravidiel konštrukcie sieťového digrafu. Iná literatúra založená na klasickej interpretácii metódy CPM (napr. [8]) je z matematického hľadiska v poriadku.

Metóda CPM so svojimi sieťovými digrafmi sa však v literatúre široko udomácnila, používa ju napr. softwarový produkt MS Projekt, (ktorú dala firma Microsoft niektorým vysokým školám za režijnú cenu), preto je dobré vedieť i o klasickej prístupe k riešeniu otázok časového plánovania.

## 5.5 Aplikácie

### 5.5.1 Prioritný strom a halda

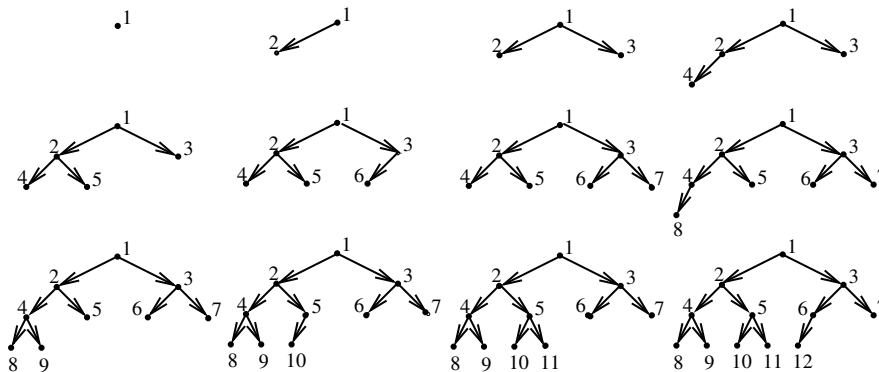
Úroveň vrchola v koreňovom strome je jeho vzdialenosť od koreňa (pri jednotkovej dĺžke hrany). Výška koreňového stromu je maximum z úrovni jeho vrcholov. **Prioritný strom** je binárny koreňový strom, vrcholovo ohodnotený, v ktorom z existencie pravého syna vrchola vyplýva existencia jeho ľavého syna a ktorý má  $2^k$  vrcholov na každej úrovni  $k$ , ktorá je menšia, než výška. O ohodnoteniach vrcholov prioritného stromu platí: Ohodnotenie vrchola  $v$  je menšie alebo rovné než ohodnotenia obidvoch jeho synov.



Obr. 5.8: Prioritný strom. Čísla v zátvorkách sú ohodnotenia vrcholov. Úroveň vrchola 1 je 0, úroveň vrcholov 2 a 3 je 1, úroveň vrcholov 4 - 7 je 2 atď.

Príklad prioritného stromu s 26 vrcholmi vidíme na obrázku 5.8. Nárast prioritného stromu postupne od jedného vrchola vidíme na obr. 5.9. Všetky prioritné stromy s  $n$  vrcholmi sú izomorfné. Všimnime si, že ak má vrchol  $k$  synov, títo sú  $2k - 1$  - ľavý a  $2k + 1$  - pravý. Koreňom prioritného stromu je vždy vrchol 1 a otcom každého vrchola  $k \neq 1$  je vrchol  $\lfloor k/2 \rfloor$ , kde hranatá zátvorka  $\lfloor x \rfloor$  znamená celú časť čísla  $x$ . Z obrázku 5.8 je tiež vidieť, že úroveň vrchola  $k$  je  $\lfloor \log_2(k) \rfloor$ , a tak výška prioritného stromu s  $n$  prvkami je menšia alebo rovná číslu  $\log_2(n)$ .

Prioritný strom má veľmi jednoduchú implementáciu ako vektor. Takúto interpretáciu nazývame **halda**. Halda pre prioritný strom z obrázku 5.8 je v nasledujúcej tabuľke



Obr. 5.9: Jediný možný spôsob pridávania vrcholov do prioritného stromu.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
3	5	7	6	7	9	19	6	10	15	9	10	9	20	25	8	6	14	17	18	15	30	9	11	29	9

kde čísla v hornom riadku tabuľky len ukazujú pozíciu, skutočne sa v počítači ukladá len druhý riadok ohodnotení vrcholov.

S haldou sa dajú efektívne robiť nasledujúce operácie:

- vloženie vrchola s ľubovoľným ohodnotením
- vybratie vrchola s najmenším ohodnotením
- zmena ohodnotenia ľubovoľného vrchola

Pri vkladaní prvku  $w$  s ľubovoľným ohodnotením  $c(w)$  rozšírime haldu podľa princípu na obrázku 5.9. V tomto momente však pravdepodobne rozšírený binárny strom prestane mať tú vlastnosť, že ohodnotenie synov ľubovoľného vrchola  $v$  je väčšie alebo rovné ako ohodnotenie vrchola  $v$ . Aby sme obnovili túto vlastnosť, povymieňame označenia vrcholov v strome nasledovne:

#### Algoritmus 5.8. Pridanie prvku do haldy.

- **Krok 1.** Označ symbolom  $r$  pridaný vrchol do haldy.  
Ak  $r$  je koreňom, STOP.  
Inak označ  $p(r)$  otca vrchola  $r$ .

- **Krok 2.** Ak  $r$  je koreňom, alebo  $c(r) \geq c(p(r))$ , STOP.  
{Strom má všetky vlastnosti haldy.}
- **Krok 3.** Ak  $c(r) < c(p(r))$ , vymeň ohodnotenia vrcholov  $r$  a  $p(r)$ , t. j. polož  $temp := c(r)$   $c(r) := c(p(r))$ ,  $c(p(r)) := temp$ . GOTO Krok 2.



Týmto spôsobom sme nechali ohodnotenie pridaného vrchola „prebublať“ po ceste do koreňa na jeho správne miesto. Pretože (jediná) cesta z pridaného vrchola do koreňa má dĺžku  $\lceil \log_2 n \rceil$ , urobíme vloženie nového vrchola do haldy v čase  $O(\log n)$ .

Vybratie prvku s minimálnym ohodnotením z haldy je trošku zložitejšie

#### Algoritmus 5.9. Vybratie minimálneho prvku z haldy.

- **Krok 1.** Vyber ohodnotenie koreňa haldy. Koreňu daj ohodnotenie posledného vrchola haldy. Zruš posledný vrchol haldy s incidentnou hranou. Označ  $r := 1$  koreň haldy. Označ  $l(r)$ ,  $p(r)$  ľavého a pravého syna prvku  $r$ .
- **Krok 2.** Ak  $r$  nemá následníkov, alebo ak  $c(r) \leq \min\{c(l(r)), c(p(r))\}$ , STOP. Strom už má vlastnosti haldy.
- **Krok 3.** Ak  $r$  nemá pravého syna, alebo ak  $c(l(r)) \leq c(p(r))$ , vymeň ohodnotenia vrcholov  $r$ ,  $l(r)$ , polož  $r := l(r)$  a GOTO Krok 2. Inak vymeň ohodnotenia vrcholov  $r$ ,  $p(r)$ , polož  $r := p(r)$  a GOTO Krok 2.



Tu sme zase nechali „utopiť“ nové ohodnotenie koreňa na jeho správne miesto, čo zase môžeme urobiť v čase  $O(\log n)$ .

Halda je veľmi dôležitá dátová štruktúra. Pomocou nej je možné zostrojiť veľmi jednoduchý triediaci algoritmus, ktorý má najlepšiu možnú zložitosť  $O(n \cdot \log n)$  takto:

Uložíme  $n$  čísel do haldy – na to potrebujeme čas  $O(n \cdot \log n)$  – a potom  $n$ -krát vyberieme z haldy minimum – na čo znova stačí čas  $O(n \cdot \log n)$ .

Použitím haldy možno zlepšiť implementáciu Dijkstrovho algoritmu nasledovne:

Dočasne označené vrcholy si ukladáme do haldy. Uloženie vrchola možno urobiť v čase  $O(\log n)$  a v takom istom čase možno vybrať z haldy vrchol s najnižším ohodnotením. Tak zložitosť tohto algoritmu zlepšime z pôvodných  $O(n^2)$  na  $O(m + n \cdot \log n)$ .<sup>8</sup>

Prioritný strom je len jednou ukážkou z veľmi veľkého počtu aplikácií stromov pri návrhu optimálnych údajových štruktúr. Čitatelia, ktorí si zvolili informatiku za svoju profesionálnu orientáciu, sa iste s teóriou a praxou údajových štruktúr stretnú bližšie.

## 5.6 Cvičenia

- Ukážte, že acyklický digraf s  $n$  vrcholmi môže mať najviac  $\frac{n \cdot (n - 1)}{2}$  orientovaných hrán. Návod: Začnite z úplného grafu  $K_n = (V, H)$  s množinou vrcholov  $V = \{1, 2, \dots, n\}$ . Pre každú hranu  $\{u, v\}$  zaveďte orientáciu  $(u, v)$  práve vtedy, keď  $u < v$ .
- Nech  $\vec{G} = (V, H)$  je acyklický digraf. Nech  $k$  je maximum dĺžok všetkých orientovaných ciest v digrafe  $\vec{G}$  pri jednotkovom ohodnutí všetkých orientovaných hrán digrafu  $\vec{G}$ . Označme  $L(0)$  množinu všetkých vrcholov so vstupným stupňom 0. Pre  $i = 1, 2, \dots, k$  označme  $L(i)$  množinu všetkých vrcholov, do ktorých vedie nejaká cesta dĺžky  $i$ , ale do ktorých nevedie žiadna cesta dĺžky  $i + 1$  (t. j. najdlhšia cesta vedúca do vrcholov z  $L(i)$  má dĺžku  $i$ ).
  - Ukážte, že  $L(i)$  je nezávislá množina – t. j. žiadne dva prvky z  $L(i)$  nie sú priľahlé.
  - Ukážte, že  $\{L(i) \mid i = 0, 1, 2, \dots, k\}$  je rozklad množiny  $V$ .
- Za rovnakých predpokladov ako v predchádzajúcom cvičení definujme  $P(0)$  ako množinu všetkých vrcholov s výstupným stupňom 0. Nech  $P(i)$  pre  $i = 1, 2, \dots, k$  je množina všetkých vrcholov digrafu  $\vec{G}$ , z ktorých

---

<sup>8</sup>Toto má však význam len vtedy, keď  $m$  je podstatne menšie než  $n^2$ . Grafové modely reálnych dopravných sietí tento predpoklad spĺňajú. Praktické výpočty však ukazujú, že pre malé  $n$  výpočtová režia na spravovanie haldy môže dokonca výpočet predĺžiť, skutočné efekty sa dosiahnu až pre veľké  $n$ .

najdlhšia cesta v  $\vec{G}$  má dĺžku  $i$ . Ukážte, že platia analogické tvrdenia ako a), b) z predchádzajúceho cvičenia.

4. Sú  $L(i)$ ,  $P(i)$  z predchádzajúcich dvoch cvičení maximálne nezávislé množiny? Je medzi nimi najpočetnejšia nezávislá množina? V akom vzťahu sú množiny  $L(i)$  a  $P(i)$ ? Aký je vzťah monotónneho očíslovania acyklického digrafu k množinám  $L(i)$ ,  $P(i)$ ?
5. Transitívny uzáver acyklického digrafu je určený jednoznačne (t. j. pre acyklický digraf  $\vec{G}$  existuje jediný digraf  $\vec{G}_T$ , ktorý je transitívnym uzáverom digrafu  $\vec{G}$ ). Je jednoznačne určená aj transitívna redukcia  $\vec{G}_R$  acyklického digrafu? Ako je to s jednoznačnosťou transitívneho uzáveru a transitívnej redukcie digrafu, ktorý nie je acyklický? (Urobte transitívnu redukciu orientovanej kružnice, t. j. digrafu, ktorého hrany a vrcholy sa dajú usporiadať do orientovaného cyklu.)
6. Preformulujte algoritmy 5.1, 5.2, 5.3, 5.6 tak, aby namiesto výstupných hviezd vrcholov využívali ich vstupné hviezdy.

### Počítačové cvičenia

7. Naprogramujte algoritmy 5.1, 5.2 na monotónne očíslovanie acyklického digrafu. Tieto algoritmy možno použiť i na rýchle zistenie, či je daný digraf acyklický (bez identifikácie konkrétneho cyklu v digrafe).
8. Napíšte program na časovú analýzu projektu zadaného technologickou tabuľkou. Zamyslite sa, čo by bolo treba urobiť pre implementáciu klasickej metódy CPM pomocou sieťového digrafu.



## Kapitola 6

# Pochôdzky v grafoch

Existuje mnoho praktických úloh, ktorých podstata tkvie v postupnom navštívení všetkých hrán alebo všetkých vrcholov nejakého súvislého grafu a návrate do východzieho miesta. Túto pochôdzku treba urobiť optimálne z hľadiska prejdenej vzdialenosti. Postupné "navštevovanie" hrán alebo vrcholov môžeme v teórii grafov veľmi dobre opísať cyklom, uzavretým ťahom alebo sledom. Hľadanie optimálnej trasy pre navštívenie všetkých hrán resp. vrcholov možno teda matematicky modelovať ako hľadanie najkratšieho sledu (ťahu resp. cyklu) obsahujúceho všetky hrany resp. vrcholy daného grafu. Úloha nájsť najkratší sled obsahujúci všetky hrany súvislého grafu je známa ako **úloha čínskeho poštára**, anglicky **Chinese Postman Problem**, úloha nájsť najkratší sled obsahujúci všetky vrcholy súvislého grafu je slávna pod menom **úloha obchodného cestujúceho**, anglicky **Travelling Salesman Problem** – skratkou **TSP** (občas ju využijeme aj my).

Aj keď sa na prvý pohľad zdá, že ide o veľmi podobné úlohy, opak je pravdou. Kým úloha čínskeho poštára má exaktný polynomiálny algoritmus riešenia, pre úlohu obchodného cestujúceho taký algoritmus nepoznáme.

### 6.1 Eulerovské ťahy

Každý z nás sa už pravdepodobne stretol s úlohou nakresliť daný obrázok jedným ťahom. Ak príslušný obrázok pokladáme za diagram niektorého grafu

alebo multigrafu, ide o úlohu zostrojiť v príslušnom grafe sled obsahujúci každú hranu grafu práve raz. Odtiaľto je i motivácia termínu teórie grafov „ťah“ ako sledu, v ktorom sa žiadna hrana neopakuje. Ukazuje sa, že hľadanie takýchto ťahov v grafoch má mnohé oveľa závažnejšie aplikácie ako sú úlohy zábavnej matematiky, a preto sa mu budeme venovať podrobnejšie.

**Definícia 6.1.** Hovoríme, že sled  $s(u, v)$  v súvislom grafe  $G = (V, H)$  je **eulerovský**, ak obsahuje všetky hrany grafu  $G$ .

**Definícia 6.2.** Hovoríme, že graf  $G = (V, H)$  je **eulerovský**, ak v ňom existuje uzavretý eulerovský ťah.

Pretože ťah je špeciálnym prípadom sledu, je definíciou 6.1 presne vymedzený pojem **eulerovský ťah** ako taký ťah  $t(u, v)$  v súvislom grafe  $G$ , ktorý obsahuje všetky hrany grafu  $G$ . Keďže ťah obsahuje každú hranu grafu  $G$  práve raz, postupnosť vrcholov a hrán ťahu  $t(u, v)$  predstavuje postup, ako nakresliť diagram grafu  $G$  „jedným ťahom“.

V každom súvislom grafe  $G$  existuje uzavretý eulerovský sled obsahujúci každú hranu grafu  $G$  práve dvakrát – postup na zostrojenie takéhoto sledu dáva napr. Tarryho algoritmus. Ľahko zistíme, že nie v každom súvislom grafe musí existovať uzavretý eulerovský ťah. Existuje však veľmi jednoduché kritérium na zistenie, či je daný súvislý graf eulerovský.

**Veta 6.1. (Euler, 1736.)** *Súvislý graf  $G = (V, H)$  je eulerovský práve vtedy, keď stupne všetkých vrcholov grafu  $G$  sú párne.*

DÔKAZ.

1. Ak v grafe  $G$  existuje uzavretý eulerovský ťah  $t$ , potom stupeň každého vrchola je párny, pretože počet hrán ktorými ťah  $t$  z každého vrchola  $v$  vyšiel sa rovná počtu hrán, ktorými sme do vrchola  $v$  vošli.

2. Majme súvislý graf  $G = (V, H)$  so všetkými vrcholmi párneho stupňa. Najprv ukážeme, že v ňom existuje uzavretý ťah. Vyjdime z ľubovoľného vrchola  $v \in V$  a konštruujeme postupne ťah  $t(v, v)$  tak, že k nemu pridávame doteraz neprejdené hrany tak dlho, pokiaľ nedôjdeme do vrchola  $w$ , z ktorého už nevychádza žiadna neprejdená hrana. Musí byť  $w = v$ , keby totiž  $w \neq v$ , počet hrán, ktorými sme do vrchola  $w$  vošli by bol o 1 väčší než počet hrán, ktorými sme z vrchola  $w$  vyšli. Ak by vrchol  $w$  neobsahoval žiadne ďalšie hrany, mal by nepárny stupeň, čo je spor s predpokladom.

Nech  $t(v, v)$  je uzavretý ťah s najväčším počtom hrán v súvislom grafe  $G = (V, H)$  so všetkými vrcholmi párneho stupňa. Označme  $H_t$  množinu všetkých

hrán grafu  $G$ , ktoré sa nevyskytujú v ťahu  $t(v, v)$  a  $V_t$  množinu všetkých vrcholov incidentných s hranami z  $H_t$ . Aspoň jeden vrchol  $w \in V_t$  musí ležať na ťahu  $t(v, v)$ , inak by neexistovala cesta z  $v$  do žiadneho vrchola z  $V_t$  a graf  $G$  by nebol súvislý. V grafe  $G_t = (V_t, H_t)$  má každý vrchol párny stupeň. Vezmime jeho komponent určený vrcholom  $w \in V_t$ ,  $w \in t(v, v)$  a v ňom uzavretý ťah  $t(w, w)$  začínajúci a končiaci vo  $w$ . Teraz môžeme ťah  $t(v, v)$  prerušiť v prvom výskyte vrchola  $w$  a do prerušeného miesta vložiť ťah  $t(w, w)$ . Dostaneme tak uzavretý ťah s väčším počtom hrán ako mal ťah  $t(v, v)$ , čo je spor s predpokladom, že ťah  $t(v, v)$  má najväčší možný počet hrán. ■

1. Definície 6.1 a 6.2 možno rozšíriť i pre multigrafy. Takisto veta 6.1 platí i pre multigrafy, pričom sa ani v dôkaze nič nezmení.

2. Dôkaz vety 6.1 dáva i návod, ako v súvislom grafe  $G$  so všetkými vrcholmi párneho stupňa zostrojiť uzavretý eulerovský ťah:

Zostrojíme najprv ľubovoľný uzavretý ťah  $t(v, v)$  tak, že začneme vo vrchole  $v$  ľubovoľnou hranou a potom k ťahu pridávame ešte neprejudené hrany, pokiaľ sa dá. Párny stupeň vrcholov grafu  $G$  zaručuje, že skončíme vo východzom vrchole  $v$ .

Ak v ťahu  $t(v, v)$  existuje vrchol  $w$  taký, že  $t(v, v)$  neobsahuje všetky hrany incidentné s vrcholom  $w$ , vložíme do ťahu  $t(v, v)$  na mieste  $w$  uzavretý ťah  $t(w, w)$ , ktorý obsahuje len (niektoré) hrany, ktoré sa nevyskytujú v  $t(v, v)$ . Takto pokračujeme v predlžovaní ťahu  $t(v, v)$  dovtedy, kým neobsahuje všetky hrany grafu  $G$ .

3. Podobne ako v neorientovanom grafe môžeme aj v digrafe  $\vec{G}$  definovať **eulerovský orientovaný ťah** ako taký orientovaný ťah v  $\vec{G}$ , ktorý obsahuje všetky orientované hrany digrafu  $\vec{G}$ . Digraf  $\vec{G}$  nazveme **eulerovský**, ak obsahuje uzavretý eulerovský orientovaný ťah. Veta 6.1 sa dá preformulovať nasledovne: Silne súvislý digraf je eulerovský práve vtedy, keď pre každý jeho vrchol  $v$  platí  $\text{iddeg}(v) = \text{odeg}(v)$ .

4. Z vety 6.1 vyplýva, že v súvislom grafe  $G$  existuje otvorený eulerovský ťah práve vtedy, keď  $G$  má práve dva vrcholy nepárneho stupňa.

**Algoritmus 6.1. Fleuryho algoritmus na hľadanie uzavretého eulerovského ťahu v súvislom grafe  $G = (V, H)$ , v ktorom majú všetky vrcholy párny stupeň.**

- **Krok 1.** Začni v ľubovoľnom vrchole a do ťahu  $\mathcal{T}$  zaraď ľubovoľnú s ním incidentnú hranu.
- **Krok 2.** Ak sú do ťahu  $\mathcal{T}$  zaradené všetky hrany grafu  $G$ , STOP.

- **Krok 3.** Ako ďalšiu hranu zaraď do ľahu  $\mathcal{T}$  takú hranu incidentnú s jeho posledným vrcholom, po vybratí ktorej sa podgraf grafu  $G$  pozostávajúci z nevybraných hrán a s nimi incidentných vrcholov nerozpadne na
  - dva netriviálne komponenty
  - netriviálny komponent a izolovaný začiatok ľahu  $\mathcal{T}$ .
- GOTO krok 2.



Pokiaľ sa Fleuryho algoritmus aplikuje s pomocou diagramu grafu, obsahuje v kroku 3. silný intuitívny prvok pri kontrole súvislosti grafu pozostávajúceho z nevybraných hrán a s nimi incidentných vrcholov. Algoritmizácia tejto kontroly je síce možná, ale takto vzniknutý algoritmus by nebol práve najefektívnejší. Pre počítačovú implementáciu je vhodnejší nasledujúci labyrintový algoritmus, ktorého správnosť síce nie je zrejماً na prvý pohľad, ale je veľmi efektívny.

**Algoritmus 6.2. Labyrintový algoritmus na hľadanie uzavretého eulovského ľahu v súvislom grafe  $G = (V, H)$ , v ktorom majú všetky vrcholy párny stupeň.**

- **Krok 1.** Začni z ľubovoľného vrchola  $u \in V$ . Nech sled  $\mathcal{S}$  inicializačne pozostáva z jediného vrchola  $u$ . Polož  $w := u$  — vrchol  $w$  je posledný vrchol doteraz vytvoreného sledu  $\mathcal{S}$ .
- **Krok 2.** Ako ďalšiu hranu vyber podľa nižšie uvedených pravidiel do sledu  $\mathcal{S}$  hranu  $\{w, v\}$ . Zaznač si smer použitia hrany  $\{w, v\}$ . Ak doteraz vrchol  $v$  ešte nebol zaradený do sledu  $\mathcal{S}$ , označ hranu  $\{w, v\}$  ako hranu prvého príchodu. Ďalej zaznamenaj tzv. **spätnú postupnosť** — poradie hrán, v ktorom sa v slede  $\mathcal{S}$  vyskytujú po druhýkrát. Pri výbere hrany dodržuj nasledujúce pravidlá:

(L1): Každú hranu možno v jednom smere použiť iba raz

(L2): Poradie zaraďovania hrán:

- nepoužité hrany
- hrany použité raz
- hrana prvého príchodu (ak niet inej možnosti)

- **Krok 3.** Ak taká hrana neexistuje – STOP.  
Spätná postupnosť určuje hľadaný eulovský ľah.

- **Krok 4.** Inak polož  $w := v$  a pokračuj krokom 2.

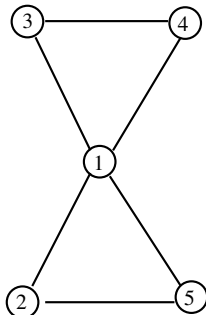


Rozbor zložitosti labyrintového algoritmu<sup>1</sup>. Pred zaradením každej hrany do sledu  $\mathcal{S}$  systematicky skúmame množinu hrán  $H$  z hľadiska zaraditeľnosti do sledu  $\mathcal{S}$ , pričom musíme vyskúšať najviac  $m$  hrán. Keďže sled  $\mathcal{S}$  má  $2m$  hrán, môžeme zhora odhadnúť počet preskúmaní hrán na  $2m \times m = 2m^2$ , kde  $m = |H|$ . Keďže rozhodnutie o zaradení jednej hrany vyžaduje konštantný počet elementárnych operácií, môžeme uzavrieť, že počet všetkých operácií nutných pre zostrojenie sledu  $\mathcal{S}$  možno zhora ohraničiť číslom  $K \cdot m^2$ . Zostrojenie sledu  $\mathcal{S}$  možno urobiť v čase  $O(m^2)$ . Pri konštrukcii sledu  $\mathcal{S}$  možno súčasne konštruovať aj spätnú postupnosť tak, že pri zaraďovaní hrany  $h$  do sledu  $\mathcal{S}$  naviac skontrolujeme, či už bola hrana  $h$  použitá aj v opačnom smere – ak áno, zaradíme ju aj do spätnej postupnosti. Tento postup vyžaduje pre každú hranu dvakrát skontrolovať, či je zaraďovaná po druhýkrát, čo sa dá urobiť v čase  $O(m)$ . Zložitosť takejto implementácie labyrintového algoritmu je  $O(m^2)$ . Ak ju chceme vyjadriť iba pomocou počtu vrcholov  $n$  grafu  $G$ , stanovíme ju na  $O(n^4)$  s využitím nerovnosti  $m < n^2$ .

*Poznámka.* Pri Tarryho algoritme sme museli dbať len na to, aby sme hranu prvého príchodu zaradili do sledu len vtedy, keď niet inej možnosti. To umožnilo udržiavať pre každý vrchol  $v \in V$  zoznam použiteľných hrán  $\mathcal{Z}(v)$ , s ktorým sme navrhli implementáciu so zložitosťou  $O(m)$ . Konštrukcia sledu  $\mathcal{S}$  pri labyrintovom algoritme vyžaduje zaraďovanie hrán v poradí 1. nepoužité, 2. použité v opačnom smere, 3. hrana prvého príchodu, čo znemožňuje použiť podobný trik so zoznamom použiteľných hrán, ako to bolo v prípade konštrukcie Tarryho sledu. Po zaradení hrany  $\{u, v\}$  do sledu  $\mathcal{S}$  v smere  $(u, v)$  treba zistiť, či  $\{u, v\} \in \mathcal{Z}(v)$  a ak áno, potom treba zmeniť poradie v zozname použiteľných hrán  $\mathcal{Z}(v)$ , čo sa použitím jednoduchých dátových štruktúr asi nedá urobiť v konštantnom čase. Literatúra [13] však uvádza, že vhodnou implementáciou labyrintového algoritmu sa sa dá dosiahnuť zložitosť  $O(m + n)$ , resp.  $O(n^2)$ .

**Príklad 6.1.** Zostrojíme uzavretý eulerovský ťah pre graf definovaný diagramom na obrázku 6.1 vľavo. Podobne ako pri Tarryho algoritme budeme zapisovať použitie hrán, hrany prvého príchodu a navštívené vrcholy do tabuľky, ktorej tvar je vidieť na obrázku 6.1 vpravo. V prvom stĺpci tabuľky je uvedená hrana, ktorú práve pridávame do sledu, v ďalších stĺpcoch tabuľky sú značky použitia hrán a navštívenia vrcholov, ktoré sa zmenili zaradením tejto hrany.

<sup>1</sup>Odporúčam čitateľovi porovnať rozbor zložitosti labyrintového algoritmu s úvahami o zložitosti Tarryho algoritmu 3.1.



Hrany sledu $\mathcal{S}$	smer použitia hrany					navštívený vrchol					
	{1,2}	{1,3}	{1,4}	{1,5}	{2,5}	{3,4}	1	2	3	4	5
-											•
{5, 1}				←			•				
{1, 2}	⇒							•			
{2, 5}					→						
{5, 2}					←						
{2, 1}	←										
{1, 3}		⇒						•			
{3, 4}						⇒			•		
{4, 1}				←							
{1, 4}				→							
{4, 3}						←					
{3, 1}				←							
{1, 5}					→						

Obr. 6.1: Postup labyrintového algoritmu ak začíname z vrchola 5.

Začneme vrcholom 5, čomu v tabuľke zodpovedá prvý riadok, v ktorom je označený vrchol 5 značkou • ako preskúmaný. Ďalej systematicky od začiatku prehľadávame zoznam hrán a hľadáme prvú hranu incidentnú s aktuálnym posledným vrcholom sledu  $\mathcal{S}$  (tým je vrchol 5) zaraditeľnú do sledu. Tou je hrana  $\{1, 5\}$  v smere  $(5, 1)$ . Pretože vrchol 1 ešte nebol navštívený, označíme hranu  $\{1, 5\}$  značkou  $\leftarrow$  ako hranu prvého príchodu a vrchol 1 značkou • ako preskúmaný – pozri druhý riadok tabuľky. Vrchol 1 sa stáva posledným vrcholom sledu  $\mathcal{S}$ . Znovu systematicky od začiatku prehľadávame zoznam hrán a hľadáme prvú hranu zaraditeľnú do sledu  $\mathcal{S}$ . Tou je hrana  $\{1, 2\}$  v orientácii  $(1, 2)$ . Pretože vrchol 2 ešte nebol preskúmaný zaznačíme v treťom riadku tabuľky hranu  $\{1, 2\}$  symbolom  $\Rightarrow$  ako hranu prvého príchodu a vrchol 2 značkou • ako preskúmaný. Novým systematickým prehľadaním zoznamu hrán od začiatku zistíme, že prvou zaraditeľnou hranou do sledu  $\mathcal{S}$  je hrana  $\{2, 5\}$  v orientácii  $(2, 5)$ . Keďže vrchol 5 už bol objavený, v ďalšom riadku tabuľky zaznačíme iba smer použitia hrany  $\{2, 5\}$  značkou  $\rightarrow$ . Ďalší, piaty riadok tabuľky zodpovedá zaradeniu hrany  $\{2, 5\}$  v orientácii  $(5, 2)$  do sledu  $\mathcal{S}$ . Pretože táto hrana je zaraďovaná do sledu  $\mathcal{S}$  po

druhýkrát, zaradíme ju aj do eulerovského ťahu. Táto skutočnosť je v prvom stĺpci tabuľky vyjadrená zarámovaním a hrubým fontom.

Takto pokračujeme ďalej a počas práce algoritmu zvlášť označíme hrany konštruovaného sledu v prvom stĺpci, ak boli použité po druhýkrát. Tieto hrany vytvárajú hľadaný eulerovský ťah. V našom prípade má tento ťah tvar

$$5, \{5, 2\}, 2, \{2, 1\}, 1, \{1, 4\}, 4, \{4, 3\}, 3, \{3, 1\}, 1, \{1, 5\}, 5.$$

Pretože značky použitia hrán a navštívenia vrcholov sa v priebehu výpočtu po označení daného objektu ďalej nemenia, dokonca ani pre ručný výpočet nie je potrebná pre ne celá tabuľka, stačí len jeden jej riadok, ak konštruovaný sled zapisujeme zvlášť. Tu je tabuľka uvádzaná pre lepšiu ilustráciu postupu výpočtu, pretože v nej vidieť stav označenia hrán po jednotlivých krokoch.

## 6.2 Úloha čínskeho poštára

Slovná formulácia úlohy čínskeho poštára je nasledujúca: Poštár má vyjsť z pošty, prejsť všetky ulice svojho rajónu a vrátiť sa na poštu tak, aby sa čo najmenej nachodil.

Rajón poštára modelujeme súvislým hranovo ohodnoteným grafom  $G = (V, H, c)$ , v ktorom vrcholy predstavujú poštu a križovatky, hrany ulice. Hrany sú ohodnotené dĺžkou príslušnej ulice. Pochôdzku poštára modelujeme uzavretým eulerovským sledom.

**Matematická formulácia úlohy čínskeho poštára.** V súvislom hranovo ohodnotenom grafe nájsť uzavretý eulerovský sled najmenej dĺžky.

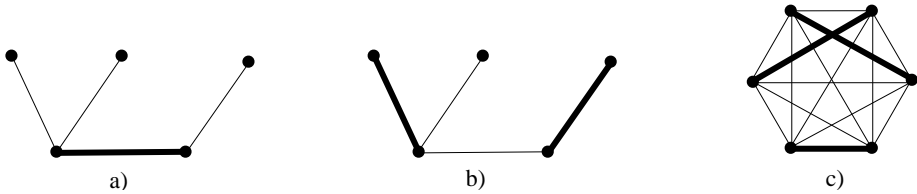
Prvýkrát túto úlohu formuloval a riešil čínsky matematik Kwan (1962), avšak nepodarilo sa mu dotiahnuť riešenie do konca. Exaktné riešenie založené na rozšírení pôvodného grafu o hrany najlacnejšieho párenia úplného grafu z vrcholov nepárneho stupňa, ktoré tu budeme prezentovať, navrhol Edmonds (1965) (a nezávisle od neho Busacker a Saaty v tom istom roku).

Keby graf  $G$  modelujúci rajón poštára obsahoval iba vrcholy párneho stupňa, úloha by bola jednoduchá – stačilo by nájsť uzavretý eulerovský ťah. V reálnom živote je však situácia zložitejšia – križovatky typu T alebo Y sú bežné a vyskytujú sa aj zložitejšie dopravné uzly, ktoré vedú k vrcholom nepárneho stupňa v grafovom modeli.

Počet vrcholov nepárneho stupňa grafu  $G$  je párny. Keby sme "spojili" dva vrcholy  $u, v$  nepárneho stupňa novou fiktívnou hranou, stanú sa z nich vrcholy párneho stupňa. Ak teda zoradíme všetky vrcholy grafu  $G$  nepárneho stupňa do dvojíc a tie "pospájame" fiktívnymi hranami, dostaneme nový graf (vo všeobecnosti multigraf)  $\overline{G}$ , ktorý má už všetky vrcholy párneho stupňa a je v ňom možné zostrojiť uzavretý eulerovský ťah.

Ako však poštar prejde fiktívnu hranu  $\{u, v\}$ ? Pretože sa môže pohybovať len po reálnych uliciach, prejde z križovatky  $u$  na križovatku  $v$  po najkratšej možnej  $u-v$  ceste. Ak teda každú fiktívnu hranu typu  $\{u, v\}$  ohodnotíme vzdialenosťou  $d(u, v)$  vrcholov  $u, v$  v grafe  $G$ , súhrnná dĺžka dodaných fiktívnych hrán bude predstavovať naprázdno prejdenú vzdialenosť pri pochôdzke. Teraz je už vidieť, že ak chceme minimalizovať naprázdno prejdenú vzdialenosť (t. j. splniť požiadavku poštaru "aby sa čo najmenej nachodil"), musíme nájsť také spárovanie dvojíc vrcholov nepárneho stupňa, aby súhrnná dĺžka dodaných fiktívnych hrán bola najmenšia.

**Definícia 6.3.** Nech  $G = (V, H, c)$  je hranovo ohodnotený graf. **Párenie** v grafe  $G$  je taký jeho podgraf  $P$ , v ktorom má každý vrchol stupeň 1. **Cena párenia**  $P$  je súčet ohodnotení jeho hrán. Hovoríme, že párenie  $P$  je **maximálne párenie** v grafe  $G$ , ak  $P$  nie je podgrafom žiadneho iného párenia v  $G$ . Párenie  $P$  je **najpočetnejšie párenie** v grafe  $G$  ak  $P$  má zo všetkých párení najväčší počet hrán. Párenie  $P$  je **úplné párenie** v  $G$ , ak  $P$  je faktorovým podgrafom grafu  $G$  ( $P$  obsahuje všetky vrcholy grafu  $G$ ).



Obr. 6.2: a) Maximálne párenie, ktoré nie je ani najpočetnejšie, ani úplné.  
 b) Najpočetnejšie párenie, ktoré nie je úplné.  
 c) Úplné párenie v  $K_6$ .

Nie každé maximálne párenie musí byť najpočetnejším párením ani úplným párením. Každé najpočetnejšie (resp. úplné) párenie je aj maximálnym párením. Avšak v úplnom grafe  $K_{2t}$  s párnym počtom vrcholov je každé maximálne párenie najpočetnejším i úplným párením.



Úlohu nájdenia optimálnej množiny fiktívnych hrán možno teraz sformulovať nasledovne:

V úplnom hranovo ohodnotenom grafe  $K_{2t}$  nájsť úplné (resp. maximálne, resp. najpočetnejšie) párenie s minimálnou cenou.

Pre práve sformulovanú úlohu existuje polynomiálny algoritmus so zložitou  $O(n^4)$ . Je to však jeden zo zložitejších algoritmov teórie grafov, využívajúci i niektoré poznatky teórie lineárneho programovania, preto ho v tejto učebnici neuvádzam. My budeme hľadať optimálne párenie prezretím všetkých možností. Skúsme vypočítať ich počet v grafe  $K_{2t}$  v závislosti na  $t$ . Prvú dvojicu môžeme vybrať  $\frac{2t \cdot (2t-1)}{2}$  možnosťami, druhú  $\frac{(2t-2) \cdot (2t-3)}{2}$  atď. poslednú  $\frac{2 \cdot 1}{2}$  možnosťami. Pretože nezáleží na poradí dvojíc, všetkých možností je

$$\begin{aligned} p(t) &= \frac{\frac{2t \cdot (2t-1)}{2} \cdot \frac{(2t-2) \cdot (2t-3)}{2} \cdot \frac{(2t-4) \cdot (2t-5)}{2} \cdots \frac{2 \cdot 1}{2}}{t!} = \\ &= \frac{(t \cdot (2t-1)) \cdot ((t-1) \cdot (2t-3)) \cdot ((t-2) \cdot (2t-5)) \cdots (1 \cdot 1)}{t!} = \\ &= (2t-1) \cdot (2t-3) \cdot (2t-5) \cdots 1 \quad (6.1) \end{aligned}$$

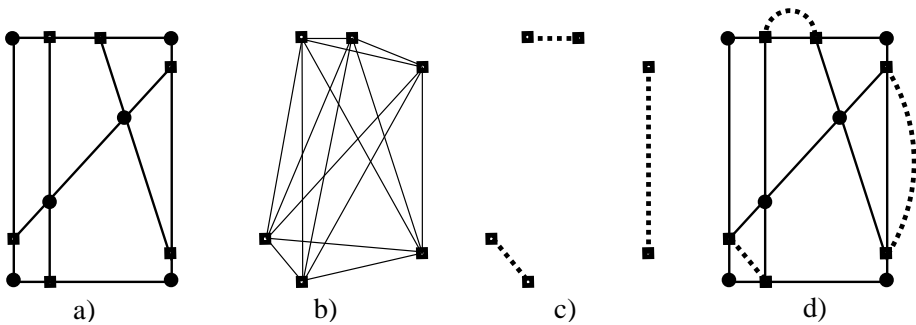
Je možné aj iné vyjadrenie, napr.:

$$p(t) = \frac{(2t)!}{2^t \cdot t!}.$$

**Algoritmus 6.3.** Edmondsov algoritmus na hľadanie najkratšieho uzavretého eulerovského sledu v súvislom hranovo ohodnotenom grafe  $G = (V, H, c)$ .

- **Krok 1.** V grafe  $G$  nájdí všetky vrcholy nepárneho stupňa. Tých je párny počet  $2t$ .  
Z vrcholov nepárneho stupňa zostroj úplný graf  $K_{2t}$ . Jeho hrany ohodnot vzdialenosťami koncových vrcholov hrany v pôvodnom grafe  $G$ .
- **Krok 2.** V grafe  $K_{2t}$  nájdí úplné párenie s minimálnou cenou.
- **Krok 3.** Hrany párenia pridaj k hranovej množine pôvodného grafu  $G$ . Dostaneš tak multigraf  $\overline{G}$ , v ktorom majú všetky vrcholy párny stupeň. V multigrafe  $\overline{G}$  zostroj uzavretý eulerovský ťah  $\mathcal{T}$ .

- **Krok 4.** Hrany párenia v ťahu  $\mathcal{T}$  nahraď príslušnými najkratšími cestami v grafe  $G$  a označ ich ako prejdené naprázdno. Dostaneš tak najkratší eulerovský uzavretý sled v grafe  $G$ .



Obr. 6.3: Postup pri Edmondsovom algoritme.

- Pôvodný graf, vrcholy nepárneho stupňa sú vyznačené štvorčkami.
- Pomocný úplný graf  $K_{2t}$  zostrojený podľa kroku 1. algoritmu 6.3.
- Úplné párenie s minimálnou cenou v  $K_{2t}$ .
- Multigraf  $\overline{G}$  zostrojený podľa kroku 3. algoritmu 6.3, kde už existuje eulerovský uzavretý ťah.

Zložitosť Edmondsovho algoritmu. Úplný graf  $K_{2t}$  môže mať najviac  $n$  vrcholov a najviac  $\frac{n \cdot (n+1)}{2}$  hrán. Pre výpočet ohodnotení hrán grafu  $K_{2t}$  stačí vypočítať maticu vzdialeností  $\mathbf{C}$  vrcholov grafu  $G$ , čo možno použitím Floydovho algoritmu urobiť v čase  $O(n^3)$ . Celý krok 1. možno teda urobiť v čase  $O(n^3)$ .

Pretože  $2t \leq n$ , úplné párenie s minimálnou cenou v grafe  $K_{2t}$  možno urobiť v čase  $O(n^4)$  (lebo najpočetnejšie párenie v grafe s  $n$  vrcholmi možno nájsť v čase  $O(n^4)$ ). Krok 2. vyžaduje čas  $O(n^4)$ .

V kroku 3. sa hľadá uzavretý eulerovský ťah v grafe  $\overline{G}$  čo možno urobiť v čase  $O(n^4)$  – pozri rozbor zložitosti labyrintového algoritmu.

Konečne v kroku 4. treba nahraď každú fiktívnu hranu príslušnou najkratšou cestou. Fiktívnych hrán môže byť najviac  $n/2$ . Ak sme v kroku 1. počítali spolu s maticou vzdialeností aj maticu  $\mathbf{X}$ , možno pomocou smerníkov z tejto matice zrekonštruovať ľubovoľnú najkratšiu  $u-v$  cestu v čase  $O(n)$  (lebo žiadna cesta v grafe  $G$  s  $n$  vrcholmi nemôže mať viac ako  $n - 1$  hrán). Pretože

fiktívnych hrán môže byť najviac  $n/2$ , stačí na ich nahradenie príslušnými najkratšími cestami  $O(n^2)$  krokov. Pre zložitosť celého Edmondsovho algoritmu bude určujúca zložitosť krokov 2. a 3. Edmondsov algoritmus má teda zložitosť  $O(n^4)$ .

### 6.3 Úloha obchodného cestujúceho – TSP

Obchodný cestujúci má navštíviť všetkých svojich zákazníkov a vrátiť sa domov tak, aby sa čo najmenej nachodil. Pochôdzku, ktorá navštívi všetky vrcholy grafu modelujeme tzv. hamiltonovským sledom.

**Definícia 6.4.** Sled v grafe  $G$  sa nazýva **hamiltonovský sled** v grafe  $G$ , ak obsahuje všetky vrcholy grafu  $G$ .

*Poznámka.* Predchádzajúca definícia definuje i hamiltonovskú cestu i hamiltonovský cyklus, pretože obe sú špeciálnym prípadom hamiltonovského sledu.

**Definícia 6.5.** Hovoríme, že graf  $G$  je **hamiltonovský**, ak v ňom existuje hamiltonovský cyklus.

Úlohu obchodného cestujúceho môžeme matematicky modelovať niekoľkými spôsobmi podľa toho, či obchodník smie alebo nesmie počas pochôdzky navštíviť jeden vrchol viackrát – podvodník sa nerád vracia na miesto činu.

Ak dovoľujeme navštíviť to isté miesto viackrát, úlohu obchodného cestujúceho môžeme formulovať nasledovne:

**V súvislom hranovo ohodnotenom grafe nájsť najkratší uzavretý hamiltonovský sled.**

Ak zakazujeme navštíviť to isté miesto viackrát, úlohu obchodného cestujúceho formulujeme takto:

**V súvislom hranovo ohodnotenom grafe nájsť najkratší hamiltonovský cyklus.**

V predchádzajúcej časti sme videli, že existuje jednoduché kritérium na zistenie, či je daný graf  $G$  eulerovský – práve vtedy ak  $G$  je súvislý a má všetky vrcholy párneho stupňa. Hamiltonovské grafy sa takto jednoducho charakterizovať nepodarilo. Máme iba postačujúce podmienky, ktoré zaručujú existenciu hamiltonovského cyklu v grafe, tie sú však veľmi silné. Ako príklad slúžia dve nasledujúce vety.

**Veta 6.2.** *Nech v grafe  $G = (V, H)$  s aspoň troma vrcholmi pre každé dva také vrcholy  $u, v$ , ktoré nie sú susedné, platí  $\deg(u) + \deg(v) \geq |V|$ . Potom je  $G$  hamiltonovský graf.*

**Veta 6.3.** *Nech v grafe  $G = (V, H)$  s aspoň troma vrcholmi platí pre každý vrchol  $v \in V$   $\deg(v) \geq \frac{1}{2} \cdot |V|$ . Potom je  $G$  hamiltonovský graf.*

Vo veľkom množstve praktických situácií hamiltonovský cyklus neexistuje (predstavme si len cestnú sieť na strednom alebo severnom Slovensku, kde vzhľadom na hornatý charakter krajiny cestné komunikácie končia v dolinách a cestná sieť má stromovitý charakter), preto častejšie prezentujeme úlohu obchodného cestujúceho ako hľadanie najkratšieho uzavretého hamiltonovského sledu v súvislom hranovo ohodnotenom grafe  $G = (V, H, c)$ . Na riešenie takejto úlohy zostrojíme k danému grafu  $G$  pomocný úplný hranovo ohodnotený graf  $\overline{G} = (V, \overline{H}, d)$  s rovnakou množinou vrcholov, akú má graf  $G$ , kde  $\overline{H}$  je množina všetkých neusporiadaných dvojíc  $\{u, v\}$  vrcholov z  $V$  takých, že  $u \neq v$  a ohodnotenie  $d(u, v)$  hrany  $\{u, v\}$  je vzdialenosťou vrcholov  $u, v$  v pôvodnom grafe  $G$ .

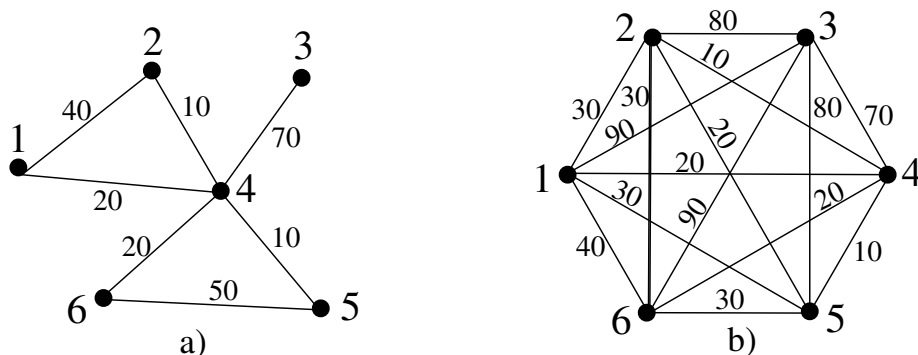
Graf  $\overline{G}$  je vlastne úplný hranovo ohodnotený graf. Ohodnotenie jeho hrán má špeciálnu vlastnosť, ktorú využívajú mnohé algoritmy, a tou je, že funkcia  $d()$  spĺňa trojuholníkovú nerovnosť, t. j.

$$\forall u, v, w \in V \quad u, v, w \text{ po dvoch rôzne, platí: } d(u, v) \leq d(u, w) + d(w, v). \quad (6.2)$$

Lahko možno matematickou indukciou ukázať, že ak v úplnom hranovo ohodnotenom grafe platí pre ohodnotenie hrán trojuholníková nerovnosť, potom pre ľubovoľné  $u \in V, v \in V$  je najkratšou  $u-v$  cestou jednohranová cesta  $u, \{u, v\}, v$  s dĺžkou  $d(u, v)$ . Každá iná  $u-v$  cesta má aspoň takú dĺžku ako ohodnotenie  $d(u, v)$  hrany  $\{u, v\}$ .

V úplnom grafe  $\overline{G}$  už každá permutácia vrcholov definuje hamiltonovský cyklus. Ak nájdeme v  $\overline{G}$  najkratší hamiltonovský cyklus  $\mathcal{H}$ , potom z neho ľahko vytvoríme uzavretý sled  $\mathcal{S}$  v pôvodnom grafe  $G$  tak, že každú hranu  $\{u, v\}$  cyklu  $\mathcal{H}$  nahradíme príslušnou najkratšou cestou  $\underline{\mu}(u, v)$  v grafe  $G$ . Je ľahko vidieť, že dĺžka sledu  $\mathcal{S}$  je rovná dĺžke cyklu  $\mathcal{H}$  v  $\overline{G}$ . Preto sa ďalej budeme venovať hľadaniu najkratšieho hamiltonovského cyklu v úplných grafoch, v ktorých platí trojuholníková nerovnosť.

**Veta 6.4.** *Nájsť najkratší hamiltonovský cyklus je NP-ťažká úloha dokonca aj v úplných grafoch, v ktorých platí trojuholníková nerovnosť.*



Obr. 6.4: V grafe a) hamiltonovský cyklus neexistuje. Keďže nám stačí nájsť hamiltonovský sled, hľadáme ho pomocou hamiltonovského cyklu v úplnom grafe  $\overline{G}$  – b), ktorý má hrany ohodnotené vzdialenosťami v pôvodnom grafe  $G$ .

Exaktný algoritmus pre riešenie úlohy obchodného cestujúceho v úplnom grafe by vyžadoval v podstate prekontrolovanie všetkých možností. Ak má graf  $n$  vrcholov, pri fixácii začiatočného (a súčasne aj koncového) vrchola hamiltonovského cyklu treba prekontrolovať  $(n - 1)!$  možností. Pri grafe s 20 vrcholmi by bolo treba prekontrolovať takmer  $19! \approx 1,22 \cdot 10^{17}$  možností. Ak by sme mali k dispozícii paralelný superpočítač, ktorý každú sekundu prekontroluje miliardu, t. j.  $10^9$  možností, trvalo by mu nájdenie optima asi 38 rokov. Pri 30 vrcholoch by už výpočet vyžadoval  $8,4 \cdot 10^{16}$  rokov. Vek vesmíru od Big Bangu doteraz sa odhaduje asi na  $1,5 \cdot 10^{10}$  rokov. Polovicu výpočtov by sme mohli ušetriť tak, že z permutácií (napr.  $\langle 1, 2, 3, 4, 5 \rangle$ ,  $\langle 1, 5, 4, 3, 2 \rangle$  v  $K_5$ ), ktoré určujú dva rôzne hamiltonovské cykly s tou istou dĺžkou, by sme preskúmali len jednu, nie je to však podstatné zlepšenie.

Ak už nie je možné dosiahnuť exaktné optimum, snažíme sa dostať aspoň "dobré" riešenie – t. j. také, ktorého hodnota kritériálnej funkcie by sa len málo líšila od hodnoty kritériálnej funkcie optimálneho riešenia. Takéto algoritmy nazývame **suboptimálne algoritmy** alebo **heuristiky**. Pre úlohu obchodného cestujúceho existuje veľké množstvo heuristik, niektoré z nich tu uvedieme. Všetky uvedené algoritmy predpokladajú, že  $G$  je úplný graf, v ktorom platí trojuholníková nerovnosť.

**Algoritmus 6.4. Pažravá metóda – Greedy Algorithm. Heuristika na hľadanie suboptimálneho riešenia úlohy obchodného cestujúceho v úplnom grafe  $G = (V, H, c)$  s aspoň tromi vrcholmi a s trojuholníkovou nerovnosťou.**

- **Krok 1.** Začni v ľubovoľnom vrchole a do (budúceho) hamiltonovského cyklu vlož najlacnejšiu hranu incidentnú s týmto vrcholom.
- **Krok 2.** Ak je vybratých  $n - 1$  hrán, uzavri cyklus. STOP
- **Krok 3.** Inak vyber takú najlacnejšiu nevybranú hranu incidentnú s posledným vrcholom doteraz vybranej postupnosti, ktorá nie je incidentná so žiadnym iným vrcholom vybranej postupnosti. GOTO Krok 2.



**Algoritmus 6.5. Metóda zdvojenia kostry. (Kim – 1975). Heuristika na hľadanie suboptimálneho riešenia úlohy obchodného cestujúceho v úplnom grafe  $G = (V, H, c)$  s trojuholníkovou nerovnosťou.**

- **Krok 1.** V grafe  $G$  zostroj najlacnejšiu kostru  $K$ .
- **Krok 2.** V kostre  $K$  zostroj uzavretý sled  $S$ , ktorý obsahuje každú hranu práve dvakrát. (Použi napr. Tarryho algoritmus).
- **Krok 3.** Z uzavretého sledu  $S$  vytvor hamiltonovský cyklus takto: Postupne prechádzaj sledom  $S$  a keď narazíš na taký vrchol, alebo úsek niekoľkých vrcholov za sebou, ktoré sa už v slede vyskytujú, premosti takýto vrchol alebo úsek priamou hranou.



**Veta 6.5.** *Nech  $G = (V, H, c)$  je úplný graf, v ktorom platí trojuholníková nerovnosť. Nech  $c(MZK)$  je dĺžka hamiltonovského cyklu získaného metódou zdvojenia kostry, nech  $c(OPT)$  je dĺžka najkratšieho hamiltonovského cyklu v grafe  $G$ . Potom*

$$\frac{c(MZK)}{c(OPT)} < 2.$$

*Naviac posledný odhad už nemožno zlepšiť – pre každé  $\varepsilon > 0$  existuje taký graf  $G_\varepsilon$ , že preň je  $c(MZK)/c(OPT) > 2 - \varepsilon$ .*

**DÔKAZ.**

Dokážeme len prvú časť vety. Nech  $C_{OPT}$  je optimálny hamiltonovský cyklus

v grafe  $G$  s cenou  $c(OPT)$ . Vylúčme z cyklu  $C_{OPT}$  jeho najcennejšiu hranu. Dostaneme súvislý acyklický faktorový podgraf  $K_C$  grafu  $G$  – teda kostru grafu  $G$ , ktorú platí  $c(K_C) < c(OPT)$ . Keďže  $K$  bola najlacnejšia kostra, platí  $c(K) \leq c(K_C)$  a pre Tarryho sled  $\mathcal{S}$  v  $K$  platí

$$c(\mathcal{S}) = 2 \cdot c(K) < 2 \cdot c(OPT).$$

Keďže hamiltonovský  $C_{MZK}$  cyklus z algoritmu 6.5 vznikol premostením niektorých úsekov sledu  $\mathcal{S}$  a keďže v  $G$  platí trojuholníková nerovnosť, je

$$c(MZK) \leq c(\mathcal{S}) < 2 \cdot c(OPT). \quad (6.3)$$

■

**Algoritmus 6.6. Algoritmus kostry a párenia. (Christofides – 1976.)** Heuristika na hľadanie suboptimálneho riešenia úlohy obchodného cestujúceho v úplnom grafe  $G = (V, H, c)$  s trojuholníkovou nerovnosťou.

- **Krok 1.** V grafe  $G$  zostroj najlacnejšiu kostru  $K$ .
- **Krok 2.** V kostre  $K$  nájdí všetky vrcholy nepárneho stupňa. Tých je  $2t$ .
- **Krok 3.** Z vrcholov nepárneho stupňa zostroj úplný graf  $K_{2t}$ , jeho hrany ohodnoť ohodnoteniami príslušných hrán v pôvodnom grafe  $G$ .
- **Krok 4.** V grafe  $K_{2t}$  nájdí úplné párenie s minimálnou cenou.
- **Krok 5.** Hrany párenia dodaj k hranovej množine najlacnejšej kostry  $K$ . Dostaneš tak graf (multigraf)  $\overline{G}$ , ktorý má všetky vrcholy párneho stupňa.
- **Krok 6.** V grafe (resp. multigrafe)  $\overline{G}$  zostroj uzavretý eulerovský ťah  $\mathcal{T}$ .
- **Krok 7.** Z uzavretého ťahu  $\mathcal{T}$  vytvor hamiltonovský cyklus takto: Postupne prechádzaj ťahom  $\mathcal{T}$  a keď naraziš na taký vrchol, alebo úsek niekoľkých vrcholov za sebou, ktoré sa už v slede vyskytujú, premosti takýto vrchol alebo úsek priamou hranou.

♣

**Veta 6.6.** *Nech  $G = (V, H, c)$  je úplný graf, v ktorom platí trojuholníková nerovnosť. Nech  $c(MKP)$  je dĺžka hamiltonovského cyklu získaného metódou kostry a párenia, nech  $c(OPT)$  je dĺžka najkratšieho hamiltonovského cyklu v grafe  $G$ . Potom*

$$\frac{c(MKP)}{c(OPT)} < \frac{3}{2}.$$

*Naviac posledný odhad už nemožno zlepšiť – pre každé  $\varepsilon > 0$  existuje taký graf  $G_\varepsilon$ , že preň je  $c(MKP)/c(OPT) > 3/2 - \varepsilon$ .*

*Poznámka.* Nie je známy polynomiálny algoritmus  $ALG$ , pre ktorý by bol zaručený lepší pomer  $c(ALG)/c(OPT)$  než  $3/2$ .

Mnohé praktické úlohy vedú k úlohe hľadania najkratšieho hamiltonovského cyklu v úplnom digrafe (predstavme si obchodného cestujúceho v meste s jednosmernými ulicami). Niektoré heuristiky možno upraviť aj pre tento prípad – napr. pažravú metódu 6.4, naopak, algoritmus zdvojenia kostry 6.5 a algoritmus kostry a párenia 6.6 sú vhodné len pre neorientovaný prípad.

## 6.4 Ďalšie heuristiky pre úlohu obchodného cestujúceho

Napriek svojej teoretickej kráse nie sú algoritmy zdvojenia kostry 6.5 alebo kostry a párenia 6.6 v praxi najlepšie. Preto v tejto časti spomeniem niekoľko heuristik, ktoré sú síce výpočtovo náročnejšie (nie sú vhodné pre ručný výpočet), ale sú algoritmicky jednoduché, dajú sa ľahko naprogramovať a často dávajú v praxi uspokojivé riešenie. Sú určené pre úplné grafy, v ktorých platí trojuholníková nerovnosť. Možno ich modifikovať i pre úplné digrafy, čo však už ponechávam na čitateľa.

Tieto heuristiky by sme mohli rozdeliť do dvoch skupín. Vytvárajúce heuristiky skonštruujú nový hamiltonovský cyklus. Zlepšujúce heuristiky sa snažia vylepšiť súčasný hamiltonovský cyklus.

**Algoritmus 6.7.** Vkladacia heuristika na konštrukciu suboptimálneho hamiltonovského cyklu v úplnom grafe  $G = (V, H, c)$  s trojuholníkovou nerovnosťou.

- **Krok 1.** Do cyklu zaraď hranu  $h = \{u, v\}$  s najmenšou cenou. Nájdi vrchol  $w \in V$ , pre ktorý je súčet  $c(u, w) + c(w, v)$  najmenší. Vytvor cyklus  $C = u, \{u, w\}, w, \{w, v\}, v, \{v, u\}, u$ .
- **Krok 2.** Ak cyklus  $C$  obsahuje všetky vrcholy grafu  $G$ , STOP. Inak pokračuj krokom 3.



- **Krok 3.** Pre každú hranu  $h = \{u, v\}$  cyklu  $C$  vypočítaj<sup>2</sup>

$$z(h) = \min\{c(u, w) + c(w, v) - c(u, v) \mid w \in V - C\}. \quad (6.4)$$

Vezmi hranu  $h = \{u, v\}$  s minimálnym  $z(h)$  a  $w$  vrchol, pre ktorý nastalo minimum v (6.4). Vytvor cyklus  $C'$  tak, že nahradíš hranu  $\{u, v\}$  dvojicou hrán  $\{u, w\}$ ,  $\{w, v\}$ . Polož  $C := C'$ . GOTO Krok 2.



*Poznámka.* Algoritmus 6.7 vytvára postupne cykly tak, že do súčasného cyklu vkladná taký vrchol, ktorým ho najmenej predĺži. Je to typická vytvárajúca heuristika.

### Algoritmus 6.8. Algoritmus prehľadávania okolí.

Ku každému riešeniu – hamiltonovskému cyklu  $C$  – definujeme jeho okolie  $\mathcal{O}(C)$  ako množinu hamiltonovských cyklov, ktorú z cyklu  $C$  dostaneme nejakými operáciami. Označme  $c(C)$  cenu hamiltonovského cyklu  $C$ .

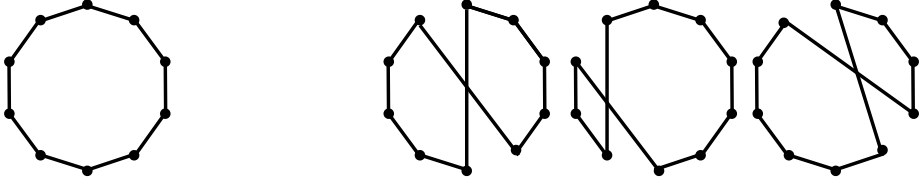
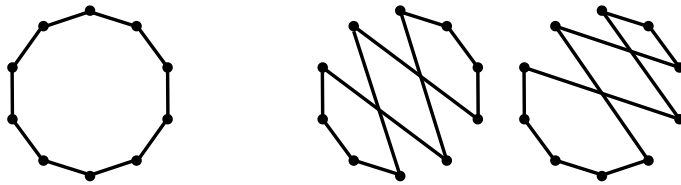
- **Krok 1.** Za počiatočný hamiltonovský cyklus  $C$  vezmi ľubovoľný hamiltonovský cyklus (dostať ho môžeš náhodným generátorom alebo ako výsledok niektorej vytvárajúcej heuristiky).
- **Krok 2.** Hľadaj  $C' \in \mathcal{O}(C)$  také, že  $c(C') < c(C)$ . Ak pre všetky  $C' \in \mathcal{O}(C)$   $c(C') \geq c(C)$ , STOP,  $C$  je suboptimálny hamiltonovský cyklus. Inak pokračuj krokom 3.
- **Krok 3.** Vezmi  $C' \in \mathcal{O}(C)$  také, že  $c(C') < c(C)$  a polož  $C := C'$ . Goto Krok 2.



Ako okolie hamiltonovského cyklu  $C$  môžeme definovať množinu všetkých cyklov, ktoré dostaneme zrušením dvoch hrán cyklu  $C$  a ich nahradením inými hranami. Niekoľko prvkov takto definovaného okolia pôvodného cyklu  $C$  je na obrázku 6.5.

Ďalšou možnosťou je definovať okolie cyklu  $C$  ako množinu všetkých cyklov získaných z cyklu  $C$  zámenou poradia dvoch ľubovoľných vrcholov. Tento spôsob vytvárania okolí ilustruje obrázok 6.6.

<sup>2</sup>Zápisom  $V - C$  vo vzťahu (6.4) sa myslí množina vrcholov z  $V$ , ktoré sa ešte nevyskytujú v cykle  $C$ . Tento zápis nie je z prísneho matematického pohľadu celkom v poriadku, pretože  $V$  je množina a  $C$  sled – postupnosť vrcholov z  $V$  a hrán z  $H$ . Presný zápis by mohol byť napr. v tvare „ $V - V_C$ “, kde  $V_C$  je množina všetkých vrcholov cyklu  $C$ “.

Obr. 6.5: Cyklus  $C$  a niekoľko prvkov jeho okolia.Obr. 6.6: Iný spôsob definovania prvkov okolia cyklu  $C$ .

Zovšeobecnením predchádzajúcej možnosti, ako definovať okolie cyklu  $C$  je táto: Vyber  $U$  časť cyklu  $C$  s tromi až siedmimi vrcholmi a za okolie cyklu  $C$  prehlás všetky cykly, ktoré dostaneš nahradením časti  $U$  ľubovoľnou permutáciou prvkov z  $U$ .

## 6.5 Aplikácie

Samotná úloha čínskeho poštára je aplikačne motivovaná. Možno na ňu previesť množstvo praktických úloh spojených s optimálnym prechádzaním všetkých hrán daného grafu. Pri zimnej údržbe ciest treba najprv odhrnúť sneh zo všetkých úsekov cestnej siete. Potom treba posypať všetky cestné úseky chemickým posypom alebo protišmykovým materiálom. Trasy odhŕňacích resp. posypacích vozidiel možno optimalizovať použitím úlohy čínskeho poštára. Na rovnaký problém vedie i úloha optimálneho trasovania smetiarskych vozidiel pri zbere domového odpadu po jednotlivých uliciach.

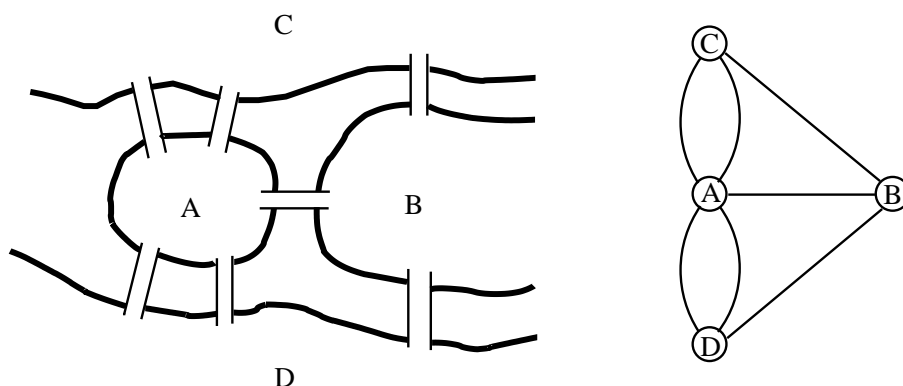
Podobne možno nájsť priame aplikácie pre úlohu obchodného cestujúceho – ako nájsť optimálnu trasu pre vozidlo rozvážajúce tovar do určených predajní, či tlač do novinových stánkov, ako optimálne určiť trasu opravára, ktorý má

navštíviť niekoľko firiem. Rovnakého charakteru je i úloha, ako optimálne autobusom porozvážať maloletých žiakov domov po neskorom návrate zo zázjazdu.

V ďalšej časti tejto kapitoly uvedieme niekoľko menej zrejmych aplikácií.

### 6.5.1 Sedem mostov mesta Königsberg

V meste Königsberg<sup>3</sup> rieka Pregel vytvárala v 18. storočí situáciu, ktorú ukazuje obrázok 6.7. Časť mesta bola na ostrove a medzi dvoma ramenami rieky.



Obr. 6.7: Sedem mostov v meste Königsberg a multigraf  $G$  modelujúci situáciu.

Cez rieku viedlo sedem mostov. Medzi obyvateľmi sa rozšírila zábavná úloha prejsť všetkými siedmimi mostami práve raz a vrátiť sa do východzieho miesta. Na riešenie úlohy sa dokonca uzavierali stávky. Pruský vládca Frederik Veľký problém predložil slávnemu švajčiarskemu matematikovi Leonhardovi Eulerovi, ktorý ukázal, že prejsť všetkými siedmimi mostami mesta nie je možné. Riešenie publikoval v roku 1736. Situáciu modeloval multigrafom  $G$  podľa obrázka 6.7. Pretože všetky vrcholy  $G$  majú nepárny stupeň, žiaden z nich nemôže byť vnútorným vrcholom trasy požadovaných vlastností.

<sup>3</sup>Königsberg, po roku 1945 Kaliningrad – pôvodne Pruské mesto založené roku 1255, rodné mesto Immanuela Kanta. Leží 280 km severne od Varšavy. Po roku 1945 súčasť Sovietskeho Zväzu. Oblasť Kaliningradu (menšia ako  $200 \times 100$  km) patrí v súčasnosti Rusku, je však od neho oddelená pobaltskými republikami.

V modernej terminológii teórie grafov sa problém siedmich mostov mesta Kaliningradu dá formulovať ako problém existencie eulerovského ťahu v multi-grafe  $G$ .

### 6.5.2 Plánovanie športových stretnutí

Stolnotenisoví hráči  $A, B, C, D, E, F$  majú zohrať v jednej herni s jedným stolom stretnutia podľa tabuľky z obrázku 6.8.

	A	B	C	D	E	F
A	–	1	0	1	1	0
B	1	–	1	1	0	1
C	0	1	–	0	1	0
D	1	1	0	–	1	1
E	1	0	1	1	–	1
F	0	1	0	1	1	–

Obr. 6.8: Tabuľka stretnutí, ktoré treba odohrať, a príslušný grafový model.

Bolo by výhodné, keby sa podarilo zostaviť taký plán stretnutí, aby každý hráč odohral dve stretnutia po sebe (nemusí meniť šatňu, stačí mu jedna doprava na dva zápasy atď.). Hráč by však nemal odohrať viac ako dve stretnutia za sebou (pri treťom by mohol mať súper výhodu unaveného protivráča). Je možné napláňovať stretnutia podľa uvedených požiadaviek?

Grafový model problému bude graf  $G$ , ktorého vrcholy sú hráči a ktorého hrany sú stretnutia, ktoré treba odohrať. Ľubovoľný ťah v grafe  $G$  predstavuje postupnosť stretnutí takú, že každý hráč okrem posledného v tomto ťahu odohrá dve stretnutia za sebou. Napláňovať stretnutia tak, aby každý hráč odohral za sebou dve stretnutia znamená nájsť v grafe  $G$  eulerovský ťah (nemusí byť nutne uzavretý).

Ak graf  $G$  obsahuje viac ako dva vrcholy nepárneho stupňa, formulovaná úloha nie je riešiteľná. V tom prípade musíme zľaviť s pôvodnej požiadavky a formulovať úlohu nasledovne: Nájsť taký plán stretnutí, ktorý minimalizuje počet prípadov, kedy hráč odohrá len jeden zápas. (Ak je  $2t$  počet vrcholov

nepárneho stupňa v grafe  $G$ , potom nastane minimálne  $t - 1$  takýchto prípadov.) Dodaním  $t$  fiktívnych hrán doplníme graf  $G$  na eulerovský graf, v ktorom je eulerovský ťah hľadaným riešením.

### 6.5.3 Riadenie súradnicového zapisovača

Napriek veľkému rozvoju laserových a atramentových tlačiarň sa ešte stále na počítačové kreslenie technických výkresov (najmä väčšieho rozmeru) používajú súradnicové zapisovače nazývané tiež plottery. Tie kreslia tak, že nad papierom sa pohybuje písací hrot, ktorý môže byť v dvoch polohách: spustený a zdvihnutý. Keď je hrot spustený, dotýka sa papiera a zapisuje súvislú čiaru. Zdvihnutá poloha hrotu sa používa na jeho presun do začiatku zápisu ďalšej čiary. Každú čiaru výkresu treba nakresliť iba raz, viacnásobné prejdenie písacieho hrotu po tej istej čiare by bolo totiž na výkrese viditeľné ako zosilnenie hrúbky tejto čiary.

Ak chceme, aby plotter nakreslil výkres čo najskôr, treba zariadiť, aby robil čo najmenej presunov zdvihnutého hrotu. Ak množinu čiar výkresu považujeme za hrany grafu  $G$  a ich koncové body za vrcholy grafu  $G$ , úlohu optimálneho riadenia súradnicového zapisovača môžeme formulovať ako úlohu čínskeho poštára v tomto grafe s tým, že za dĺžku (ohodnotenie) každej hrany v príslušnom pomocnom grafe  $K_{2t}$  vezmeme euklidovskú vzdialenosť jej koncových vrcholov. Páriaca hrana vo výslednom eulerovskom ťahu v príslušnom grafe  $\bar{G}$  bude znamenať pohyb zdvihnutého hrotu najkratším smerom do druhého vrchola páriacej hrany.

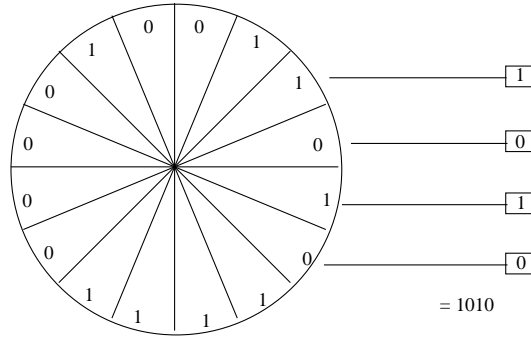
### 6.5.4 DeBruijnovské postupnosti

**Definícia 6.6.** Cyklická postupnosť bitov dĺžky  $2^n$  sa volá  **$(2, n)$ -deBruijnovská postupnosť**, ak každá z  $2^n$  binárnych postupností dĺžky  $n$  sa v nej vyskytne ako jej úsek práve raz.

Príklady  $(2, n)$ -deBruijnovských postupností pre  $n = 1, 2, 3, 4$  sú

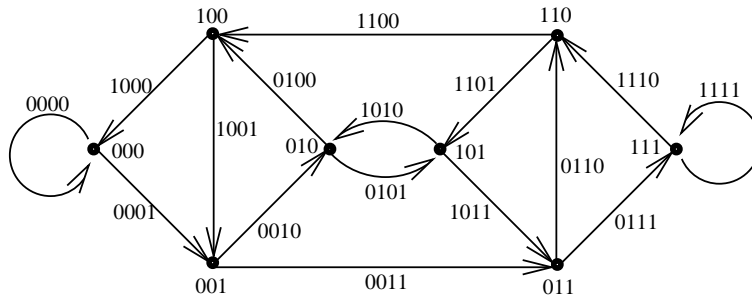
01 0110 01110100 0000100110101111

Predpokladajme, že rotujúci bubon má 16 sektorov a 4 senzory. Chceli by sme označiť jednotlivé sektory bubna tak, aby sme z obsahu štyroch senzorov



Obr. 6.9: Rotujúci bubon so 16 sektormi a štyrmi senzormi.

dokázali identifikovať jednu zo šesťnástich možných polôh bubna. Sektory môžeme označiť iba dvoma spôsobmi (vodivým alebo nevodivým materiálom, svetlo odrazivým alebo pohltivým povrchom atď.). Označenie sektora môže teda byť iba 1 alebo 0. Obsah štyroch sensorov môže definovať 16 rôznych polôh. Polohu bubna budeme však môcť jednoznačne identifikovať iba vtedy, ak sa každé štvormiestne binárne číslo vyskytne ako úsek cyklickej postupnosti značiek sektorov práve raz, t. j. ak značky sektorov tvoria  $(2, 4)$ -deBruijnovskú postupnosť. Vzniká teda otázka, či pre dané  $n = 4$  vôbec  $(2, 4)$ -deBruijnovská postupnosť existuje a ak áno, ako ju skonštruovať.

Obr. 6.10:  $(2, 4)$ -deBruijnovský pseudodigraf.

**Definícia 6.7.**  $(2, n)$ -deBruijnovský pseudodigraf je pseudodigraf  $\vec{D}_{2,n} = (V, H)$ , ktorý má za množinu vrcholov množinu všetkých  $(n - 1)$ -miestnych

binárnych čísel a za množinu všetkých hrán všetky usporiadané dvojice vrcholov typu

$$h = (b_1b_2 \dots b_{n-1}, b_2b_3 \dots b_n). \quad (6.5)$$

Hrane typu (6.5) priradíme značku  $b_1b_2 \dots b_{n-1}b_n$ .

Ukazuje sa, že každý deBruijnovský pseudodigraf  $\vec{D}_{2,n}$  je silne súvislý a keďže vstupný stupeň každého vrchola sa rovná jeho výstupnému stupňu, existuje v každom pseudodigrafe  $\vec{D}_{2,n}$  uzavretý eulerovský orientovaný ťah. Ak však nájdeme v pseudodigrafe  $\vec{D}_{2,n}$  uzavretý eulerovský orientovaný ťah, tento nám už definuje hľadanú  $(2, n)$ -deBruijnovskú postupnosť.

Okrem uvedenej aplikácie na identifikáciu polohy rotujúceho bubna, existuje viacero aplikácií deBruijnovských postupností v teórii kódovania a v kryptografii.

### 6.5.5 Miešačka farieb

V stroji na výrobu farieb sa vyrábajú farby Biela, Žltá, Modrá, Červená. Po ukončení výroby jednej farby sa musí stroj očistiť od zvyškov predchádzajúcej farby. Zvyšky žltej farby neovplyvnia tak výrobu červenej ako zvyšky modrej výrobu bielej. Preto dôkladnosť čistenia stroja – a teda aj nastavovacia doba – závisí od predchádzajúcej a nasledujúcej farby. Nastavovacie doby možno zadať vo forme tabuľky

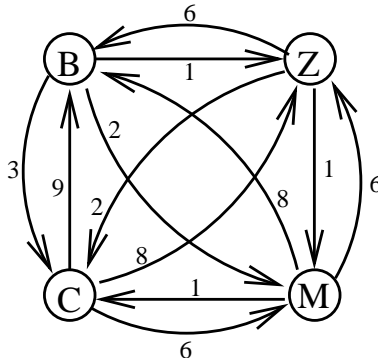
	B	Ž	M	Č
B	0	1	2	3
Ž	6	0	1	2
M	8	6	0	1
Č	9	8	6	0

Existuje  $(4 - 1)! = 6$  rôznych výrobných cyklov, ktoré majú podľa tabuľky všeobecne rôzne sumárne nastavovacie doby:

1. B – Ž – Č – M – B     $1+2+6+8=17$
2. B – Ž – M – Č – B     $1+1+1+9=12$
3. B – Č – Ž – M – B     $3+8+1+8=20$

4. B – Č – M – Ž – B     $3+6+6+6=21$
5. B – M – Ž – Č – B     $2+6+2+9=19$
6. B – M – Č – Ž – B     $2+1+8+6=17$

K tejto úlohe si môžeme zostrojiť úplný digraf  $\vec{G} = (V, H, c)$ , kde  $V$  je množina farieb a ohodnotenie orientovanej hrany  $(i, j)$  znamená dobu čistenia stroja, keď po farbe  $i$  budeme vyrábať farbu  $j$ .



Obr. 6.11: Digraf k predchádzajúcej tabuľke.

Teraz už možno úlohu hľadania optimálneho poradia farieb preformulovať na úlohu hľadania najkratšieho hamiltonovského cyklu v digrafe  $G$ .

Tento princíp môžeme použiť pri určovaní optimálneho poradia spracovania operácií na jednom stroji, ak sú dané medzioperačné nastavovacie časy.

### 6.5.6 Optimálne poradie fáz v svetelne riadenej križovatke

Časť 8.5.5 na str. 238 pojednáva o riadení cestnej križovatky svetelným signalizačným zariadením. Pri použití techniky fázových skupín sa prúdy zoradia do niekoľkých fáz, ktoré potom postupne v istom poradí dostávajú zelenú.

Predpokladajme, že už máme dané zadelenie prúdov do fáz. Fáza  $F_j$  nemôže dostať zelenú v okamihu skončenia fázy  $F_i$ , ale až po uplynutí medzičasu  $M_{ij}$  daného maximom z medzičasov  $m_{kl}$ , kde  $k \in F_i$  a  $l \in F_j$ . Súčet medzičasov počas jedného cyklu bude teda závisieť od poradia fáz. Našou úlohou je určiť cyklické poradie fáz tak, aby súčet medzičasov medzi fázami bol čo najmenší.



Ako model teórie grafov pre riešenie tejto úlohy použijeme úplný digraf  $\vec{G} = (V, H, c)$ , ktorého vrcholmi budú všetky fázy, a pre ohodnotenie hrany  $(F_i, F_j)$  bude platiť  $c((F_i, F_j)) = M_{ij}$ . Potom úloha nájsť optimálne poradie fáz je ekvivalentná s úlohou obchodného cestujúceho v digrafe  $\vec{G}$ . Počet fáz v signálnom pláne bežnej križovatky býva malý, a tak je možné riešiť túto úlohu prezretím všetkých možností.

### 6.5.7 Grayov kód – Gray Code

Majme 8-bitový binárny kód, ktorým v dvojkovej sústave kódujeme čísla od 0 do 255. Týmto kódom chceme prenášať číselné hodnoty – napríklad hodnoty farby obrazového bodu, alebo namerané hodnoty intenzity elektrického poľa, žiarenia, teploty a podobne.

Ak správu zakódovanú takýmto kódom prenášame cez kanál so šumom, môžu pri prenose vzniknúť chyby. Ak sa zmení hodnota najnižšieho bitu prenášaného 8-bitového slova, prenášaná hodnota sa veľmi nezmení. Ak sa však zmení najvyšší bit, dostávame diametrálne odlišný výsledok. Vzniká otázka, či je možné nájsť také kódovanie, pri ktorom by zmena jedného bitu neznamenal veľkú zmenu prenášanej hodnoty.

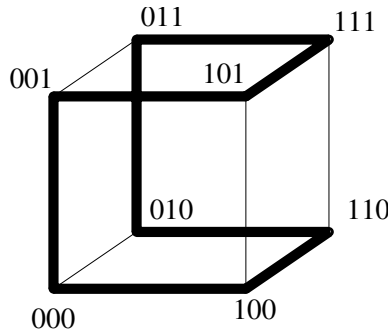
Jedným z možných riešení je tzv. Grayov kód, nazvaný podľa Franka Graya, výskumníka AT&T Bell Laboratories, ktorý síce tento kód nevynašiel, ale pracoval na mnohých jeho explikáciách.

**Definícia 6.8. Grayov kód – Gray Code rádu  $n$**  je taká postupnosť  $n$ -miestnych binárnych čísel, že každé dva po sebe idúce členy tejto postupnosti sa líšia iba v jednom bite.

Zostrojme graf  $Q_n = (V, H)$ , kde množina  $V$  je množinou všetkých  $n$ -prvkových slov v abecede  $B = \{0, 1\}$  (t. j.  $n$ -prvkových postupností núl a jedničiek). Vrcholy  $\mathbf{v} = (v_1, v_2, \dots, v_n) \in V$ ,  $\mathbf{w} = (w_1, w_2, \dots, w_n) \in V$  tvoria neorientovanú hranu množiny  $H$  práve vtedy, keď sa slová  $\mathbf{v}$ ,  $\mathbf{w}$  líšia len na jednom mieste. Takto definovaný graf  $Q_n$  sa niekedy volá aj  **$n$ -rozmerná kocka**.

Úloha nájsť Grayov kód rádu  $n$  je úlohou nájsť v grafe  $Q_n$  ľubovoľný hamiltonovský cyklus. Predpokladajme, že postupnosť vrcholov  $b_1, b_2, \dots, b_{2^n}, b_1$  definuje hamiltonovský cyklus v  $n$ -rozmernej kocke  $Q_n$ . Potom postupnosť vrcholov

$$b_1 0, b_2 0, \dots, b_{2^n} 0, b_{2^n} 1, b_{2^n-1} 1, \dots, b_1 1, b_1 0 \quad (6.6)$$



Obr. 6.12: Hamiltonovský cyklus v 3-rozmernej kocke.

definuje hamiltonovský cyklus v  $n + 1$  rozmernej kocke  $Q_{n+1}$ . V dôsledku toho existuje Grayov kód rádu  $n$  pre ľubovoľné  $n \geq 1$ .

### 6.5.8 Vrtanie otvorov na plošných spojoch

Elektronické prístroje sa montujú na tzv. dosky plošných spojov. Sú to dosky z izolačného materiálu, na ktorých sú z medenej fólie vytvorené elektrické spoje. Do každej takejto dosky treba vyvrtáť množstvo otvorov pre privody súčiastok.

Otvory sa vrtajú na automatickej vyvrtávačke. Treba určiť, v akom poradí sa majú otvory vrtáť tak, aby celkové vzájomné posuny dosky a vrtáku boli čo najmenšie.

Problém môžeme modelovať na úplnom grafe  $G = (V, H, c)$ , ktorého vrcholy sú otvory a ohodnotenie hrany medzi ľubovoľnými dvoma vrcholmi je vzdialenosť príslušných otvorov na doske. Určiť optimálne poradie vyvrtávania otvorov znamená nájsť najkratší hamiltonovský cyklus v grafe  $G$ . Pretože vzdialenosti vrcholov v grafe  $G$  sú euklidovské vzdialenosti otvorov na doske, ohodnotenia hrán v  $G$  spĺňajú trojuholníkovú nerovnosť.

Problémy podobného charakteru môžu vznikáť pri určovaní poradia operácií na viacúčelových strojoch, pri osadzovaní dosiek súčiastkami atď.

### 6.5.9 Švajčiarsky systém šachového turnaja

Nasledujúca aplikácia je ukázkou použitia úlohy o najlacnejšom maximálnom párení. Táto úloha bola súčasťou riešenia úlohy čínskeho poštára.

Mnohé najmä krátkodobé športové turnaje sa hrajú vylučovacím spôsobom. Pri tomto spôsobe sa hráči alebo kolektívy stretnú vo dvojiciach v prvom kole, víťazi prvého kola postupujú do druhého kola atď. až nakoniec postúpia poslední dvaja neporazení do finále, z ktorého vyjde víťaz. Jedným z dôvodov tohto dosť krutého spôsobu je, že usporiadanie jedného zápasu je materiálovo, časovo i finančne náročné.

Pre šachové stretnutia sú náklady na usporiadanie partie nepomerne nižšie, preto navrhovatelia tzv. švajčiarskeho systému šachového turnaja chceli dopriať každému účastníkovi turnaja približne rovnaký počet príležitostí. Pre veľký počet účastníkov turnaja však systém každý s každým nie je možný.

V švajčiarskom systéme nasadzovania hráčov sa hrá predom určený počet kôl. V každom kole sú hráči nasadení do dvojíc podľa výsledkov predchádzajúcich kôl. V každom kole musí byť splnených niekoľko podmienok:

1. Žiadni dvaja účastníci sa nesmú v priebehu turnaja stretnúť dvakrát.
2. Počet partií odohraných s bielymi a čiernymi figúrkami má byť pre každého hráča približne rovnaký.
3. Žiaden hráč nesmie odohrať viac než dve partie za sebou s tou istou farbou figúr.
4. Rozdiel výkonnosti (aktuálnych postavení hráčov v turnaji) proti sebe nasadených hráčov má byť čo najmenší.
5. Pri nepárnom počte účastníkov turnaja žiaden hráč nesmie pauzovať viac než jedno kolo.

Ručné nasadzovanie hráčov do ďalšieho kola podľa týchto pravidiel je veľmi náročné najmä vo vyšších kolách, keď pribúda obmedzení. Je možné švajčiarsky systém nasadzovania hráčov algoritmizovať?

Ako model pre tento problém bude slúžiť úplný graf  $K_n$ , ktorého vrcholy budú hráči. Cenu  $c(h)$  hrany  $h = \{i, j\}$  grafu  $K_n$  určíme ako mieru porušenia pravidiel švajčiarskeho systému, ak by sa mali hráči  $i, j$  stretnúť medzi sebou v nasledujúcom kole. Optimálne nasadenie hráčov v ďalšom kole zistíme ako maximálne párenie s minimálnou cenou v úplnom grafe  $K_n$  s cenou hran  $c()$ .

## 6.6 Cvičenia

1. Ukážte, že v silne súvislom digrafe  $\vec{G} = (V, H)$  existuje uzavretý eulero­vský orientovaný ťah práve vtedy, keď pre každý vrchol  $v \in V$  platí  $ideg(v) = odeg(v)$ . Návod: Prispôsobte dôkaz vety 6.1 zo strany 162.
2. Na základe predchádzajúceho cvičenia navrhните algoritmus na hľadanie uzavretého euklidovského orientovaného ťahu v euklidovskom digrafe.
3. Ako sa zmení postup hľadania uzavretého euklidovského ťahu v multigrafe, multidigrafe, pseudografe a pseudodigrafe?
4. Môže existovať v grafe, ktorý má most, hamiltonovský cyklus? Existuje v grafe s artikuláciou hamiltonovský cyklus? Je každý eulerovský graf aj hamiltonovským grafom a naopak?
5. Určte nutnú a postačujúcu podmienku pre existenciu hamiltonovského cyklu v úplnom bipartitnom grafe  $K_{m,n}$ . Formulujte nutnú podmienku pre existenciu hamiltonovskej kružnice pre bipartitné grafy.
6. V rohu šachovnice o rozmeroch  $7 \times 7$  políčok stojí kôň. Je možné nájsť takú jeho trasu, aby prešiel každým políčkom práve raz a vrátil sa do východzieho miesta?
7. Vyriešte úlohu čínskeho poštára v  $n$ -rozmernej kocke  $Q_n$ .

### Počítačové cvičenia

8. Naprogramujte niektorú heuristiku pre úlohu obchodného cestujúceho.
9. Navrhните a naprogramujte heuristický algoritmus pre hľadanie úplného párenia s minimálnou cenou v úplnom grafe typu  $K_{2t}$ .
10. S použitím výsledku predchádzajúceho cvičenia napíšte program pre úlohu čínskeho poštára.

# Kapitola 7

## Toky v sieťach

### 7.1 Siete a toky v sieťach

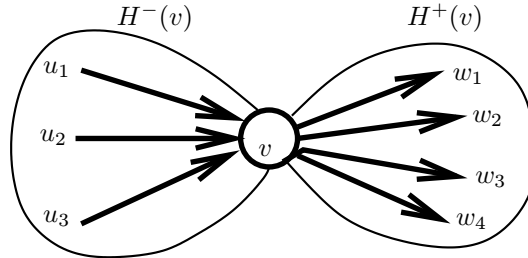
**Definícia 7.1.** Sieťou nazveme neorientované súvislý hranovo ohodnotený digraf  $\vec{G} = (V, H, c)$ , v ktorom ohodnotenie  $c(h) > 0$  každej hrany  $h \in H$  je celočíselné a predstavuje priepustnosť hrany  $h$ , a v ktorom existuje práve jeden vrchol  $z$  taký, že  $\text{iddeg}(z) = 0$  a práve jeden vrchol taký  $u$ , že  $\text{odeg}(u) = 0$ . Vrchol  $z$  nazveme **zdroj**, a vrchol  $u$  nazveme **ústie**.

V definícii 1.18 na strane 23 bolo zavedené nasledujúce označenie: Pre každý vrchol  $v \in V$  digrafu  $\vec{G} = (V, H, c)$  je  $H^+(v)$  množina všetkých hrán z vrchola  $v$  vychádzajúcich a  $H^-(v)$  množina všetkých hrán do vrchola  $v$  vchádzajúcich.  $H^+(v)$  je vlastne množina všetkých hrán výstupnej hviezdy vrchola  $v$  a  $H^-(v)$  je množina všetkých hrán vstupnej hviezdy vrchola  $v$ . Pre množiny  $H^+(v)$ ,  $H^-(v)$  platí:

$$\begin{aligned}H^-(v) &= \{(u, j) \mid j = v, (u, j) \in H\}, \\H^+(v) &= \{(i, w) \mid i = v, (i, w) \in H\}.\end{aligned}$$

Toto označenie budeme v tejto kapitole často využívať bez ďalšieho komentára.

Hrany siete si môžeme predstaviť ako jednosmerné potrubia, priepustnosť (kapacita) hrany predstavuje maximum prietoku, ktoré hranou môže tiecť.



Obr. 7.1: Množina  $H^-(v) = \{(u_1, v), (u_2, v), (u_3, v)\}$   
 a množina  $H^+(v) = \{(v, w_1), (v, w_2), (v, w_3), (v, w_4)\}$

Dopravná interpretácia si hrany siete predstavuje ako jednosmerné cestné úseky s príslušnou kapacitou. Ďalšie interpretácie hrán môžu byť úseky počítačovej siete, úseky rozvodnej elektrickej siete atď.

Zo zdroja do ústia siete tečie nejaký substrát (kvapalina, vozidlá, dáta, elektrický prúd). Tento sa rozloží po hranách siete. Hovoríme, že sieťou tečie nejaký tok. Ten je úplne charakterizovaný prietokmi cez jednotlivé hrany. Na popis toku v sieti nám stačí, keď pre každú hranu  $h \in H$  zadáme množstvo substrátu  $\mathbf{y}(h)$  tečúce touto hranou.

Tok bude teda funkcia  $\mathbf{y} : H \rightarrow \mathbb{R}$ , splňujúca isté požiadavky. V prvom rade budeme požadovať, aby tok bol celočíselná funkcia (podobne ako sme v definícii siete požadovali celočíselnosť kapacít). Neceločíselnosť kapacít hrán a hranových prietokov totiž so sebou prináša niektoré komplikácie, ktorým sa takto vyhneme. V praxi môžeme vhodnou voľbou merných jednotiek vždy dôjsť k celočíselnej formulácii úlohy bez toho, aby sme sa dopustili veľkých nepresností.

Ďalej žiadame, aby množstvo substrátu do vrchola vchádzajúceho sa rovnalo množstvu substrátu z vrchola vychádzajúceho pre každý vrchol mimo zdroj a ústie (analógia 1. Kirchhoffovho zákona). A nakoniec chceme, aby množstvo substrátu zo zdroja vytekajúceho sa rovnalo množstvu substrátu do ústia vchádzajúceho. Na základe tohto rozboru môžeme formulovať nasledujúcu definíciu:

**Definícia 7.2.** Tokom v sieti  $\vec{G} = (V, H, c)$  nazveme celočíselnú funkciu  $\mathbf{y} : H \rightarrow \mathbb{R}$  definovanú na množine orientovaných hrán  $H$ , pre ktorú platí:

$$1. \quad \mathbf{y}(h) \geq 0 \quad \text{pre všetky } h \in H \quad (7.1)$$

$$2. \quad \mathbf{y}(h) \leq c(h) \quad \text{pre všetky } h \in H \quad (7.2)$$

$$3. \quad \sum_{h \in H^+(v)} \mathbf{y}(h) = \sum_{h \in H^-(v)} \mathbf{y}(h) \quad \text{pre všetky také } v \in V, \text{ že } v \neq u, v \neq z \quad (7.3)$$

$$4. \quad \sum_{h \in H^+(z)} \mathbf{y}(h) = \sum_{h \in H^-(u)} \mathbf{y}(h) \quad (7.4)$$

**Velkosťou** toku  $\mathbf{y}$  nazveme číslo  $F(\mathbf{y}) = \sum_{h \in H^+(z)} \mathbf{y}(h)$  (ktoré sa rovná  $\sum_{h \in H^-(u)} \mathbf{y}(h)$ ). Hovoríme, že tok  $\mathbf{y}$  v sieti  $\vec{G}$  je **maximálny**, ak má najväčšiu veľkosť zo všetkých možných tokov v sieti  $\vec{G}$ . Orientovanú hranu  $h \in H$  nazveme **nasýtenou**, ak  $\mathbf{y}(h) = c(h)$ .

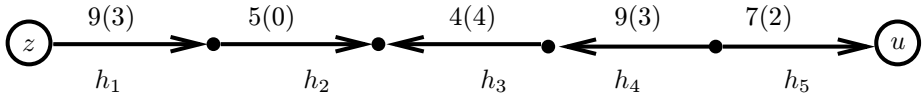
Tok v sieti je teda reálna funkcia  $\mathbf{y} : H \rightarrow \mathbb{R}$  definovaná na množine všetkých hrán. Číslo  $\mathbf{y}(h)$  je funkčná hodnota funkcie  $\mathbf{y}$  v jednom prvku  $h$  svojho definičného oboru (porovnaj  $\mathbf{y}$  a  $\mathbf{y}(h)$  s dvojicou pojmov funkcia  $\log$  a  $\log(2)$ ) a budeme ho volať **tok hranou**  $h$ . Poznamenajme ešte, že tok  $\mathbf{y}$  v sieti  $\vec{G}$  je vlastne ďalšie hranové ohodnotenie, takže sieť  $\vec{G}$  s tokom  $\mathbf{y}$  môžeme považovať za digraf  $\vec{G} = (V, H, c, \mathbf{y})$  s dvomi ohodnoteniami hrán.

## 7.2 Rezervná a zväčšujúca polocesta

**Definícia 7.3.** Nech  $\vec{G} = (V, H, c)$  je sieť s tokom  $\mathbf{y}$ , nech  $v, w \in V$ . Nech  $\mu(v, w)$  je  $v$ - $w$  polocesta, nech  $h$  je orientovaná hrana tejto polocesty. Definujeme  $r(h)$  **rezervu hrany** v poloceste  $\mu(v, w)$  nasledovne:

$$r(h) = \begin{cases} c(h) - \mathbf{y}(h) & \text{ak je hrana } h \text{ použitá v } \mu(v, w) \\ & \text{v smere orientácie} \\ \mathbf{y}(h) & \text{ak je hrana } h \text{ použitá v } \mu(v, w) \\ & \text{proti smeru orientácie} \end{cases} \quad (7.5)$$

**Rezerva polocesty**  $\mu(v, w)$  je minimum rezerv hrán tejto polocesty. Hovoríme, že polocesta  $\mu(v, w)$  je **rezervná polocesta** ak má kladnú rezervu.



Obr. 7.2: Rezervná polocesta.

Ohodnotenie  $9(3)$  hrany  $h_1$  znamená, že  $c(h_1) = 9$ ,  $\mathbf{y}(h_1) = 3$ .

Je  $r(h_1) = 9 - 3 = 6$ ,  $r(h_2) = 5 - 0 = 5$ ,  $r(h_3) = 4$ ,  $r(h_4) = 3$ ,  $r(h_5) = 7 - 2 = 5$ .

Rezerva polocesty je  $\min\{6, 5, 4, 3, 5\} = 3$ .

Rezerva  $r(\boldsymbol{\mu}(v, w))$  polocesty  $\boldsymbol{\mu}(v, w)$  závisí na toku  $\mathbf{y}$ , ktorý už v sieti je, preto by sme mali písať  $r(\boldsymbol{\mu}(v, w), \mathbf{y})$  a hovoriť o rezerve polocesty  $\boldsymbol{\mu}(v, w)$  v sieti  $\vec{G}$  s tokom  $\mathbf{y}$ . Rezerva  $r(h)$  hrany  $h$  závisí nielen od existujúceho toku  $\mathbf{y}$  v sieti  $\vec{G}$ , ale aj od  $v-w$  polocesty, ktorú uvažujeme. Dá sa vymyslieť príklad, keď jedna hrana môže byť v jednej  $v-w$  poloceste použitá v smere orientácie, v inej proti smeru orientácie. Zvláštny význam má rezervná  $z-u$  polocesta, ktorú voláme aj **zvčšujúca polocesta**, ako ukazuje nasledujúce tvrdenie.

**Veta 7.1.** *Nech v sieti  $\vec{G} = (V, H, c)$  s tokom  $\mathbf{y}$  existuje zvčšujúca polocesta. Potom tok  $\mathbf{y}$  nie je maximálny.*

DŮKAZ.

Nech  $\boldsymbol{\mu}(z, u)$  je rezervná  $z-u$  polocesta zo zdroja do ústia s rezervou  $r$ . Definujme tok  $\mathbf{y}'$

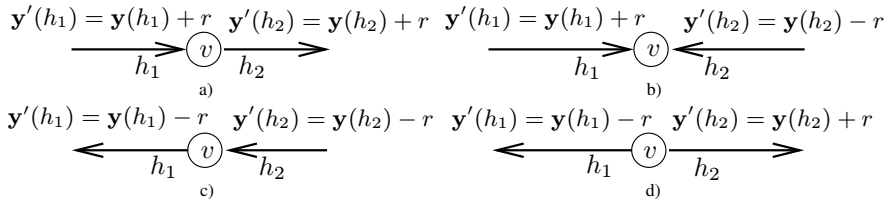
$$\mathbf{y}'(h) = \begin{cases} \mathbf{y}(h) & \text{ak } h \text{ neleží na ceste } \boldsymbol{\mu}(z, u) \\ \mathbf{y}(h) + r & \text{ak } h \text{ leží na ceste } \boldsymbol{\mu}(z, u) \text{ v smere svojej orientácie} \\ \mathbf{y}(h) - r & \text{ak } h \text{ leží na ceste } \boldsymbol{\mu}(z, u) \text{ proti smeru svojej orientácie} \end{cases}$$

Najprv treba ukázať, že  $\mathbf{y}'$  je tok, t. j. že spĺňa (7.1) až (7.4) z definície toku 7.2. Pretože rezerva rezervnej polocesty bola počítaná ako minimum z rezerv hrán definovaných vzťahmi (7.5), musia aj hodnoty  $\mathbf{y}'(h)$  toku  $\mathbf{y}'$  spĺňovať (7.1), (7.2).

Ak vrchol  $v$  neležal na zvčšujúcej poloceste, nezmenil sa tok žiadnou hranou incidentnou s vrcholom  $v$ , preto preň ostáva vzťah (7.3) v platnosti. Ak vrchol  $v$  ležal na zvčšujúcej poloceste, potom sú možné štyri prípady znázornené na obrázku 7.3.

V prípadoch a) a c) vidíme, že o čo sa zmení pritekajúce množstvo do vrchola  $v$ , o to sa zmení odtekajúce množstvo z vrchola  $v$  a preto (7.3) zostáva v platnosti.





Obr. 7.3: Štyri možnosti orientácie hrán incidentných s vrcholom  $v$  na rezervnej polocesti.

V prípade b) (resp. d)) sa síce zmení pritekajúce (resp. odtekajúce) množstvo dvoma hranami, avšak o čo sa tok hranou  $h_1$  (resp. hranou  $h_2$ ) zväčší, o to sa tok hranou  $h_2$  (resp. hranou  $h_1$ ) zmenší, takže celková bilancia ostane zachovaná a (7.3) ostane v platnosti.

Pretože zo zdroja  $z$  hrany len vychádzajú a do ústia len vchádzajú, prvá hrana aj posledná hrana zväčšujúcej polocesty boli použité v smere orientácie, a teda tok  $\mathbf{y}'$  prvou resp. poslednou hranou zväčšujúcej cesty je o  $r$  väčší, než tok  $\mathbf{y}$  pre tieto hrany. Prvá hrana zväčšujúcej polocesty patrí do  $H^+(z)$ , jej posledná hrana patrí do  $H^-(u)$ . Preto

$$F(\mathbf{y}') = \sum_{h \in H^+(z)} \mathbf{y}'(h) = \sum_{h \in H^+(z)} \mathbf{y}(h) + r = F(\mathbf{y}) + r \quad (7.6)$$

$$\sum_{h \in H^-(u)} \mathbf{y}'(h) = \sum_{h \in H^-(u)} \mathbf{y}(h) + r = F(\mathbf{y}) + r \quad (7.7)$$

Z (7.6) vidíme, že aj vzťah (7.4) ostal v platnosti, pričom sa však veľkosť toku zväčšila o hodnotu  $r$ . ■

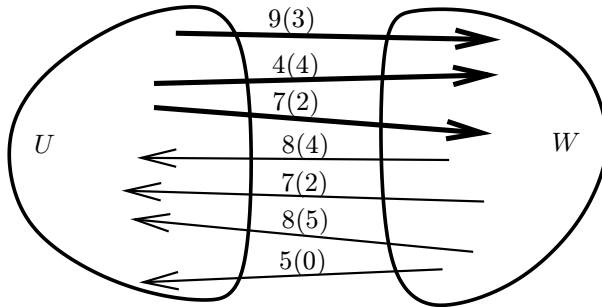
### 7.3 Rezové množiny

**Dohoda o značení.** Majme sieť  $\vec{G} = (V, H, c)$  so zdrojom  $z$  a ústím  $u$ . Nech  $U \subseteq V$ ,  $W \subseteq V$  sú dve disjunktné podmnožiny vrcholovej množiny  $V$ . Symbolom  $H(U, W)$  budeme značiť množinu všetkých orientovaných hrán vychádzajúcich z množiny  $U$  a vchádzajúcich do množiny  $W$ , t. j.

$$H(U, W) = \{(u, w) \mid (u, w) \in H, u \in U, w \in W\}.$$

Symbolom  $c[H(U, W)]$  budeme značiť súhrnnú kapacitu množiny hrán  $H(U, W)$  a pod zápisom  $y[H(U, W)]$  budeme rozumieť súhrnný tok tečúci všetkými hranami množiny  $H(U, W)$ . Matematicky vyjadrené

$$c[H(U, W)] = \sum_{h \in H(U, W)} c(h) \quad \mathbf{y}[H(U, W)] = \sum_{h \in H(U, W)} \mathbf{y}(h)$$



Obr. 7.4: Do  $H(U, W)$  patria len hrany začínajúce v  $U$  a končiace vo  $W$  (hrubé šípky).

Hrany začínajúce vo  $W$  a končiace v  $U$  (tenké šípky) do  $H(U, W)$  nepatria.

Platí:  $c[H(U, W)] = 9 + 4 + 7 = 20$ ,  $\mathbf{y}[H(U, W)] = 3 + 4 + 2 = 9$ .

**Definícia 7.4.** Majme sieť  $\vec{G} = (V, H, c)$  so zdrojom  $z$  a ústím  $u$ . Nech  $R \subseteq V$  je taká podmnožina vrcholovej množiny, že  $z \notin R$ ,  $u \in R$ . Označme  $R^C = V - R$ . Potom **rezová množina** medzi zdrojom a ústím prislúchajúca množine  $R$  je definovaná nasledovne:

$$H_R = H(R^C, R) = \{(u, v) \mid (u, v) \in H, u \notin R, v \in R\}.$$

**Priepustnosťou rezovej množiny  $H_R$  nazveme číslo**

$$c(H_R) = \sum_{h \in H_R} c(h).$$

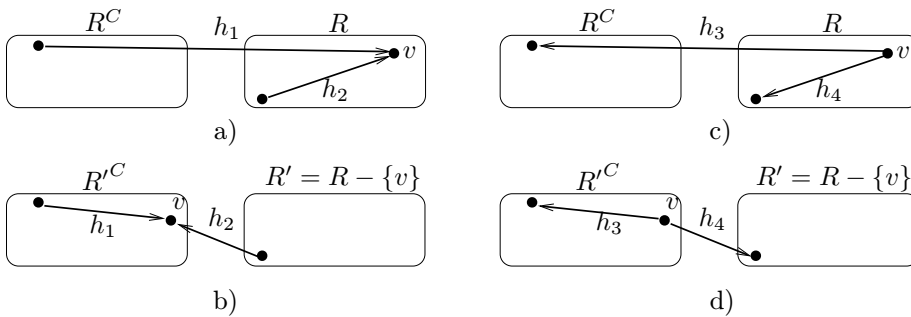
**Veta 7.2.** Nech  $\mathbf{y}$  je tok v sieti  $\vec{G} = (V, H, c)$ ,  $F(\mathbf{y})$  veľkosť toku  $\mathbf{y}$ . Pre každú množinu  $R \subseteq V$  takú, že  $z \notin R$  a  $u \in R$  platí

$$F(\mathbf{y}) = \mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)] \leq c[H(R^C, R)] = c(H_R). \quad (7.8)$$

DŮKAZ.

Z nerovnosti (7.2) máme  $\mathbf{y}[H(R^C, R)] = \mathbf{y}(H_R) \leq c(H_R)$  a z nerovnosti (7.1) vyplýva, že  $\mathbf{y}[H(R, R^C)]$  je nezáporné číslo, takže nerovnosť v (7.8) vždy platí.

Ukážeme, že  $F(\mathbf{y}) = \mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)]$ . Nech najprv  $R = V - \{z\}$ . Potom  $H_R$  je množinou práve všetkých hrán výstupnej hviezdy zdroja  $z$ , t. j.  $H_R = H^+(z)$  a  $H(R, R^C) = \emptyset$ . Preto je  $\mathbf{y}[H(R^C, R)] = \mathbf{y}(H_R) = \sum_{h \in H^+(z)} \mathbf{y}(h) = F(\mathbf{y})$  a  $\mathbf{y}[H(R, R^C)] = 0$  a rovnosť v (7.8) platí.



Obr. 7.5: Účinok vybratia vrchola  $v$  z množiny  $R$  – vznikne množina  $R' = R - \{v\}$ .

- a) Hrany  $h_1, h_2$  predstavujú dva možné prípady hrán z  $H^-(v)$ ,  
 $h_1 \in H(R^C, R)$ ,  $h_2 \notin H(R^C, R)$ ,  $h_2 \notin H(R, R^C)$ .
- b) Po vybratí vrchola  $v$  z množiny  $R$  vznikne množina  $R'$ , pre ktorú platí  
 $h_1 \notin H(R'^C, R')$ ,  $h_1 \notin H(R', R'^C)$ ,  $h_2 \in H(R', R'^C)$ .
- c) Hrany  $h_3, h_4$  predstavujú dva možné prípady hrán z  $H^+(v)$ ,  
 $h_3 \in H(R, R^C)$ ,  $h_4 \notin H(R^C, R)$ ,  $h_4 \notin H(R, R^C)$ .
- d) Po vybratí vrchola  $v$  z množiny  $R$  vznikne množina  $R'$ , pre ktorú platí  
 $h_3 \notin H(R'^C, R')$ ,  $h_3 \notin H(R', R'^C)$ ,  $h_4 \in H(R'^C, R')$ .

Nech rovnosť v (7.8) platí pre nejaké  $R \subseteq V$  také, že  $z \notin R$  a  $u \in R$ . Budeme skúmať, čo sa stane, ak vyberieme ľubovoľný vrchol  $v$  z množiny  $R$ . Označme  $R' = R - \{v\}$  pre ľubovoľné  $v \in R$ ,  $v \neq u$ . Skúmame hrany z množiny  $H^-(v)$ . Na obrázku 7.5 a) sú zobrazené dva možné prípady polohy hrán z  $H^-(v)$  ako  $h_1, h_2$ , na obrázku 7.5 b) ich poloha po vylúčení vrchola  $v$  z množiny  $R$ .

Ak  $h_1 \in H^-(v)$  bola v  $H(R^C, R)$ , jej začiatok ležal v  $R^C$  a preto táto hrana nebude ležať ani v  $H(R'^C, R')$ , ani v  $H(R', R'^C)$  a preto zmenší rozdiel

$\mathbf{y}[H(R^C, R')] - \mathbf{y}[H(R', R'^C)]$  oproti rozdielu  $\mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)]$  o hodnotu  $\mathbf{y}(h_1)$ .

Ak hrana  $h_2 \in H^-(v)$  neležala v  $H(R^C, R)$  jej začiatok ležal v  $R$ . Po vylúčení konca  $v$  hrany  $h$  z množiny  $R$  jej začiatok ostáva v  $R'$ , a preto sa hrana  $h$  dostáva do množiny  $H(R', R'^C)$  a znova zmenší výraz  $\mathbf{y}[H(R'^C, R')]$  oproti výrazu  $\mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)]$  o hodnotu  $\mathbf{y}(h_2)$ .

Vezmime teraz hranu z množiny  $H^+(v)$ . Na obrázku 7.5 c) sú zobrazené dva možné prípady polohy hrán z  $H^+(v)$  ako  $h_3, h_4$ , na obrázku 7.5 d) ich poloha po vylúčení vrchola  $v$  z množiny  $R$ . Ak  $h_3 \in H^+(v)$  bola v  $H(R, R^C)$ , jej koniec ležal v množine  $R^C$ , a preto táto hrana nebude ležať ani v  $H(R'^C, R')$ , ani v  $H(R', R'^C)$ , a preto zväčší rozdiel  $\mathbf{y}[H(R'^C, R')]$  oproti rozdielu  $\mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)]$  o hodnotu  $\mathbf{y}(h_3)$ .

Ak hrana  $h_4$  z množiny  $H^+(v)$  neležala v  $H(R, R^C)$  jej koniec ležal v  $R$ . Po vylúčení začiatku  $v$  hrany  $h$  z množiny  $R$  jej koniec ostáva v  $R'$ , a preto sa hrana  $h$  dostáva do množiny  $H(R'^C, R')$  a znova zväčší výraz  $\mathbf{y}[H(R'^C, R')]$  oproti výrazu  $\mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)]$  o hodnotu  $\mathbf{y}(h_4)$ .

Môžeme preto písať

$$\begin{aligned} \mathbf{y}[H(R'^C, R')] - \mathbf{y}[H(R', R'^C)] &= \\ &= \mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)] + \underbrace{\sum_{h \in H^+(v)} \mathbf{y}(h) - \sum_{h \in H^-(v)} \mathbf{y}(h)}_{=0} = \\ &= \mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)] \quad (7.9) \end{aligned}$$

Vylúčením ľubovoľného prvku (mimo ústia) z množiny  $R$  sa (7.8) nemení. Ak začneme množinou  $V - \{z\}$ , postupným vylučovaním jej prvkov rôznych od ústia možno dospieť k ľubovoľnej množine  $R$  obsahujúcej ústie, pričom platí (7.8). ■

Predchádzajúca veta hovorí, že súčet hranových prietokov z množiny  $R^C$  do množiny  $R$  mínus súčet hranových prietokov tečúcich z množiny  $R$  do množiny  $R^C$  sa rovná veľkosti toku za predpokladu, že množina  $R$  obsahuje ústie a neobsahuje zdroj.

**Veta 7.3. Ford – Fulkerson** *Velkosť maximálneho toku v sieti sa rovná minimu rezovej priepustnosti, t. j.:*

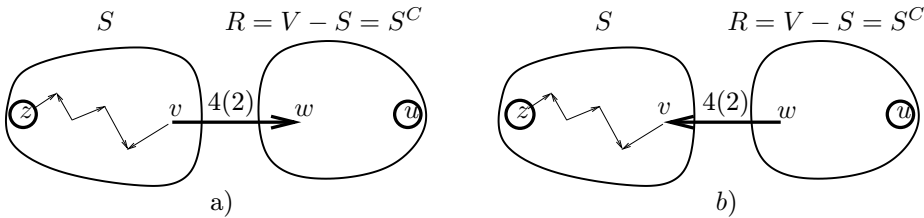
$$\max \left\{ \sum_{h \in H^+(z)} \mathbf{y}(h) \mid \mathbf{y} \text{ je tok v sieti } \vec{G} \right\} = \\ = \min \{c(H_R) \mid H_R \text{ je rezová množina medzi } z, u\}.$$

DŮKAZ.

Nech  $\mathbf{y}^*$  je maximálny tok v sieti  $\vec{G}$ , nech  $c^*$  je minimum priepustností všetkých rezových množín. Pretože veta 7.2 platí pre ľubovoľný tok a ľubovoľný rez, máme

$$F(\mathbf{y}^*) \leq c^*. \quad (7.10)$$

Nech  $S$  je množina vrcholov obsahujúca zdroj  $z$  a všetky také  $v \in V$ , že v sieti  $\vec{G}$  s tokom  $\mathbf{y}^*$  existuje rezervná  $z$ - $v$  polocesta. Nech  $R = V - S$ . Platí  $u \in R$ , lebo keby  $u \in S$ , tok  $\mathbf{y}^*$  by nebol maximálny.



Obr. 7.6: a) Keby existovala nenasýtená hrana  $(v, w)$  z  $S$  do  $R$ ,  
t. j.  $(v, w) \in H_R = H(S, R)$ ,  
možno predĺžiť rezervnú polocestu do vrchola  $w$ .

b) Keby existovala hrana  $(w, v)$  z  $R$  do  $S$ , t. j.  $(w, v) \in H(R, S)$   
s nenulovým tokom, možno predĺžiť rezervnú polocestu do vrchola  $w$ .

Všetky hrany začínajúce v  $S$  a končiacie v  $R$  sú nasýtené. Inak by bolo možné predĺžiť rezervnú polocestu do niektorého vrchola množiny  $R$ , čo nie je možné. Tok každou hranou začínajúcou v  $R$  a končiacou v  $S$  je nulový, lebo inak by bolo možné predĺžiť rezervnú polocestu do niektorého vrchola množiny  $R$ , čo zase nie je možné. Preto platí  $\mathbf{y}^*(H_R) = c(H_R)$ ,  $\mathbf{y}^*[H(R, R^C)] = 0$ , čo po dosadení do (7.8) dáva  $F(\mathbf{y}^*) = c(H_R)$ . Posledná rovnosť spolu s (7.10) dáva  $c(H_R) \leq c^*$ . Ale  $c^*$  bolo definované ako minimum priepustností všetkých rezových množín, preto  $c^* \leq c(H_R)$ . Teda  $c(H_R) = c^*$ , čo znamená, že  $H_R$  je rezová množina s minimálnou priepustnosťou. ■



musí obsahovať ústie  $u$ . Podobne ako v dôkaze vety 7.3 sa ukáže, že  $H(R^C, R)$  obsahuje len nasýtené hrany a  $H(R, R^C)$  obsahuje iba hrany s nulovým tokom – inak by bolo možné rezervnú polocestu predĺžiť. Pretože  $F(\mathbf{y}) = \mathbf{y}[H(R^C, R)] - \mathbf{y}[H(R, R^C)]$  a  $\mathbf{y}[H(R, R^C)] = 0$ , je  $F(\mathbf{y}) = \mathbf{y}[H(R^C, R)]$  – veľkosť toku  $\mathbf{y}$  dosiahla kapacitu rezovej množiny  $H(R^C, R)$  (lebo všetky jej hrany boli nasýtené). Podľa Ford – Fulkersonovej vety je tok  $\mathbf{y}$  maximálny. ■

**Algoritmus 7.1. Fordov – Fulkersonov algoritmus na hľadanie maximálneho toku v sieti  $\vec{G} = (V, H, c)$ .**

- **Krok 1.** Zvoľ v sieti začiatočný tok  $\mathbf{y}$ , napríklad nulový tok.
- **Krok 2.** Nájdi v sieti  $\vec{G}$  s tokom  $\mathbf{y}$  zväčšujúcu polocestu  $\mu(z, u)$ .
- **Krok 3.** Ak zväčšujúca polocesta neexistuje, tok  $\mathbf{y}$  je maximálny. STOP.
- **Krok 4.** Ak zväčšujúca polocesta  $\mu(z, u)$  existuje a má rezervu  $r$ , zmeň tok  $\mathbf{y}$  nasledujúco:

$$\mathbf{y}(h) := \begin{cases} \mathbf{y}(h) & \text{ak } h \text{ neleží na ceste } \mu(z, u) \\ \mathbf{y}(h) + r & \text{ak } h \text{ leží na ceste } \mu(z, u) \text{ v smere svojej orientácie} \\ \mathbf{y}(h) - r & \text{ak } h \text{ leží na ceste } \mu(z, u) \text{ proti smeru svojej orientácie} \end{cases}$$

GOTO Krok 2.



Uvedieme ešte jeden z možných algoritmov na hľadanie zväčšujúcej polocesty  $\mu(z, u)$ .

**Algoritmus 7.2. Algoritmus na hľadanie zväčšujúcej polocesty  $\mu(z, u)$  v sieti  $\vec{G} = (V, H, c)$  s tokom  $\mathbf{y}$ .**

Vrcholom siete okrem zdroja priradíme značku  $x(i)$  s nasledujúcim významom: Ak  $x(i) = \infty$ , potom do vrchola  $i$  doteraz nebola nájdená zlepšujúca  $u-i$  cesta. Ak  $x(i) < \infty$ , potom bola nájdená zlepšujúca  $u-i$  cesta, pričom jej predposledný vrchol je  $|x(i)|$  (absolútna hodnota  $x(i)$ ). Ak navyiac  $x(i) > 0$ , potom v tejto zlepšujúcej polocesta bola použitá hrana  $(x(i), i)$  v smere orientácie, ak  $x(i) < 0$ , potom v tejto zlepšujúcej polocesta bola použitá hrana  $(i, x(i))$  proti smeru orientácie. Pre zdroj  $z$  položíme  $x(z) := 0$ .

Ďalej zavedieme tieto označenia:

- $\mathcal{I}$  – množina nepreskúmaných vrcholov označených konečnou značkou  
 $\mathcal{N}$  – množina vrcholov s nekonečnou značkou

• **Krok 1.** Inicializácia.

$\mathcal{N} := V - \{z\}$ ,  $\mathcal{I} := \{z\}$ .

Polož  $x(z) := 0$  a pre všetky  $i \in \mathcal{N}$  polož  $x(i) := \infty$ .

• **Krok 2.** Ak  $x(u) < \infty$ , zostroj zlepšujúcu  $z$ - $u$  polocestu pomocou značiek  $|x(\cdot)|$ :

$$(z = |x^{(k)}(u)|, |x^{(k-1)}(u)|, \dots, |x^{(2)}(u)|, |x(u)|, u,)$$

a STOP.

• **Krok 3.** Ak  $\mathcal{I} = \emptyset$ , neexistuje zlepšujúca  $\mu(z, u)$  polocesta. STOP.

• **Krok 4.** Vyber vrchol  $i \in \mathcal{I}$ .

Polož  $\mathcal{I} := \mathcal{I} - \{i\}$ .

Pre každý vrchol  $j \in V^+(i) \cap \mathcal{N}$  urob:

Ak  $\mathbf{y}(i, j) < c(i, j)$ , potom polož  $x(j) := i$ ,  $\mathcal{I} := \mathcal{I} \cup \{j\}$  a  $\mathcal{N} := \mathcal{N} - \{j\}$ .

Pre každý vrchol  $j \in V^-(i) \cap \mathcal{N}$  urob:

Ak  $\mathbf{y}(j, i) > 0$ , potom polož  $x(j) := -i$ ,  $\mathcal{I} := \mathcal{I} \cup \{j\}$  a  $\mathcal{N} := \mathcal{N} - \{j\}$ .

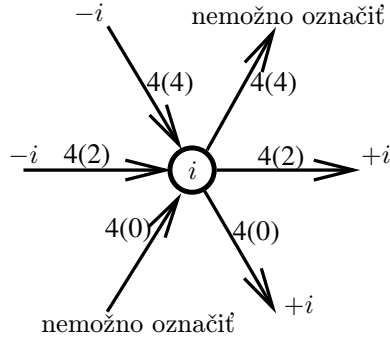
GOTO Krok 2.



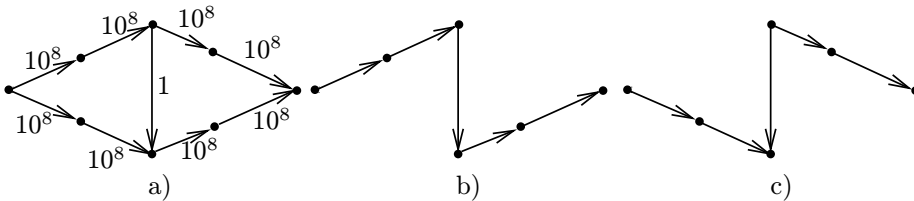
Skúmame zložitosť Fordovho–Fulkersonovho algoritmu. Ak budeme udržiavať sieť vo forme výstupných a vstupných hviezd, pri označovaní vrcholov značkami  $+i$ ,  $-i$  použijeme každú hranu najviac raz. Nájdenie zväčšujúcej polocesty zo zdroja do ústia možno urobiť v čase  $O(m)$ . To je najlepší z výsledkov, aký bolo možné očakávať. Avšak v zápätí prichádza zlá správa – pri zlom výbere poradia označovania vrcholov môžeme sústavne dostávať zväčšujúce polocesty iba s rezervou rovnou 1, čo znamená, že ak je veľkosť maximálneho toku  $Q$ , je zložitosť celého algoritmu  $O(Q.m)$ .

Edmonds a Karp dokázali, že ak volíme vždy najkratšiu zväčšujúcu polocestu (t. j. polocestu s najmenším počtom hrán), počet zmien toku nepresiahne číslo  $\frac{mn}{2}$ , kde  $n$  je počet vrcholov a  $m$  je počet hrán siete.





Obr. 7.8: Spôsob označovania z vrchola  $i$ .  
Označenie hrany  $4(2)$  znamená, že hranou kapacity 4 tečie tok 2.

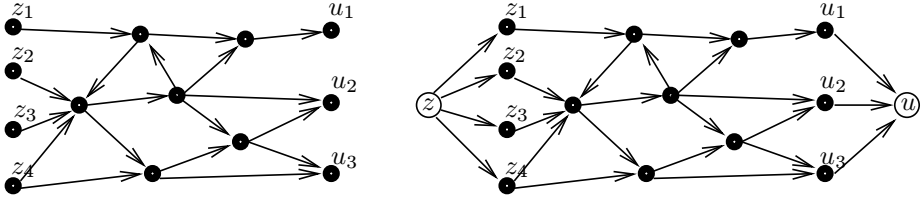


Obr. 7.9: V sieti a) je veľkosť maximálneho toku  $2 \cdot 10^8$ .  
Nešťastným poradím označovania vrcholov možno dôjsť k striedaniu dvoch zväčšujúcich polociest b), c) s rezervou 1.

## 7.5 Siete s viacerými zdrojmi a ústiami

V praktickom živote sa stretávame aj so sieťami, ktoré majú mnoho zdrojov a mnoho ústí. Elektrifikačná sieť má niekoľko zdrojov – elektrární a veľmi veľa ústí – za to môžeme považovať každú továreň, úrad, školu, nemocnicu či domácnosť, ktorá elektrickú energiu spotrebováva. Takúto situáciu možno modelovať sieťou  $\vec{G}$  s niekoľkými zdrojmi a ústiami. Aby sme túto situáciu prispôbili nášmu modelu s jedným zdrojom a jedným ústím, pridáme ku grafu  $\vec{G}$  jeden fiktívny zdroj a jedno fiktívne ústie. Od zdroja vedieme orientované hrany ku všetkým zdrojom siete  $\vec{G}$ . Kapacitu hrany typu (fiktívny zdroj, zdroj) môžeme definovať ako nekonečnú, ale ponúka sa aj možnosť definovaním konečnej kapacity tejto hrany definovať kapacitu príslušného zdroja. Podobne vedieme

orientované hrany s neohraničenou kapacitou (alebo kapacitou príslušného ústia) od všetkých ústí siete  $\vec{G}$  k fiktívnemu ústiu. Dostávame tak sieť v našom doterajšom zmysle, v ktorej môžeme vyriešiť všetky kapacitné problémy týkajúce sa siete  $\vec{G}$ .



Obr. 7.10: Pridanie fiktívneho zdroja a fiktívneho ústia.

## 7.6 Dolné medze pre tok

Existuje zovšeobecnenie úlohy o maximálnom toku v tom zmysle, že okrem kapacitného ohodnotenia hrany  $c(h)$  je dané ešte jedno ohodnotenie hrany  $0 \leq b(h) \leq c(h)$  znamenajúce dolné ohraničenie na veľkosť toku hranou.

**Definícia 7.5.** Digraf  $\vec{G} = (V, H, b, c)$  s jedným zdrojom  $z$  a jedným ústím  $u$  s celočíselnými ohodnoteniami hrán  $b, c$  takými, že  $b(h) \leq c(h)$  nazveme **intervalovo ohodnotenou sieťou**.

**Definícia 7.6.** **Prípustným tokom v intervalovo ohodnotenej sieti**  $\vec{G} = (V, H, b, c)$  nazveme celočíselnú funkciu  $\mathbf{y} : H \rightarrow \mathbb{R}$  definovanú na množine orientovaných hrán  $H$ , pre ktorú platí:

1.  $\mathbf{y}(h) \geq b(h)$  pre všetky  $h \in H$  (7.11)

2.  $\mathbf{y}(h) \leq c(h)$  pre všetky  $h \in H$  (7.12)

3.  $\sum_{h \in H^+(v)} \mathbf{y}(h) = \sum_{h \in H^-(v)} \mathbf{y}(h)$  pre všetky také  $v \in V$ , že  $v \neq u, v \neq z$  (7.13)

4.  $\sum_{h \in H^+(z)} \mathbf{y}(h) = \sum_{h \in H^-(u)} \mathbf{y}(h)$  (7.14)

Vidíme, že oproti definícii toku 7.2 (str. 190) sa v definícii toku v intervalovo ohodnotenej sieti zmenila len vlastnosť (7.11), vlastnosti (7.12), (7.13) a (7.14)

ostali rovnaké ako (7.2), (7.3) a (7.4). Skrátene môžeme definovať prípustný tok v intervalovo ohodnotenej sieti aj ako tok  $\mathbf{y}$ , pre ktorý platí  $b(h) \leq y(h)$  pre všetky hrany  $h \in H$ .

Tok v sieti bez dolnej medze môžeme považovať za špeciálny prípad toku v intervalovo ohodnotenej sieti, kde  $b(h) = 0$  pre všetky hrany  $h \in H$ . Na rozdiel od siete bez dolnej medze, kde vždy musí existovať prípustný tok (je to napríklad nulový tok), v intervalovo ohodnotenej sieti to nemusí byť tak.



Obr. 7.11: Intervalovo ohodnotená sieť bez prípustného toku.

**Algoritmus 7.3. Algoritmus na zistenie prípustného toku v intervalovo ohodnotenej sieti  $\vec{G} = (V, H, b, c)$ .**

- **Krok 1.** K sieti  $\vec{G} = (V, H, b, c)$  vytvor pomocnú sieť  $\vec{G}'$  nasledovne
  - pridaj nový zdroj  $z'$  a nové ústie  $u'$
  - pridaj orientovanú hranu  $(u, z')$  s kapacitou  $\infty$
  - ku každej orientovanej hrane  $(v, w)$  takej, že  $b(v, w) > 0$  pridaj dve orientované hrany  $(z', w)$  a  $(v, u')$  obe s kapacitami  $b(v, w)$  a priepustnosť pôvodnej hrany  $(v, w)$  stanov na  $c(v, w) - b(v, w)$ .
  - Ak by pri predchádzajúcom postupe vznikli viacnásobné hrany typu  $(z', w)$  alebo  $(v, u')$ , nahraď ich jednou orientovanou hranou so súčtom priepustností všetkých násobných hrán.
- **Krok 2.** V pomocnej sieti  $\vec{G}'$  nájdi maximálny tok  $\bar{\mathbf{y}}$ . Ak pre veľkosť  $F(\bar{\mathbf{y}})$  toku  $\bar{\mathbf{y}}$  platí

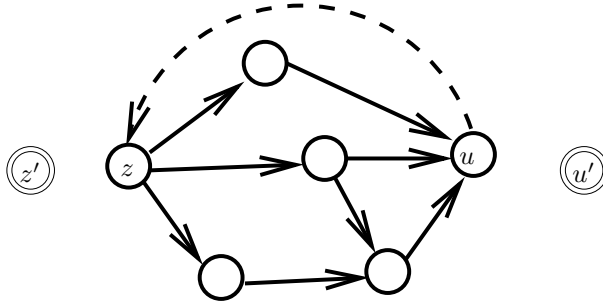
$$F(\bar{\mathbf{y}}) = \sum_{h \in H} b(h),$$

potom v sieti  $\vec{G}$  existuje prípustný tok, pre ktorý platí

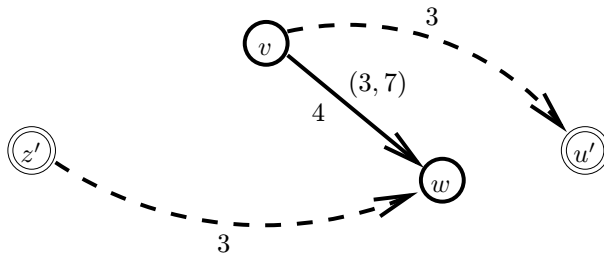
$$\mathbf{y}(h) = b(h) + \bar{\mathbf{y}}(h).$$



Ak už máme prípustný tok v intervalovo ohodnotenej sieti  $\vec{G}$ , môžeme použiť Fordov–Fulkersonov algoritmus na výpočet maximálneho toku s tým, že musíme predefinovať rezervu hrany v poloceste.



Obr. 7.12: Pridanie hrany  $(u, z)$  s priepustnosťou  $\infty$ , nového zdroja  $z'$  a nového ústia  $u'$ .

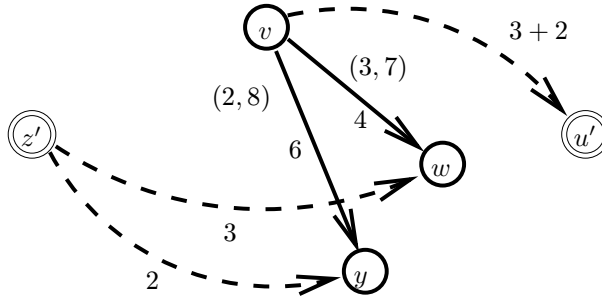


Obr. 7.13: Pridanie orientovaných hrán  $(z', w)$ ,  $(v, u')$  s priepustnosťou  $b(v, w) = 3$  a stanovenie novej priepustnosti orientovanej hrany  $(v, w)$  na  $c(v, w) - b(v, w) = 7 - 3 = 4$ .

**Definícia 7.7.** Nech  $\vec{G} = (V, H, b, c)$  je intervalovo ohodnotená sieť s tokom  $\mathbf{y}$ ,  $v, w \in V$ . Nech  $\mu(v, w)$  je  $v$ - $w$  polocesta, nech  $h$  je orientovaná hrana tejto polocesty. Definujeme  $r(h)$  **rezervu hrany** v poloceste  $\mu(v, w)$  nasledovne:

$$r(h) = \begin{cases} c(h) - \mathbf{y}(h) & \text{ak je hrana } h \text{ použitá v } \mu(v, w) \\ & \text{v smere orientácie} \\ \mathbf{y}(h) - b(h) & \text{ak je hrana } h \text{ použitá v } \mu(v, w) \\ & \text{proti smeru orientácie} \end{cases} \quad (7.15)$$

**Rezerva polocesty**  $\mu(v, w)$  je minimum rezerv hrán tejto polocesty. Hovoríme, že polocesta  $\mu(v, w)$  je **rezervná polocesta** ak má kladnú rezervu.



Obr. 7.14: Pridanie orientovanej hrany  $(z', y)$ , s priepustnosťou  $b(v, y) = 2$ .  
 Pridaním ďalšej hrany  $(v, u')$  by vznikla násobná hrana,  
 preto stačí kapacitu existujúcej hrany  $(v, u')$  zvýšiť o  $b(v, y) = 2$ .

## 7.7 Najlacnejší tok danej veľkosti

Vráťme sa k štandardnej sieti  $\vec{G} = (V, H, c)$  so zdrojom  $z$  a ústím  $u$ . Tok  $\mathbf{y}$  môže predstavovať istý spôsob presunu nejakého substrátu zo zdroja do ústia tak, že každou hranou  $h \in H$  prepravujeme  $\mathbf{y}(h)$  jednotiek substrátu. Preprava substrátu hranou čosi stojí. Cena za prepravu jednotky substrátu závisí od hrany, ktorou sa substrát prepravuje – označíme ju  $d(h)$ . Ak za prepravu jednotky substrátu po hrane  $h \in H$  zaplatíme  $d(h)$  korún, za prepravu  $\mathbf{y}(h)$  jednotiek zaplatíme  $d(h) \cdot \mathbf{y}(h)$  korún. Za prepravu všetkého tovaru zo zdroja do ústia teda zaplatíme  $\sum_{h \in H} d(h) \cdot \mathbf{y}(h)$  korún. Táto praktická paralela vedie k nasledujúcej definícii.

**Definícia 7.8.** Nech  $\vec{G} = (V, H, c, d)$  je sieť, kde  $d(h)$  je ďalšie ocenenie hrany  $h$  predstavujúce cenu za jednotku toku na hrane  $h$ . Nech  $\mathbf{y}$  je tok v sieti  $\vec{G}$ . **Cena toku  $\mathbf{y}$**  je definovaná

$$D(\mathbf{y}) = \sum_{h \in H} d(h) \cdot \mathbf{y}(h)$$

Ak máme v sieti  $\vec{G} = (V, H, c, d)$  tok  $\mathbf{y}$  veľkosti  $F(\mathbf{y})$  s cenou  $D(\mathbf{y})$ , môže nás zaujímať, či možno rovnaké množstvo substrátu prepraviť zo zdroja do ústia za menšiu cenu, t. j. či existuje tok  $\mathbf{y}'$  taký, že  $F(\mathbf{y}) = F(\mathbf{y}')$  s cenou  $D(\mathbf{y}') < D(\mathbf{y})$ . Zvlášť nás bude zaujímať otázka, ako nájsť taký tok  $\mathbf{y}'$ , ktorého

cena bude najmenšia zo všetkých tokov rovnakej veľkosti, ako má tok  $\mathbf{y}$ . Takýto tok budeme volať **najlacnejší tok veľkosti**  $F(\mathbf{y})$ .

Existuje algoritmicky rozhodnuteľné kritérium na to, či tok  $\mathbf{y}$  je najlacnejším tokom svojej veľkosti. Na formuláciu tohto kritéria potrebujeme rozšíriť definíciu rezervnej polocesty aj na polocykly.

**Definícia 7.9.** Nech  $\vec{G} = (V, H, c, d)$  je sieť s tokom  $\mathbf{y}$ ,  $C$  polocyklus v sieti  $\vec{G}$ . **Rezerva**  $r(h)$  **orientovanej hrany**  $h$  **v polocykle**  $C$  je  $r(h) = c(h) - \mathbf{y}(h)$ , ak je hrana použitá v polocykle v smere svojej orientácie a  $r(h) = \mathbf{y}(h)$ , ak je hrana použitá v polocykle  $C$  proti smeru svojej orientácie. **Rezerva polocyklu**  $C$  je minimum rezerv jeho hrán. Polocyklus  $C$  nazveme **rezervný polocyklus**, ak jeho rezerva je kladná. **Cena**  $d(C)$  **polocyklu**  $C$  je definovaná ako súčet cien hrán súhlasne orientovaných s polocyklom mínus súčet cien hrán s ním opačne orientovaných.

Sľubované kritérium je formulované v nasledujúcej vete.

**Veta 7.5.** Tok  $\mathbf{y}$  v sieti  $\vec{G} = (V, H, c, d)$  je najlacnejším tokom svojej veľkosti práve vtedy, ak v sieti  $\vec{G}$  neexistuje rezervný polocyklus zápornej ceny.

DŮKAZ.

Dokážeme, že neexistencia rezervného polocyklu zápornej ceny je nutnou podmienkou pre to, aby tok  $\mathbf{y}$  bol najlacnejší.

Ak v sieti  $\vec{G}$  existuje rezervný polocyklus  $C$  zápornej ceny s kladnou rezervou  $r$ , potom môžeme vytvoriť nový tok  $\mathbf{y}'$  predpisom:

$$\mathbf{y}'(h) := \begin{cases} \mathbf{y}(h) & \text{ak } h \text{ neleží na polocykle } C \\ \mathbf{y}(h) + r & \text{ak } h \text{ leží na polocykle } C \text{ v smere svojej orientácie} \\ \mathbf{y}(h) - r & \text{ak } h \text{ leží na polocykle } C \text{ proti smeru svojej orientácie} \end{cases}$$

Podobne ako v dôkaze vety 7.1 (str. 192) sa ukáže, že nový tok  $\mathbf{y}'$  spĺňa (7.1) až (7.4) definície toku. Veľkosť toku  $\mathbf{y}'$  zostáva rovnaká ako veľkosť toku  $\mathbf{y}$ . Ak by totiž polocyklus  $C$  aj obsahoval zdroj  $z$ , musí doň vchádzať orientovanou hranou  $h_1$  v protismere a vychádzať inou hranou  $h_2$  v smere. Pretože  $\mathbf{y}'(h_1) = \mathbf{y}(h_1) - r$ ,  $\mathbf{y}'(h_2) = \mathbf{y}(h_2) + r$  a toky ostatnými hranami vychádzajúcimi zo zdroja sa nemenia, veľkosť nového toku sa nezmení, t. j.  $F(\mathbf{y}') = F(\mathbf{y})$ .

Pre cenu nového toku  $\mathbf{y}'$  môžeme písať

$$\begin{aligned}
 D(\mathbf{y}'(h)) - D(\mathbf{y}(h)) &= \sum_{h \in H} d(h) \cdot \mathbf{y}'(h) - \sum_{h \in H} d(h) \cdot \mathbf{y}(h) = \sum_{h \in H} d(h) \cdot [\mathbf{y}'(h) - \mathbf{y}(h)] = \\
 &= \sum_{h \in C} d(h) \cdot [\mathbf{y}'(h) - \mathbf{y}(h)] = \\
 &= \sum_{\substack{h \in C \\ \text{v smere}}} d(h) \cdot [\mathbf{y}'(h) - \mathbf{y}(h)] + \sum_{\substack{h \in C \\ \text{proti smeru}}} d(h) \cdot [\mathbf{y}'(h) - \mathbf{y}(h)] = \\
 &= \sum_{\substack{h \in C \\ \text{v smere}}} d(h) \cdot r - \sum_{\substack{h \in C \\ \text{proti smeru}}} d(h) \cdot r = r \cdot \left[ \underbrace{\sum_{\substack{h \in C \\ \text{v smere}}} d(h)}_{\text{v smere}} - \underbrace{\sum_{\substack{h \in C \\ \text{proti smeru}}} d(h)}_{\text{proti smeru}} \right] < 0 \quad (7.16) \\
 &= D(C) \text{ cena polocyklu } C, D(C) < 0
 \end{aligned}$$

Tretia rovnosť v (7.16) vyplýva z toho, že pre hrany  $h$  neležiace na polocykle  $C$  je  $\mathbf{y}(h) = \mathbf{y}'(h)$ . Rozdiel v hranatej zátvorke je nenulový len pre hrany cyklu  $C$ . Z (7.16) máme  $D(\mathbf{y}'(h)) < D(\mathbf{y}(h))$  – tok  $\mathbf{y}'$  má menšiu cenu než tok  $\mathbf{y}$ .

Dokázali sme, že ak v sieti  $\vec{G}$  s tokom  $\mathbf{y}$  existuje rezervný polocyklus zápornej ceny, tok  $\mathbf{y}$  nie je najlacnejší. Dôkaz toho, že neexistencia rezervného cyklu zápornej ceny implikuje, že tok  $\mathbf{y}$  je najlacnejší je ťažší, a preto ho vynecháme. ■

**Algoritmus 7.4.** Algoritmus na hľadanie najlacnejšieho toku danej veľkosti v sieti  $\vec{G} = (V, H, c, d)$ .

- **Krok 1.** Začni tokom  $\mathbf{y}$  v sieti  $\vec{G} = (V, H, c, d)$  danej veľkosti.<sup>1</sup>
- **Krok 2.** V sieti  $\vec{G}$  s tokom  $\mathbf{y}$  nájdí rezervný polocyklus  $C$  so zápornou cenou a rezervou  $r$ , alebo zisti, že taký polocyklus neexistuje.
- **Krok 3.** Ak rezervný polocyklus zápornej ceny neexistuje, tok  $\mathbf{y}$  je najlacnejší zo všetkých tokov svojej veľkosti. STOP.

<sup>1</sup>Ak má byť veľkosť toku  $\mathbf{y}$  maximálna, za  $\mathbf{y}$  vezmeme maximálny tok nájdený Fordovým–Fulkersonovým algoritmom. Ak má byť  $\mathbf{y}$  tokom menšej veľkosti než maximálnej, môžeme ho dostať z maximálneho toku jeho zmenšením.

- **Krok 4.** Ak taký polocyklus  $C$  existuje, zmeň tok  $\mathbf{y}$  nasledujúco:

$$\mathbf{y}(h) := \begin{cases} \mathbf{y}(h) & \text{ak } h \text{ neleží na polocykle } C \\ \mathbf{y}(h) + r & \text{ak } h \text{ leží na polocykle } C \text{ v smere svojej orientácie} \\ \mathbf{y}(h) - r & \text{ak } h \text{ leží na polocykle } C \text{ proti smeru svojej orientácie} \end{cases}$$

GOTO Krok 2.



Pre nájdenie rezervného polocyklu zápornej ceny možno použiť upravený Floydov algoritmus takto. Definujeme maticu  $\mathbf{D} = (d_{ij})$  takto

$$d_{ij} := \begin{cases} -d(j, i) & \text{ak } (j, i) \in H \text{ a } \mathbf{y}(j, i) > 0 \\ d(i, j) & \text{ak } \left( (i, j) \in H \text{ a } \mathbf{y}(i, j) < c(i, j) \right) \text{ a} \\ & \left( (j, i) \notin H \text{ alebo } \mathbf{y}(j, i) = 0 \right) \\ \infty & \text{inak} \end{cases}$$

Pri určovaní prvku  $d_{ij}$  matice  $\mathbf{D}$  zistíme najprv, či  $(j, i) \in H$  a hrana  $(j, i)$  má nenulový tok. Ak áno, položíme  $d_{ij} := -d(j, i)$  a máme prvok  $d_{ij}$  určený. Ak nie a  $(i, j)$  je nenasýtená hrana, položíme  $d_{ij} = d(i, j)$ , inak položíme  $d_{ij} = \infty$ . Všimnime si, že matica  $\mathbf{D}$  má na hlavnej diagonále iba hodnoty  $\infty$ .

Zároveň s maticou  $\mathbf{D}$  definujeme maticu  $\mathbf{X} = (x_{i,j})$  prepisom

$$x_{ij} = \begin{cases} i & \text{ak } d_{ij} < \infty \\ \infty & \text{inak.} \end{cases}$$

Teraz už na maticu  $\mathbf{D}$  s maticou  $\mathbf{X}$  možno aplikovať Floydov algoritmus. Nech  $\mathbf{D}^{(k)} = (d_{ij}^{(k)})$  resp.  $\mathbf{X}^{(k)} = (x_{ij}^{(k)})$  je matica, ktorá vznikne z matice  $\mathbf{D}$  resp.  $\mathbf{X}$  po  $k$ -tom kroku Floydovho algoritmu. Ak matica  $\mathbf{D}^{(k)}$  obsahuje niektorý diagonálny prvok  $d_{jj}^{(k)}$  záporný, potom sme objavili  $j$ -j polocyklus  $C$  so zápornou cenou. Tento polocyklus určíme pomocou matice smerníkov  $\mathbf{X}^{(k)}$  nasledovne. Predposledný vrchol  $j_1$  hľadaného  $j$ -j polocyklu je  $j_1 = x_{jj}^{(k)}$ . Ďalší vrchol tohto polocyklu (odzadu) je  $j_2 = x_{jj_1}^{(k)}$ , ďalší  $j_3 = x_{jj_2}^{(k)}$  atď., pokiaľ nedôjdeme znovu do vrchola  $j$ .

Ešte treba rozhodnúť, ktorú hranu použiť medzi dvoma susednými vrcholmi  $p, q$  polocyklu  $C$ . Ak  $d_{pq} > 0$  (prvok pôvodnej matice  $\mathbf{D}$ ), použijeme hranu



$(p, q)$  v smere jej orientácie, ak  $d_{pq} < 0$  použijeme hranu  $(q, p)$  proti smeru jej orientácie.

Ak po skončení Floydovho algoritmu matica  $\mathbf{D}^{(n)}$  neobsahuje na hlavnej diagonále ani jedno záporné číslo, tak v sieti  $\vec{G}$  s tokom  $\mathbf{y}$  neexistuje rezervný polocyklus zápornej ceny.

Zložitosť takto formulovaného algoritmu je  $O(n^3)$ , kde  $n = |V|$  je počet vrcholov siete  $\vec{G}$  – môže sa totiž stať, že záporný cyklus nájdeme až v matici  $\mathbf{D}^{(n)}$ , na čo potrebujeme  $n$ -krát prepočítať maticu rozmeru  $n \times n$ .

Iný spôsob na hľadanie rezervného polocyklu so zápornou cenou je v úprave základného alebo Fordovho algoritmu. Pozor, Dijkstrov algoritmus sa pre digrafy so zápornou cenou hrán nehodí. Pre jednoduchosť uvedieme úpravu základného algoritmu.

**Algoritmus 7.5. Algoritmus na hľadanie rezervného polocyklu zápornej ceny v sieti  $\vec{G} = (V, H, c, d)$  s tokom  $\mathbf{y}$ .**

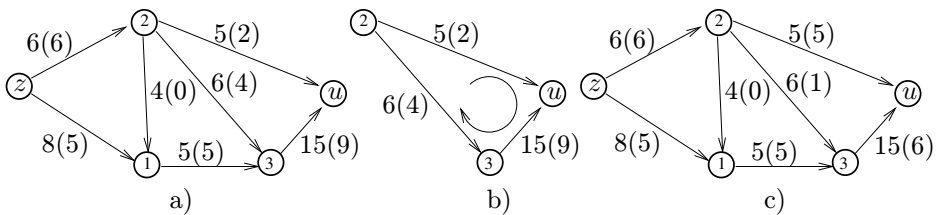
- **Krok 1.** Inicializácia. Pre každý vrchol  $i \in V$  prirad' dve značky  $t(i)$  a  $x(i)$ .  
Polož  $t(z) := 0$  pre zdroj,  $t(i) := \infty$  pre  $i \in V$ ,  $i \neq z$  a  $x(i) = 0$  pre každé  $i \in V$ .
- **Krok 2.** Postupne pre všetky orientované hrany  $h = (i, j) \in H$  urob
  - Ak  $\mathbf{y}(i, j) > 0$  a  $t(i) > t(j) - d(i, j)$ , potom
    - = polož  $t(i) := t(j) - d(i, j)$ ,  $x(i) := j$ .
    - = skontroluj, či sa v postupnosti  $x(i), x(x(i)), \dots$  vyskytuje  $i$ .  
Ak áno, STOP, našiel si rezervný polocyklus zápornej ceny.
  - Ak  $\mathbf{y}(i, j) < c(i, j)$  a  $t(j) > t(i) + d(i, j)$ , potom
    - = polož  $t(j) = t(i) + d(i, j)$ ,  $x(j) := i$ .
    - = skontroluj, či sa v postupnosti  $x(j), x(x(j)), \dots$  vyskytuje  $j$ .  
Ak áno, STOP, našiel si rezervný polocyklus zápornej ceny.
- **Krok 3.** Ak v predchádzajúcom kroku nastala aspoň jedna zmena značky  $t()$  opakuj krok 2.
- **Krok 4.** Ak boli prezreté všetky orientované hrany  $h \in H$  a nenašlo sa ani jedno zlepšenie značky  $t()$ , a ak všetky vrcholy majú konečné značky  $t()$ , STOP. V sieti  $\vec{G}$  s tokom  $\mathbf{y}$  neexistuje rezervný polocyklus zápornej ceny.

Ak existuje aspoň jeden vrchol  $j$  so značkou  $t(j) = \infty$ , vrchol  $j$  nie je dosiahnuteľný z vrchola  $z$ . Zopakuj algoritmus s vrcholom  $j$  v úlohe vrchola  $z$ .



Zložitosť algoritmu 7.5. Algoritmus hľadania rezervných polocyklov zápornej ceny urobí v Kroku 2. najviac  $m = |H|$  kontrol hrán. Kontrola na vznik cyklu sa dá urobiť najviac  $n - 1$  krokmi. Krok 2. teda algoritmus urobí v čase  $O(m.n)$ . Keďže Krok 2. vykoná najviac  $(n - 1)$  krát, celý algoritmus má zložitosť  $O(m.n^2)$ , čo je horší odhad zložitosti, ako je zložitosť  $O(n^3)$  modifikácie Floydovho algoritmu.

Ukazuje sa, že zložitosť uvedeného algoritmu na hľadanie najlacnejšieho toku danej veľkosti nie je polynomiálna, hoci v praxi dosahuje veľmi dobré výsledky.



Obr. 7.15:

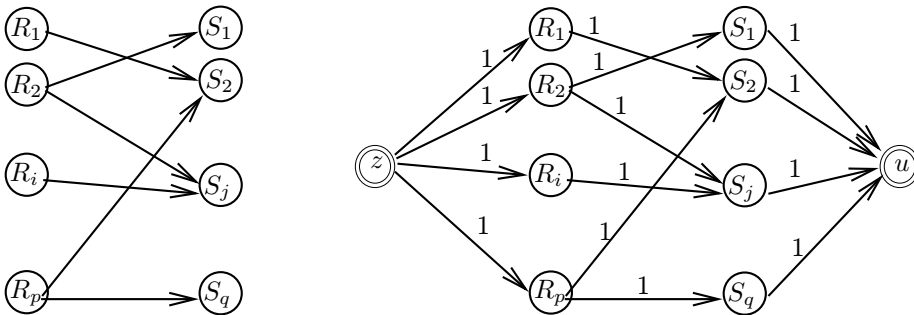
- a) Sieť  $\vec{G} = (V, H, c, d)$  s maximálnym tokom  $\mathbf{y}$ .  
 Predpokladajme, že pre každú hranu  $h$  siete  $\vec{G}$  je  $d(h) = 10$ .  
 Cena toku  $\mathbf{y}$  je  $D(\mathbf{y}) = 310$ .
- b) Rezervný polocyklus  $u, (3, u), 3, (2, 3), 2, (2, u), u$   
 so zápornou cenou  $-10$  a s rezervou 3.
- c) Upravený tok s rovnakou veľkosťou,  
 ale menšou cenou ( $=280$ ), ako mal pôvodný tok  $\mathbf{y}$ .

Praktické úlohy vedú veľmi často k hľadaniu najlacnejšieho maximálneho toku v sieti. Takúto úlohu budeme riešiť v dvoch etapách – najprv nájdeme niektorý maximálny tok a potom metódou hľadania rezervných polocyklov zápornej ceny z neho vyrobíme najlacnejší tok s rovnakou – t. j. maximálnou veľkosťou.

## 7.8 Aplikácie

### 7.8.1 Priradovacia úloha

Mažeme  $p$  robotníkov  $R_1, R_2, \dots, R_p$  a  $q$  strojov  $S_1, S_2, \dots, S_q$ . Robotníci sú špecializovaní, každý ovláda prácu na niekoľkých (ale nie nutne na všetkých) strojoch. Jeden robotník môže súčasne pracovať iba na jednom stroji a na žiadnom stroji nepracujú viacerí robotníci. Našou úlohou je priradiť robotníkom stroje tak, aby čo najviac robotníkov malo priradený stroj.



Obr. 7.16: Bipartitný digraf modelujúci schopnosť robotníkov pracovať na strojoch (vľavo) a jeho rozšírenie na tokový model (vpravo).

Popísanú situáciu môžeme modelovať bipartitným digrafom  $\vec{G} = (V, H)$ ,  $V = V_1 \cup V_2$ , v ktorom jednu časť vrcholov  $V_1$  tvoria robotníci a druhú časť vrcholov  $V_2$  tvoria stroje. Množinu hrán  $H$  budú tvoriť všetky usporiadané dvojice  $(R_i, S_j)$  také, že robotník  $R_i$  ovláda stroj  $S_j$ .

V digrafe  $\vec{G}$  možno úlohu optimálneho priradenia robotníkov a strojov formulovať ako hľadanie najpočetnejšej podmnožiny hrán takej, že žiadne jej dve hrany nemajú spoločný incidentný vrchol. Z digrafu  $\vec{G}$  vytvoríme sieť  $\vec{G}' = (V', H', c)$  tak, že k množine vrcholov digrafu  $\vec{G}$  pridáme dva nové vrcholy  $z, u$  – fiktívny zdroj a fiktívne ústie, k množine orientovaných hrán dodáme všetky možné orientované hrany typu  $(z, R_i)$  a  $(S_j, u)$  a kapacitu  $c(h)$  všetkých orientovaných hrán digrafu  $\vec{G}'$  položíme rovnú 1. V sieti  $\vec{G}'$  nájdeme maximálny tok  $\mathbf{y}$ . Potom optimálne priradenie robotníkov a strojov

je nasledujúce: Robotníkovi  $R_i$  je priradený stroj  $S_j$  práve vtedy, keď hranou  $(R_i, S_j)$  tečie jednotkový tok, t. j. keď  $\mathbf{y}(R_i, S_j) = 1$ .

Existuje aj zložitejšia úloha priradovania robotníkov strojom. Znovu majme  $p$  robotníkov a  $q$  strojov. Robotníci sú rôzne špecializovaní a rôzne kvalifikovaní a ovládajú rôzne stroje. Preto ich mzda nie je rovnaká. Označme  $\mathbf{D} = (d_{ij})$  obdĺžnikovú maticu typu  $p \times q$ , kde  $d_{ij}$  je mzda, ktorú musíme robotníkovi  $R_i$  zaplatiť, ak bude robiť na stroji  $S_j$ ,  $d_{ij} = \infty$ , ak robotník  $R_i$  neovláda stroj  $S_j$ . Našou úlohou bude teraz priradiť robotníkom stroje tak, aby čo najviac robotníkov malo priradený stroj a aby celková čiastka, ktorú robotníkom pri tomto priradení zaplatíme, bola čo najmenšia.

Na riešenie tejto úlohy zostrojíme sieť  $\vec{G}' = (V', H', c)$  podobne ako v predchádzajúcom prípade, avšak jej hranám priradíme okrem kapacity  $c$  aj ďalšie ohodnotenie  $d$  nasledovne: Hranám typu  $(z, R_i)$  a  $(S_j, u)$  priradíme nulové ohodnotenie  $d$  a pre hrany typu  $(R_i, S_j)$  položíme  $d(R_i, S_j) = d_{ij}$ . Optimálne priradenie v tomto prípade nájdeme nasledovným spôsobom: V sieti  $\vec{G}'$  nájdeme najlacnejší maximálny tok  $\mathbf{y}$  (vzhľadom na cenu  $d$ ). Potom optimálne priradenie robotníkov a strojov je nasledujúce: Robotníkovi  $R_i$  je priradený stroj  $S_j$  práve vtedy, keď hranou  $(R_i, S_j)$  tečie jednotkový tok, t. j. keď  $\mathbf{y}(R_i, S_j) = 1$ .

Na záver tejto aplikácie poznamenajme, že úloha o priradení robotníkov a strojov sa dá formulovať a riešiť ako úloha bivalentného lineárneho programovania. Pre jednoduchosť predpokladajme, že počet robotníkov a počet strojov je rovnaký, t. j.  $p = q = n$ . Priradenie robotníkov a strojov bude popisovať štvorcová matica  $\mathbf{X} = (x_{ij})$  typu  $n \times n$ , pre prvky ktorej platí:  $x_{ij} = 1$  práve vtedy, keď robotníkovi  $R_i$  je priradený stroj  $S_j$ . Nech  $\mathbf{D} = (d_{ij})$  je matica cien priradení ako v predchádzajúcom prípade (avšak v tomto prípade štvorcová typu  $n \times n$ ). Potom priradiť robotníkom stroje tak, aby súčet ich miezd bol čo najmenší, znamená určiť prvky matice  $\mathbf{X}$  tak, aby

$$\sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij} \text{ bolo minimálne} \quad (7.17)$$

$$\text{za predpokladov} \quad \sum_{j=1}^n x_{ij} = 1 \quad (i = 1, 2, \dots, n) \quad (7.18)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad (j = 1, 2, \dots, n) \quad (7.19)$$

$$x_{ij} \in \{0, 1\} \quad (i, j = 1, 2, \dots, n) \quad (7.20)$$

Dvojitá suma v (7.17) znamená celkovú čiastku vyplatenú robotníkom pri priradení danom maticou  $\mathbf{X}$ , podmienka (7.18) znamená, že jednému robotníkovi má byť priradený práve jeden stroj a podmienka (7.19) znamená, že jednému stroju má byť priradený práve jeden robotník. Keďže každému robotníkovi musí byť teraz priradený stroj, môže sa stať, že matica  $\mathbf{X}$  priradí robotníkovi  $R_i$  stroj  $S_j$ , na ktorom robotník nevie pracovať. V tomto prípade je  $d_{ij} = \infty$  a takýto výsledok znamená, že nebolo možné priradiť každému robotníkovi stroj, na ktorom by vedel pracovať.

Úloha formulovaná v (7.17) až (7.20) sa volá **priradovacia úloha** a dá sa riešiť prostriedkami lineárneho programovania. Priradovacia úloha má nespočetné množstvo aplikácií, často sa používa aj ako súčasť iných exaktných alebo heuristických algoritmov.

## 7.8.2 Dopravná úloha

Máme  $p$  dodávateľov  $D_1, D_2, \dots, D_p$  s celočíselnými kapacitami  $a_1, a_2, \dots, a_p$  a  $q$  odberateľov  $O_1, O_2, \dots, O_q$  s celočíselnými požiadavkami  $b_1, b_2, \dots, b_q$ . Náklady na prepravu jednotky tovaru od dodávateľa  $D_i$  k odberateľovi  $O_j$  sú  $d_{ij}$ . Našou úlohou je určiť, koľko tovaru od ktorého dodávateľa ku ktorému odberateľovi doviezť tak, aby sme rozviezli čo najviac tovaru tak, aby ani kapacity dodávateľov, ani požiadavky odberateľov neboli prekročené a tak, aby celková cena za prepravu všetkého tovaru bola čo najmenšia. Takto formulovaná úloha sa volá **dopravná úloha**.

Pre riešenie tejto úlohy zostrojme sieť  $\vec{G}$  s množinou vrcholov  $V = V_1 \cup V_2 \cup \{z, u\}$ , kde  $V_1$  je množina všetkých dodávateľov,  $V_2$  množina všetkých odberateľov,  $z$  je fiktívny zdroj a  $u$  fiktívne ústie. Množinu orientovaných hrán  $H$  siete  $\vec{G}$  budú tvoriť všetky orientované hrany typu  $(z, D_i)$  s kapacitou  $a_i$  a nulovou cenou za prepravu jednotky tovaru, všetky orientované hrany typu  $(O_j, u)$  s kapacitou  $b_j$  a nulovou cenou za prepravu jednotky tovaru a nakoniec všetky orientované hrany typu  $(D_i, O_j)$  s neobmedzenou kapacitou a cenou za prepravu jednotky tovaru rovnou  $d_{ij}$ .

Ak v takto definovanej sieti  $\vec{G}$  nájdeme najlacnejší maximálny tok  $\mathbf{y}$ , potom  $\mathbf{y}(D_i, O_j)$  pre všetky  $i = 1, 2, \dots, p$ ,  $j = 1, 2, \dots, q$  sú optimálne množstvá tovaru, ktoré treba doviezť od dodávateľa  $D_i$  k odberateľovi  $O_j$  tak, aby boli splnené požiadavky zadania.

Pre dopravnú úlohu tiež existuje model lineárneho programovania. Sformulujeme ho najprv pre prípad, že súčet kapacít dodávateľov sa rovná súčtu požiadaviek odberateľov, t. j.  $\sum_{i=1}^p a_i = \sum_{j=1}^q b_j$ . V tomto prípade chceme rozviezť všetko dodávateľmi ponúkané množstvo tovaru a plne uspokojiť každého odberateľa.

Označme  $x_{ij}$  množstvo tovaru dovezené od dodávateľa  $D_i$  k odberateľovi  $O_j$ . Vyhriešiť dopravnú úlohu znamená určiť prvky matice  $\mathbf{X}$  tak, aby

$$\sum_{i=1}^p \sum_{j=1}^q d_{ij} x_{ij} \text{ bolo minimálne} \quad (7.21)$$

$$\text{za predpokladov} \quad \sum_{j=1}^q x_{ij} = a_i \quad (i = 1, 2, \dots, p) \quad (7.22)$$

$$\sum_{i=1}^p x_{ij} = b_j \quad (j = 1, 2, \dots, q) \quad (7.23)$$

$$x_{ij} \geq 0 \quad (i = 1, 2, \dots, p, \quad j = 1, 2, \dots, q) \quad (7.24)$$

Dvojitá suma v (7.21) znamená celkové prepravné náklady, ak sú prepravované množstvá určené maticou  $\mathbf{X}$ , podmienka (7.22) znamená, že od každého dodávateľa  $D_i$  prepravíme celé ponúkané množstvo  $a_i$ , podmienka (7.23) znamená, že každému odberateľovi dovezieme všetko požadované množstvo tovaru. Podmienka (7.24) hovorí, že nemožno prepravovať záporné množstvá tovaru.

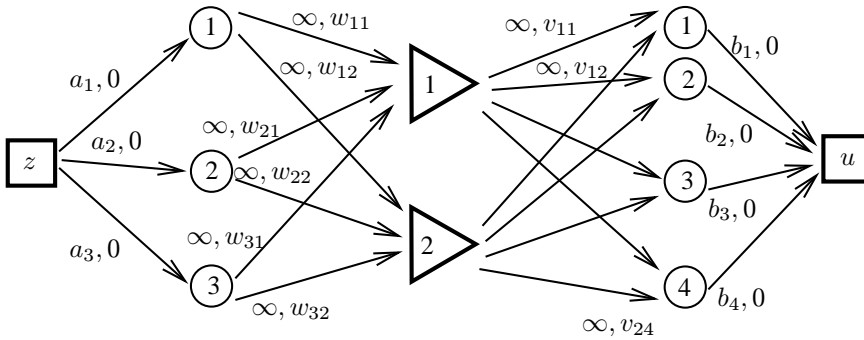
Ak by ponuka dodávateľov bola väčšia, než požiadavky odberateľov, t. j.  $\sum_{i=1}^p a_i > \sum_{j=1}^q b_j$ , model úlohy sa zmení tak, že v podmienkach (7.22) bude namiesto „=“ vzťah „ $\leq$ “ t. j. od každého dodávateľa vyvezieme najviac  $a_i$  jednotiek tovaru. Ostatné obmedzenia (7.23), (7.24) ostávajú nezmenené. Podobne, ak  $\sum_{i=1}^p a_i < \sum_{j=1}^q b_j$ , potom sa v podmienkach (7.23) zmení „=“ na „ $\leq$ “.

### 7.8.3 Dopravná úloha s medziskladmi

V niektorých prípadoch tovar od prvotných dodávateľov (napr. výrobcov) nejde priamo k odberateľom, ale dopravuje sa najprv do medziskladov, odkiaľ sa expeduje k samotným odberateľom.

Máme  $p$  dodávateľov  $D_1, D_2, \dots, D_p$  s kapacitami  $a_1, a_2, \dots, a_p$  a  $q$  odberateľov  $O_1, O_2, \dots, O_q$  s požiadavkami  $b_1, b_2, \dots, b_q$ . Prepravovaný tovar sa musí najprv doviezť do  $r$  medziskladov  $S_1, S_2, \dots, S_r$ . Náklady na prepravu jednotky

tovaru od dodávateľa  $D_i$  k medziskladu  $S_j$  sú  $w_{ij}$ , do medziskladu  $S_j$  k odberateľovi  $O_k$   $v_{jk}$ . Treba určiť množstvá tovaru  $x_{ij}$  prevezené od  $D_i$  k  $S_j$  a tiež množstvá  $y_{jk}$  od  $S_j$  k  $O_k$  tak, aby sme rozviezli čo najviac tovaru tak, aby ani kapacity dodávateľov, ani požiadavky odberateľov neboli prekročené a tak, aby celková cena za prepravu všetkého tovaru bola čo najmenšia.



Obr. 7.17: Model teórie grafov pre dopravnú úlohu s dvoma medziskladmi s tromi dodávateľmi a štyrmi odberateľmi

Ako model teórie grafov pre formulovanú úlohu bude digraf  $G$  s množinou vrcholov  $V = V_1 \cup V_2 \cup V_3 \cup \{z, u\}$ , kde  $V_1$  je množina všetkých dodávateľov,  $V_2$  množina všetkých odberateľov,  $V_3$  množina všetkých medziskladov,  $z$  je fiktívny zdroj a  $u$  fiktívne ústie. Množinu hrán  $H$  siete  $G$  budú tvoriť všetky hrany typu  $(z, D_i)$  s kapacitou  $a_i$  a nulovou cenou za prepravu jednotky tovaru, všetky hrany typu  $(O_j, u)$  s kapacitou  $b_j$  a nulovou cenou za prepravu jednotky tovaru a nakoniec všetky hrany typu  $(D_i, S_j)$  a  $(S_j, O_k)$  s neobmedzenou kapacitou a cenou za prepravu jednotky tovaru rovnou  $w_{ij}$ , resp.  $v_{jk}$ . Príklad grafu  $G$  pre 3 dodávateľov a 4 odberateľov a 2 medzisklady je na obrázku 7.17.

Riešiť takto formulovanú úlohu znamená nájsť v sieti  $G$  najlacnejší maximálny tok.

Dodatočným výsledkom riešenia môže byť i požadovaná kapacita každého skladu  $S_j$ , ktorú dostaneme ako množstvo tovaru dovezeného do  $S_j$  od všetkých dodávateľov.

### 7.8.4 Optimalizácia turnusov v autobusovej doprave

Základnou jednotkou prepravnej práce v pravidelnej autobusovej doprave je spoj. Spoj  $s_i$  je charakterizovaný miestom odchodu  $mo_i$ , miestom príchodu  $mp_i$ , časom odchodu  $co_i$  a časom príchodu  $cp_i$ .<sup>2</sup>

Spoje v regionálnej autobusovej doprave sú časovo i kilometricky krátke na to, aby jeden takýto spoj vyplnil celodenný výkon vozidla a vodiča. Preto sa jednému vozidlu do denného rozpisu práce prideluje vykonanie postupnosti spojov. Táto postupnosť spojov pre jedno vozidlo sa volá **turnus vozidla**.

Hovoríme, že **spoj  $s_i$  predchádza spoj  $s_j$**  a píšeme  $s_i \prec s_j$ , ak po príchode spoja  $s_i$  do svojho miesta príchodu  $mp_i$  ostáva autobusu ešte dosť času na to, aby prázdny prejazdom prešiel do miesta odchodu  $mo_j$  spoja  $s_j$  a stihol ešte čas odchodu  $co_j$  spoja  $s_j$ . Ak  $s_i \prec s_j$ , potom sú spoje  $s_i, s_j$  zaraditeľné do jedného turnusu. So zaradením spoja  $s_j$  bezprostredne za spoj  $s_i$  do toho istého turnusu je spojený prejazd z miesta príchodu  $mp_i$  spoja  $i$  do miesta odchodu  $mo_j$  spoja  $j$ . Dĺžku tohto prejazdu (v kilometroch) označme  $d_{ij}$ . Ak  $s_i \not\prec s_j$ , potom  $d_{ij} = \infty$ .

Majme množinu spojov  $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ . Našou úlohou je zaradiť tieto spoje do turnusov tak, aby každý spoj bol práve v jednom turnuse. Pritom chceme, aby počet týchto turnusov bol čo najmenší a aby pritom vozidlá najazdili čo najmenší počet prázdnych kilometrov. Na základe týchto požiadaviek definujeme dve nasledujúce základné úlohy optimalizácie turnusov.

**Základnou úlohou I. optimalizácie turnusov** je zaradiť všetky spoje danej množiny  $\mathcal{S}$  do minimálneho počtu turnusov.

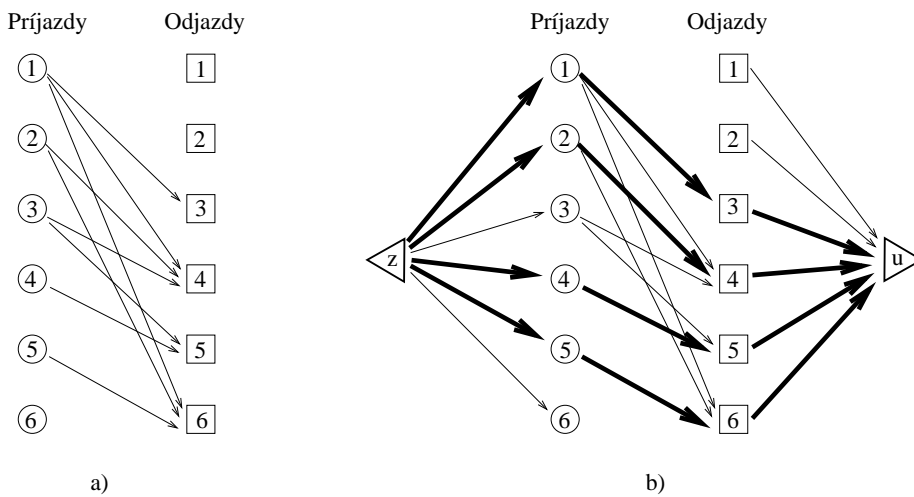
**Základnou úlohou II. optimalizácie turnusov** je zaradiť všetky spoje danej množiny  $\mathcal{S}$  do minimálneho alebo daného počtu turnusov tak, aby súčet všetkých prázdnych prejazdov bol minimálny.

Základnú úlohu I. optimalizácie turnusov riešime pomocou nasledujúceho grafového modelu. Zostrojíme sieť  $\vec{G}$  s množinou vrcholov  $V = V_1 \cup V_2 \cup \{z, u\}$ , kde  $V_1$  je množina všetkých príchodov spojov z množiny  $\mathcal{S}$ ,  $V_2$  je množina všetkých odchodov spojov z  $\mathcal{S}$ ,  $z$  je fiktívny zdroj a  $u$  fiktívne ústie. Symbolom  $v_i$ , resp.  $w_j$  budeme značiť vrchol množiny  $V_1$  resp.  $V_2$  zodpovedajúci spoju  $s_i$  resp.

<sup>2</sup>V skutočnosti pre plnú charakterizáciu spoja je potrebné zadať aj postupnosť nácestných zastávok a časy príchodov do týchto zastávok, ako sú uvedené v cestovnom poriadku. Ďalej sú so spojom zviazané aj číslo linky a číslo spoja, ktoré jednoznačne identifikujú spoj. Pre účely optimalizácie turnusov sú však potrebné iba štyri spomenuté údaje.



$s_j$ . Množina orientovaných hrán  $H$  siete  $\vec{G}$  bude obsahovať orientované hrany trojakého typu. Hrany 1. typu budú všetky orientované hrany tvaru  $(z, v_i)$ , kde  $v_i \in V_1$  s kapacitou 1 a nulovou cenou za jednotku toku. Hrany 2. typu budú všetky orientované hrany tvaru  $(w_j, u)$ , kde  $w_j \in V_2$  s jednotkovou kapacitou a nulovou cenou. Množina  $H$  bude ešte obsahovať všetky orientované hrany typu  $(v_i, w_j)$  také, že  $s_i \prec s_j$  s jednotkovou kapacitou a cenou za jednotku toku  $d(v_i, w_j) = d_{ij}$ .



Obr. 7.18: a) Digraf modelujúci reláciu  $\prec$ .

b) Sieť  $\vec{G}$  spolu s maximálnym tokom. Všetky hrany v  $\vec{G}$  majú kapacity = 1. Príslušné turnusy sú  $T_1 : 1 \rightarrow 3$ ,  $T_2 : 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$ .

Predstavme si, že začneme s riešením, kde použijeme toľko autobusov, koľko je spojov. Každý autobus vykoná jeden spoj a skončí svoju dennú prácu. Ako môžeme znížiť počet autobusov? Môžeme to urobiť tak, že zrealizujeme niektorú z hrán typu  $(v_i, w_j)$  siete  $\vec{G}$  – pod zrealizovaním tu rozumieme, že autobus po vykonaní spoja  $s_i$  vykoná aj spoj  $s_j$ . S každou ďalšou zrealizovanou hranou ušetríme ďalší autobus. Medzi zrealizovanými hranami však nesmú byť žiadne dve incidentné s rovnakým vrcholom – ide teda o priradenie odchodov spojov príchodom spojov tak, aby čo najviac odchodov spojov malo priradený príchod nejakého spoja. Z časti o priraďovacej úlohe už vieme, že to možno urobiť hľadaním maximálneho toku v sieti  $\vec{G}$ .

Vyriešiť základnú úlohu I. optimalizácie turnusov znamená nájsť v sieti  $\vec{G}$  maximálny tok. Potom spoj  $s_j$  bude zaradený bezprostredne za spoj  $s_i$  v jednom turnuse práve vtedy, keď  $\mathbf{y}(v_i, w_j) = 1$ .

Vyriešiť základnú úlohu II. optimalizácie turnusov znamená nájsť v sieti  $\vec{G}$  najlacnejší maximálny tok (resp. najlacnejší tok danej veľkosti). Potom spoj  $s_j$  bude zaradený bezprostredne za spoj  $s_i$  v jednom turnuse práve vtedy, keď  $\mathbf{y}(v_i, w_j) = 1$ .

## 7.9 Cvičenia

1. Ak v sieti  $\vec{G} = (V, H, c)$  existuje cyklus  $C$ , možno v nej zostrojiť tok  $\mathbf{y}$  s nulovou veľkosťou (t. j.  $\sum_{h \in H^+(z)} \mathbf{y}(h) = \sum_{h \in H^-(u)} \mathbf{y}(h) = 0$ ), avšak pre niektoré hrany (špeciálne hrany cyklu  $C$ ) môže byť  $\mathbf{y}(h) > 0$ . Nakreslite diagramy niekoľkých takých prípadov. Aký najjednoduchší príklad sa vám podarí nájsť?
2. Objasnite rozdiel medzi pojmami „ústie“ a „prameň“, „zdroj“ a „stok“.
3. Sieť môže obsahovať aj vrcholy, ktoré nie sú dosiahnuteľné zo zdroja, alebo z ktorých nie je dosiahnuteľné ústie. Aký význam majú tieto vrcholy (a s nimi incidentné hrany) pre výpočet maximálneho toku v sieti?
4. Ručným výpočtom nájdite maximálny tok vo vami navrhnutej sieti. Po nájdení maximálneho toku identifikujte rezovú množinu s minimálnou rezovou priepustnosťou.
5. Formulujte úlohu hľadania najdrahšieho toku v sieti a navrhните algoritmus na jej riešenie

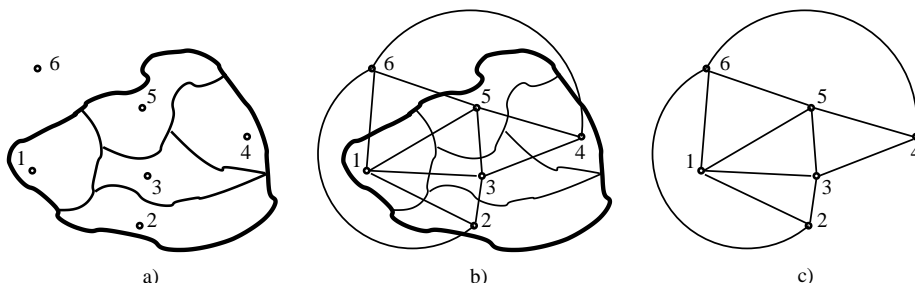
### Počítačové cvičenia

6. Napíšte program pre hľadanie maximálneho toku v sieti. Experimentujte s rôznymi spôsobmi hľadania zväčšujúcej polocesty (podľa algoritmu 7.2, najkratšia polocesta čo do počtu hrán, polocesta s najväčšou rezervou atď.)
7. Napíšte program pre hľadanie najlacnejšieho maximálneho toku.

# Kapitola 8

## Farbenie grafov

Z praktického hľadiska sú veľmi dôležité úlohy o farbení grafov. Tieto úlohy vznikli z praktických požiadaviek tlačiarň pri tlači politických máp. Pri farbení politických máp je treba farebne odlišiť územia jednotlivých štátov tak, aby žiadne dva štáty, ktoré majú spoločnú hranicu, neboli vyfarbené rovnakou farbou. Pri začiatkoch farebnej tlače bolo tlačenie obrázku tým ťažšie a nákladnejšie, čím viac farieb obrázok obsahoval. Preto ďalšou prirodzenou požiadavkou bolo, aby pri farbení mapy bol použitý minimálny počet farieb.



Obr. 8.1: Grafový model pre problém farbenia máp.

- každému štátu i moru priradíme vrchol grafu,
- pospájame vrcholy susedných štátov,
- diagram výsledného grafu.

K práve formulovanému problému si vytvoríme graf nasledovne: Každému štátu a tiež aj moru priradíme vrchol grafu  $G$  – tým máme definovanú množinu vrcholov  $V$ . Neusporiadanú dvojicu vrcholov  $u, v \in V$  prehlásime za hranu  $\{u, v\}$  práve vtedy, keď štáty  $u, v$  majú spoločnú hranicu.

Zafarbiť mapu minimálnym počtom farieb tak, aby žiadne dva z nich, ktoré majú spoločnú hranicu, nemali rovnakú farbu, znamená zafarbiť vrcholy grafu  $G$  minimálnym počtom farieb tak, aby žiadne dva susedné vrcholy nemali pridelenú rovnakú farbu.

Pretože grafy modelujúce problém farbenia máp sú rovinné, budeme sa najprv venovať rovinným grafom.

## 8.1 Rovinné grafy

Podľa definície 1.12 na strane 19 je graf **rovinný** práve vtedy, keď k nemu existuje rovinný diagram. **Rovinný diagram** grafu je taký diagram, v ktorom čiary zodpovedajúce hranám sa nepretínajú nikde okrem vrcholov. Obrázok 1.2 na strane 20, ktorý ukazuje dva diagramy toho istého grafu môže slúžiť ako príklad toho, že k rovinnému grafu možno zostrojiť aj nerovinný diagram.

V súvislosti so skúmaním vlastností rovinných grafov budeme používať viaceré pojmy z teórie rovinných útvarov. Na tomto mieste chcem upozorniť, že presná definícia mnohých takýchto útvarov by veľmi rozšírila rozsah tejto publikácie, pričom by neprinesla podstatné prínosy k teórii grafov. Preto mnohé z týchto pojmov budeme chápať intuitívne.

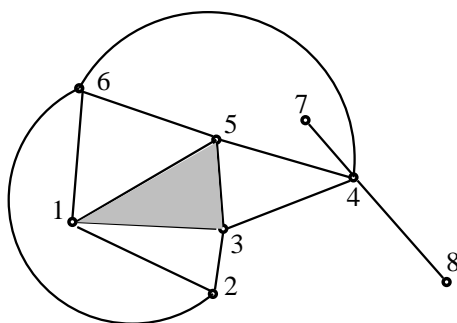
**Definícia 8.1.** **Stenou** rovinného diagramu nazveme maximálnu časť roviny, ktorej ľubovoľné dva body možno spojiť súvislou čiarou nepretínajúcou žiadnu hranu diagramu.

Máme steny dvoch druhov – práve jedna stena je neohraničená, a nazýva sa **vonkajšia stena**. Ostatné steny sa nazývajú **vnútorné**.

Všimnime si ešte, že vrcholy a hrany diagramu, ktoré vymedzujú ktorúkoľvek stenu, tvoria „cyklus“.<sup>1</sup>

V diagrame však vidíme aj také hrany – na obr. 8.2 sú to hrany  $\{4, 7\}$ ,  $\{4, 8\}$  – ktoré nevymedzujú žiadnu stenu. Hrana vymedzuje niektorú stenu práve vtedy,

<sup>1</sup>Presnejšie: Vrcholy a hrany grafu  $G$  zodpovedajúce vrcholom a hranám jeho diagramu, ktoré vymedzujú ktorúkoľvek stenu, tvoria cyklus v grafe  $G$ .



Obr. 8.2: Jedna stena rovinného diagramu.

Aj časť roviny ohraničená hranami  $\{4, 5\}$ ,  $\{5, 6\}$ ,  $\{6, 4\}$  je stena.

keď leží aspoň na jednom cykle. Vylúčením ktorejkoľvek hrany ležiacej na cykle klesne počet stien diagramu o 1.

**Veta 8.1. Eulerova polyedrická formula.** *Nech  $G = (V, H)$  je súvislý rovinný graf, nech  $S$  je množina stien jeho rovinného diagramu. Potom platí:*

$$|S| = |H| - |V| + 2. \quad (8.1)$$

DŔKAZ.

Matematickou indukciou podľa počtu stien rovinného grafu. Najjednoduchší súvislý graf s  $|V|$  vrcholmi je strom. V strome je  $|H| = |V| - 1$ . Rovinný diagram stromu má iba jedinú, a to vonkajšiu stenu – je teda  $|S| = 1$ . Počítajme  $|H| - |V| + 2 = (|V| - 1) - |V| + 2 = 1$ . Pre súvislé rovinné grafy s jedinou stenou (sú to práve stromy) tvrdenie vety platí.

Predpokladajme, že tvrdenie vety platí pre všetky súvislé rovinné grafy, ktorých diagramy majú  $s$  stien. Majme graf  $G = (V, H)$ , ktorého rovinný diagram má  $|S| = s + 1 > 1$  stien. V rovinnom diagrame grafu  $G$  existuje aspoň jedna hrana  $h$  vymedzujúca nejakú stenu. Hrana  $h$  musí ležať na nejakom cykle grafu  $G$ . Odstránením tejto hrany z diagramu dostaneme diagram grafu  $G - \{h\}$ , ktorý má  $s$  stien,  $|V|$  vrcholov a  $|H| - 1$  hrán. Podľa indukčného predpokladu platí

$$s = (|H| - 1) - |V| + 2,$$

čo je to isté ako

$$|S| = s + 1 = |H| - |V| + 2.$$

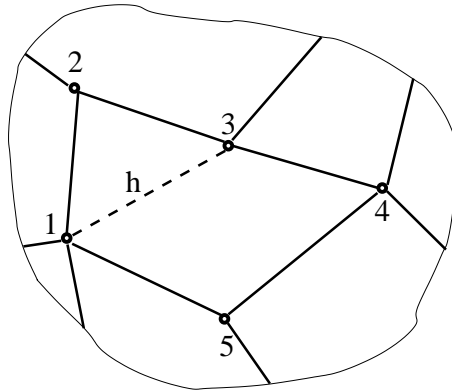


**Veta 8.2.** *Nech  $G = (V, H)$  je maximálny rovinný graf s množinou vrcholov  $V$ , kde  $|V| \geq 3$ . Potom*

$$|H| = 3 \cdot |V| - 6. \quad (8.2)$$

DŮKAZ.

Ak má byť  $G$  maximálny rovinný graf s  $n \geq 3$  vrcholmi, potom musí byť každá stena (včítane vonkajšej tzv. trojuholníkom – t. j. časťou roviny ohraničenou iba tromi hranami. Ak by totiž existovala stena ohraničená štyrmi alebo viac hranami – t. j.  $r$ -uholník, kde  $r > 3$ , dodaním hrany  $h$  príslušnej k niektorej uhlopriečke tohto  $r$ -uholníka dostaneme rovinný graf  $G \cup \{h\}$ , čo je v spore s maximalitou grafu  $G$ .



Obr. 8.3: Ak existuje stena, ktorá nie je trojuholník, (na tomto obrázku  $1, \{1, 2\}, 2, \{2, 3\}, 3, \{3, 4\}, 4, \{4, 5\}, 5, \{5, 1\}$ ), možno dodaním hrany  $h = \{1, 3\}$  zvýšiť počet stien diagramu grafu.

Diagram grafu  $G$  je teda tvorený  $s$  trojuholníkmi. Keby trojuholníky boli disjunktné (každá hrana len v jednom trojuholníku) potom by sme na ich vytvorenie potrebovali  $3 \cdot s$  hrán. Pretože však každá hrana je použitá v dvoch susedných trojuholníkoch, je v maximálnom grafe  $G$  presne  $\frac{3 \cdot s}{2}$  hrán, t. j.  $|H| = \frac{3 \cdot s}{2}$  alebo  $s = \frac{2}{3} \cdot |H|$ . Použitím Eulerovej polyedrickej formuly z vety

8.1 máme

$$\begin{aligned} |S| &= \frac{2}{3} \cdot |H| = |H| - |V| + 2 \\ 2 \cdot |H| &= 3 \cdot |H| - 3 \cdot |V| + 6 \\ |H| &= 3 \cdot |V| - 6 \end{aligned}$$

■

**Veta 8.3** (Dôsledok). *V každom rovinnom grafe  $G = (V, H)$ , kde  $V \geq 3$ , je*

$$|H| \leq 3 \cdot |V| - 6. \quad (8.3)$$

DÔKAZ.

Nech  $\overline{G} = (V, \overline{H})$  je maximálny rovinný graf obsahujúci ako faktorový podgraf graf  $G$ . Potom podľa (8.2) je  $|\overline{H}| = 3 \cdot |V| - 6$ . Pretože  $H \subseteq \overline{H}$  je  $|H| \leq |\overline{H}|$ , a preto  $H \leq 3 \cdot |V| - 6$ . ■

**Veta 8.4.** *Úplný graf s piatimi vrcholmi  $K_5$  nie je rovinný. Úplný bipartitný graf  $K_{3,3}$  nie je rovinný.*

DÔKAZ.

Úplný graf  $K_5$  má  $(5 \times 4)/2 = 10$  hrán a 5 vrcholov. Keby bol rovinný mohol, by mal podľa (8.3) najviac  $3 \times 5 - 6 = 9$  hrán, čo je spor.

Predpokladajme, že graf  $K_{3,3}$  je rovinný. Potom jeho diagram neobsahuje ani jeden trojuholník – t. j. všetky steny sú aspoň štvoruholníky.<sup>2</sup> Nech diagram grafu  $K_{3,3}$  má  $s$  stien. Ak by všetky štvor- a viac-uholníky boli disjunktné potrebovali by sme na ich konštrukciu aspoň  $4 \cdot s$  hrán. Pretože však v diagrame je každá hrana v dvoch viac-uholníkoch, potrebujeme aspoň  $4 \cdot s/2 = 2 \cdot s$  hrán, t. j.  $|H| \geq 2s$

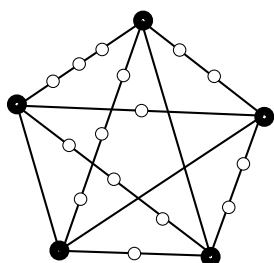
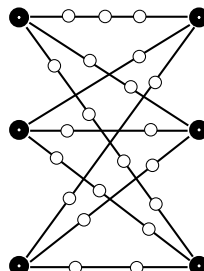
$$\begin{aligned} |H| &\geq 2 \cdot |S| = 2 \cdot |H| - 2 \cdot |V| + 2 \cdot 2 \\ -|H| &\geq -2 \cdot |V| + 4 \\ |H| &\leq 2 \cdot |V| - 4 \end{aligned}$$

---

<sup>2</sup>Predpokladajme, že v diagrame bipartitného grafu  $G = (V_1 \cup V_2, H)$  existuje trojuholník, k nemu príslušný cyklus nech je  $(v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \{v_3, v_1\}, v_1)$ . Ak  $v_1 \in V_1$ , potom  $v_2 \in V_2$ ,  $v_3 \in V_1$  a nakoniec  $v_1 \in V_2$ , čo je spor, lebo v bipartitnom grafe je  $V_1 \cap V_2 = \emptyset$ . Podobne sa dá ukázať, že každý cyklus v bipartitnom grafe má párny počet hrán (a párny počet vrcholov).





a) Graf homeomorfný s  $K_5$ b) Graf homeomorfný s  $K_{3,3}$ 

Obr. 8.5: Dva prototypy nerovinných grafov.

z prototypov nerovinnosti –  $K_5$  alebo  $K_{3,3}$ . Napriek matematickej elegancii Kuratowského vety algoritmus na rozhodnutie o rovinnosti grafu postavený na hľadaní podgrafov homeomorfných s  $K_5$  a  $K_{3,3}$  nie je efektívny.

## 8.2 Chromatické číslo a $k$ -zafarbitelnosť

**Definícia 8.3. Zafarbenie grafu** je funkcia, ktorá každému vrcholu grafu priraduje farbu. Zafarbenie, ktoré žiadnym dvom susedným vrcholom nepriraduje tú istú farbu nazveme **prípustným zafarbením**. Graf  $G = (V, H)$  nazveme  **$k$ -zafarbitelným**, ak jeho vrcholy možno prípustne zafarbiť  $k$  farbami (t. j. tak, aby žiadne dva susedné vrcholy neboli zafarbené rovnakou farbou.)

**Chromatické číslo grafu** je najmenšie prirodzené číslo  $k$  také, že graf  $G$  je  $k$ -zafarbitelný. Chromatické číslo grafu  $G$  budeme značiť symbolom  $\chi(G)$ .

*Poznámka.* Všimnime si, že ak je graf  $G$   $k$ -zafarbitelný, potom aj ľubovoľný jeho podgraf  $G' \subseteq G$  je  $k$ -zafarbitelný. Prípustné zafarbenie grafu  $G$  jednoznačne definuje prípustné zafarbenie ľubovoľného jeho podgrafu.

Majme graf  $G = (V, H)$  zafarbený  $k$  farbami tak, že žiadne dva susedné vrcholy nie sú zafarbené tou istou farbou. Relácia „vrchol  $v_i$  je zafarbený tou istou farbou ako vrchol  $v_j$ “ je reláciou ekvivalencie na množine vrcholov  $V$ , a preto definuje rozklad množiny  $V$  na triedy ekvivalencie. Počet neprázdnych tried tohto rozkladu je práve  $k$ . V každej triede ekvivalencie sú všetky vrcholy rovnakej farby. Z vlastnosti zafarbenia vyplýva, že žiadne dva vrcholy v tej istej triede rozkladu nie sú susedné – triedy rozkladu sú tzv. nezávislé množiny.

**Veta 8.6.** *Problém zafarbiť graf s minimálnym počtom farieb je NP-ťažký.*

**Algoritmus 8.1. Sekvenčné farbenie grafu.**

- **Krok 1.** Nech  $\mathcal{P} = v_1, v_2, \dots, v_n$  je ľubovoľná postupnosť vrcholov grafu  $G = (V, H)$ .
- **Krok 2.** Postupne pre  $i = 1, 2, \dots, n$  urob:  
Zafarbi vrchol  $v_i$  farbou najmenšieho čísla takou, že žiaden zo zafarbených susedov vrchola  $v_i$  nie je zafarbený touto farbou.



Algoritmus 8.1 nedáva zaručene optimálne zafarbenie grafu, možno ho použiť pre rýchle zistenie horného odhadu chromatického čísla grafu s využitím nasledujúcej vety.

**Veta 8.7.** *Algoritmus na sekvenčné farbenie grafu potrebuje na zafarbenie ľubovoľného grafu najviac*

$$\max\{\deg(v) \mid v \in V\} + 1 \quad (8.4)$$

*fariieb.*

**DŮKAZ.**

Algoritmus na sekvenčné farbenie grafu zafarbí vrchol 1 farbou 1. Nech už máme zafarbených  $i$  vrcholov  $v_1, v_2, \dots, v_i$  s použitím najviac  $(\max\{\deg(v) \mid v \in V\} + 1)$  farieb. Ďalší vrchol  $v_{i+1}$  algoritmus zafarbí farbou najnižšieho čísla, ktorá nie je priradená žiadnemu jeho susedovi. Keďže vrchol  $v_{i+1}$  môže mať najviac  $\max\{\deg(v) \mid v \in V\}$  susedov, medzi  $(\max\{\deg(v) \mid v \in V\} + 1)$  farbami ostáva aspoň jedna, ktorou môžeme zafarbiť vrchol  $v_{i+1}$ . ■

Dôsledkom tejto vety je nasledujúce tvrdenie.

**Veta 8.8.** *Pre chromatické číslo  $\chi(G)$  ľubovoľného grafu  $G$  platí:*

$$\chi(G) \leq 1 + \max\{\deg(v) \mid v \in V\} \quad (8.5)$$

**Veta 8.9.** *Graf  $G = (V, H)$  je 2-zafarbitelný práve vtedy, keď neobsahuje cyklus s nepárnym počtom hrán.*

DŔKAZ.

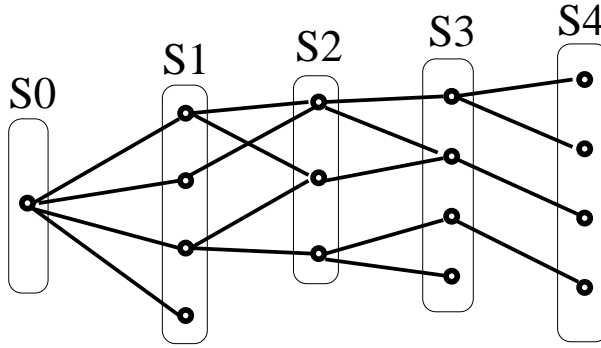
Uvedomme si, že existencia cyklu v grafe je ekvivalentná s existenciou kružnice indukovanej týmto cyklom.

Kružnica s nepárnym počtom hrán nie je 2-zafarbiteľná. Ak je graf  $G$  2-zafarbiteľný, potom aj každý jeho podgraf je 2-zafarbiteľný, a preto nemôže obsahovať cyklus s nepárnym počtom hrán.

Nech graf  $G$  neobsahuje cyklus s nepárnym počtom hrán. Chceme dokázať, že  $G$  je 2-zafarbiteľný. Ak by bol graf  $G$  nesúvislý, žiaden jeho komponent neobsahuje cyklus s nepárnym počtom hrán. Pritom zafarbenie niektorého komponentu nijako neovplyvňuje zafarbenie iného komponentu. Preto stačí dokázať vetu pre ľubovoľný komponent.

Predpokladajme teda, že  $G$  je súvislý. Vezmime ľubovoľný vrchol  $v$  a poloŹme  $S_0 = \{v\}$ ,  $S_1 = \{u \mid u \in V, \{v, u\} \in H\}$ . Ak už máme definované  $S_1, S_2, \dots, S_{k-1}$ , definujeme

$$S_k = \{u \mid u \in V - \bigcup_{i=0}^{k-1} S_i, \exists w \in S_{k-1} \{w, u\} \in H\} \quad (8.6)$$



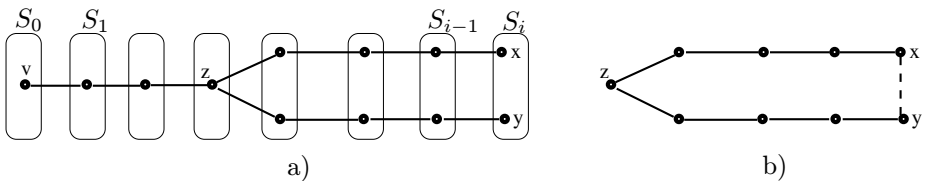
Obr. 8.6: MnoŹiny  $S_i$ .

Z definície množín  $S_i$  vyplýva, že pre  $i \neq j$  je  $S_i \cap S_j = \emptyset$ . Najneskôr pre  $k = n$  je  $S_k = \emptyset$ . Pretože  $G$  je súvislý, každý vrchol sa objaví v niektorej množine  $S_i$  práve raz.

UkáŹeme, že  $S_i$  je množina práve tých vrcholov, ktoré majú od vrchola  $v$  vzdialenosť  $i$  (pri jednotkovej dĺŹke hrán). Pre  $S_1$  tvrdenie platí. Nech tvrdenie

platí pre  $i = 1, 2, \dots, k - 1$ . Nech  $u \in S_k$ . Keby  $u$  malo menšiu vzdialenosť od  $v$  ako  $k$ , bolo by podľa indukčného predpokladu  $u \in S_j$  pre nejaké  $j < k$  a potom by nemohlo byť aj  $u \in S_k$ . Pretože existuje  $w \in S_{k-1}$  také, že  $\{w, u\} \in H$  a pretože existuje  $v-w$  cesta dĺžky  $k-1$ , táto  $v-w$  cesta predĺžená o hranu  $\{w, u\}$  dá cestu  $v-u$  cestu dĺžky  $k$ . Ukázali sme, že každý vrchol z množiny  $S_k$  má od vrchola  $v$  vzdialenosť  $k$ . Ukážme ešte, že ak  $d(v, x) = k$ , potom  $x \in S_k$ . Nech  $y$  je predposledný vrchol najkratšej  $v-x$  cesty s dĺžkou  $k$ , potom  $d(v, y) = k - 1$  a podľa indukčného predpokladu  $y \in S_{k-1}$ . Keďže existuje hrana  $\{y, x\}$  a  $x$  doteraz nebolo zaradené v žiadnej množine  $S_m$ ,  $m < k$ , je  $x \in S_k$ .

Ďalej si všimnime, že neexistuje žiadna hrana typu  $\{x, y\}$ ,  $x \in S_j$ ,  $y \in S_i$ , kde  $j \leq i - 2$ , pretože všetky susedné vrcholy s vrcholom  $x$  sú buď v množinách typu  $S_k$ , kde  $k \leq j$  (ak by  $y \in S_k$ , potom by  $i = k \leq j$ ) alebo v množine  $S_{j+1}$  (keby  $y \in S_{j+1}$ , potom by  $i = j + 1$ ). Doteraz sme ešte predpoklad neexistencie cyklu nepárnej dĺžky nepotrebovali.



Obr. 8.7: Z existencie hrany medzi vrcholmi množiny  $S_i$  vyplýva existencia cyklu nepárnej dĺžky.

- a) ak  $x \in S_i$ ,  $y \in S_i$ , potom obe najkratšie cesty  $\mu(v, x)$ ,  $\mu(v, y)$  majú rovnaký počet hrán  $i$
- b) cesty  $\mu(z, x)$ ,  $\mu(z, y)$  majú tiež rovnaký počet hrán a spolu s hranou  $\{x, y\}$  vytvárajú cyklus nepárnej dĺžky.

Nakoniec ukážeme, že žiadne dva prvky z množiny  $S_i$  nie sú susedné. Nech existujú  $x \in S_i$ ,  $y \in S_i$  také že  $\{x, y\} \in H$  pozri obrázok 8.7. Keďže  $x, y \in S_i$ , existujú cesty  $\mu(v, x)$ ,  $\mu(v, y)$  dĺžky  $i$ . Nech  $z$  je posledný spoločný vrchol týchto ciest. Časť cesty  $\mu(v, x)$  počínajúc vrcholom  $z$  a končiac vrcholom  $x$  (označme ju  $\mu(z, x)$ ) má rovnaký počet hrán ako časť cesty  $\mu(v, y)$  počínajúc vrcholom  $z$  a končiac vrcholom  $y$ , ktorú označíme  $\mu(z, y)$ . Zreťazenie cesty  $\mu(z, x)$  s jednohranovou cestou  $x, \{x, y\}, y$  a opačne vzatej cesty  $\mu(z, y)$  vytvára cyklus s nepárnym počtom hrán, čo je v spore s predpokladom, že v grafe  $G$  neexistuje cyklus nepárnej dĺžky.

Označme  $V_1$  zjednotenie všetkých  $S_i$  s nepárnymi indexami a  $V_2$  zjednotenie všetkých  $S_i$  s párnymi indexami. Žiadne dva vrcholy z  $V_1$  nie sú susedné, podobne žiadne dva vrcholy z  $V_2$  nie sú susedné, a preto môžeme zafarbiť vrcholy z  $V_1$  farbou 1 a vrcholy z  $V_2$  farbou 2. Keďže  $V = V_1 \cup V_2$ , je toto zafarbenie 2-zafarbením grafu  $G$ . ■

**Veta 8.10. Dôsledky.** *Každý strom je 2-zafarbiteľný. Každý bipartitný graf je 2-zafarbiteľný.*

DÔKAZ.

Žiaden z týchto grafov neobsahuje cyklus nepárnej dĺžky. ■

**Veta 8.11. Appel, Haken, 1976.** *Každý rovinný graf je 4-zafarbiteľný.*

Dlho pred dokázaním tejto vety sa vedelo, že každý rovinný graf je 5-zafarbiteľný. Nenašiel sa však žiaden rovinný graf  $G$  s chromatickým číslom  $\chi(G) = 5$ . Veta o 4-zafarbitelnosti rovinných grafov je jedna z prvých, na dokázanie ktorej bol použitý počítač. Počítačový postup navrhol pôvodne Heesch, Appel a Haken zredukovali problém na skontrolovanie viac ako 1900 konfigurácií. Na vyriešenie problému sa spotrebovalo viac ako 1200 hodín strojového času. Dnes sú počítače takmer o tri rády rýchlejšie, ale aj tak by tento výpočet vyžadoval výpočtový čas meraný v hodinách.

### 8.3 Heuristiky pre farbenie grafu

Keďže problém zafarbenia grafu minimálnym počtom farieb je NP-ťažký, na jeho riešenie pri úlohách väčšieho rozmeru používame heuristiky. Jednou z nich je algoritmus na sekvenčné farbenie grafu uvedený v časti 8.2.

Algoritmus sekvenčného farbenia grafu postupne vyberal vrcholy a farbil ich najnižšou možnou farbou. Nasledujúci algoritmus zoberie jednu farbu a ňou zafarbí pokiaľ možno najväčší počet vrcholov grafu. Potom vezme ďalšiu farbu a zafarbí ňou ďalšie vrcholy atď. Je založený na domnienke, že najprv treba zafarbiť vrcholy s najväčším stupňom.

**Algoritmus 8.2. Paralelné farbenie grafu.**

- **Krok 1.** Zoraď vrcholy grafu  $G = (V, H)$  do postupnosti  $\mathcal{P} = v_1, v_2, \dots, v_n$  podľa stupňa vrchola nerastúco. Inicializuj množinu farieb  $\mathcal{F} := \{1\}$ ,  $j := 1$ .

- **Krok 2.** Postupne s prvkami  $v_1, v_2, \dots, v_n$  postupnosti  $\mathcal{P}$  urob: Ak vrchol  $v_i$  nie je zafarbený a nemá suseda zafarbeného farbou  $j$ , tak ho farbou  $j$  zafarbi.
- **Krok 3.** Ak sú všetky vrcholy postupnosti  $\mathcal{P}$  zafarbené, STOP.
- **Krok 4.** Ak nie sú všetky vrcholy postupnosti  $\mathcal{P}$  zafarbené, zväčš počet farieb, t. j.  $j:=j+1$ ,  $\mathcal{F} := \mathcal{F} \cup \{j\}$  a GOTO Krok 2.



Nasledujúci algoritmus je v podstate sekvenčný algoritmus, ktorý si však v priebehu výpočtu stanovuje, ktorému z vrcholov sa bude pridelať najnižšia prideliteľná farba.

### Algoritmus 8.3. Farbenie grafu LDF (Largest Degree First).

Pre účel tohto algoritmu definujeme farebný stupeň vrchola  $v$  ako počet rôznych farieb, ktorými sú zafarbení susedia vrchola  $v$ .

- **Krok 1.** Zo všetkých nezafarbených vrcholov s najväčším stupňom vyber vrchol  $v$  s najväčším farebným stupňom.
- **Krok 2.** Prirad' vrcholu  $v$  farbu najnižšieho možného čísla.
- Ak sú všetky vrcholy zafarbené, STOP. Inak GOTO **Krok 1**.



Uvedené algoritmy sú veľmi jednoduché. Nemajú žiadne opravné kroky, keď raz vrcholu pridelia farbu, toto pridelenie je definitívne. Bolo by ich možné modifikovať tak, že by sa menilo vstupné poradie vrcholov, ktorým sa pridieva farba.

## 8.4 Exaktný algoritmus na farbenie grafov

Na zafarbenie grafu  $G = (V, H)$  potrebujeme v najhoršom prípade  $n = |V|$  farieb (pre úplné grafy  $K_n$ ). Zafarbenie grafu  $G$  je vlastne funkcia  $n$ -prvkovej množiny  $V$  do množiny  $\{1, 2, \dots, n\}$ . Takýchto zobrazení je  $n^n$ , nie každé z nich však spĺňa podmienku, že žiadni dvaja susedia v grafe  $G$  nie sú zafarbení tou istou farbou. Ak by sme však chceli riešiť problém zafarbenia grafu  $G$  minimálnym počtom farieb prezretím všetkých možností, museli by

sme prezrieť všetky zobrazenia  $\phi : V \rightarrow \{1, 2, \dots, n\}$ , z nich vylúčiť tie, ktoré nedefinujú prípustné zafarbenie grafu  $G$  a z tých, ktoré ostanú vybrať jedno s najmenej početným oborom hodnôt. Pre grafy s maximálnym stupňom  $d = \max\{\deg(v) \mid v \in V\}$  stačí namiesto množiny farieb  $\{1, 2, \dots, n\}$  uvažovať jej podmnožinu  $\{1, 2, \dots, d, d+1\}$ , čím sa zníži počet zobrazení na  $(d+1)^n$ , ale ani toto nedáva veľké nádeje na použitie úplného prehľadávania.

Zoradíme vrcholy grafu  $G$  do postupnosti  $v_1, v_2, \dots, v_n$  a toto poradie považujeme v rámci celej tejto časti za pevné. Pri pevnom poradí vrcholov možno zafarbenie grafu reprezentovať postupnosťou farieb  $f_1, f_2, \dots, f_n$ , kde  $f_i$  je farba priradená vrcholu  $v_i$  pre  $i = 1, 2, \dots, n$ . K ľubovoľnému zafarbeniu  $f_1, f_2, \dots, f_n$  grafu  $G$  možno zostrojiť zafarbenie  $f'_1, f'_2, \dots, f'_n$  takto:

$$f'_1 = 1$$

$$f'_2 = \begin{cases} 1 & \text{ak } f_1 = f_2 \\ 2 & \text{ak } f_1 \neq f_2 \end{cases}$$

$$f'_k = \begin{cases} f'_j & \text{ak } f_k = f_j \text{ pre niektoré } 1 \leq j < k \\ \max\{f_j \mid 1 \leq j < k\} + 1 & \text{ak } f_k \neq f_j \text{ pre všetky } j \text{ také, že } (1 \leq j < k) \end{cases}$$

Je ľahko vidieť, že aj  $f'_1, f'_2, \dots, f'_n$  je zafarbenie grafu  $G$  s rovnakým počtom farieb ako pôvodné zafarbenie  $f_1, f_2, \dots, f_n$  a platí: Vrcholy  $v_i, v_j$  sú rovnako zafarbené pri farbení  $f$  práve vtedy, keď sú rovnako zafarbené pri farbení  $f'$ , t. j.  $f_i = f_j$  práve vtedy, keď  $f'_i = f'_j$ . Obe zafarbenia definujú ten istý rozklad množiny  $V$  na triedy rovnako zafarbených vrcholov.

Ak fixujeme poradie vrcholov, potom pre ľubovoľný rozklad množiny  $V$  na triedy rovnako zafarbených vrcholov existuje jediné zafarbenie vrcholov  $f_1, f_2, \dots, f_n$  také, že každá čiastočná podpostupnosť typu  $f_1, f_2, \dots, f_k$ , kde  $1 \leq k \leq n$  obsahuje všetky prirodzené čísla  $1, 2, \dots, \max\{f_j \mid 1 \leq j \leq k\}$ . Prípustné zafarbenie  $f$  s práve opísanou vlastnosťou budeme nazývať **systematické zafarbenie**.

Systematických zafarbení je už podstatne menej ako všetkých zafarbení – vrchol  $v_1$  môže mať len farbu 1, vrchol  $v_2$  nemôže byť zafarbený farbou 3 alebo vyššou, všeobecne vrchol  $v_i$  nemôže byť zafarbený farbou  $k$ , kde  $k > i$ .

Pre navrhovaný farbiaci algoritmus stotožníme vrcholy množiny  $V$  s ich indexami, budeme predpokladať, že  $\{v_1, v_2, \dots, v_n\} = \{1, 2, \dots, n\}$ . Nech  $V_x$  je množina všetkých susedov vrchola  $x$ , definujeme

$$P(x) = V_x \cap \{1, 2, \dots, x-1\}. \quad (8.7)$$

Sústavou množín  $P(x)$ , kde  $x \in V$ , je graf  $G$  pre účely farbenia dostatočne opísaný.

Označme  $G_1$  podgraf grafu  $G$  indukovaný množinou  $\{1\}$ ,  $G_2$  podgraf grafu  $G$  indukovaný množinou  $\{1, 2\}$ , atď.,  $G_k$  podgraf grafu  $G$  indukovaný množinou  $\{1, 2, \dots, k\}$  pre  $1 \leq k \leq n$ . Ak máme systematické zafarbenie  $f_1, f_2, \dots, f_n$  grafu  $G$ , potom  $f_1, f_2, \dots, f_k$  je systematické zafarbenie grafu  $G_k$  pre všetky  $1 \leq k \leq n$ .

Optimálne riešenie budeme hľadať v pomyselnom koreňovom strome možných riešení  $T$ . Aby sme minimalizovali prehľadávanie, budeme chcieť, aby strom  $T$  modeloval len systematické zafarbenia. Keďže sa zaujímate o zafarbenie grafu  $G$  minimálnym počtom farieb, budeme v strome modelovať len zafarbenia s  $FMAX = \max\{\deg(v) \mid v \in V\} + 1$  farbami.

Každý vrchol stromu bude mať priradenú farbu. Koreň stromu zodpovedajúci vrcholu 1 bude mať priradenú farbu 1, prvá úroveň stromu  $T$  bude zodpovedať možným zafarbeniam vrchola 2 a každá ďalšia úroveň  $k$  bude hovoriť o možných zafarbeniach vrchola  $k + 1$ .

Koreň stromu so svojou značkou farby 1 zodpovedá systematickému zafarbeniu grafu  $G_1$  (keďže  $G_1$  je triviálny graf, také zafarbenie existuje jediné). Nech na úrovni  $k - 1$  každý vrchol  $t$  stromu  $T$  zodpovedá systematickému zafarbeniu grafu  $G_k$ , nech značky farieb vrcholov cesty z koreňa do tohto vrchola definujú systematické zafarbenie grafu  $G_k$ . Pre vrchol  $t$  na úrovni  $k - 1$  definujeme prvého pravého následníka so značkou najnižšej farby, ktorá sa nevyskytuje v zafarbeniach vrcholov z  $P(x)$ , ďalších pravých následníkov definujeme a označíme značkou ďalšej najnižšej nepoužitej farby len vtedy, ak takýmto zafarbením vrchola  $k + 1$  vznikne systematické zafarbenie grafu  $G_{k+1}$ .

Prvá vetva sprava v strome  $T$  zodpovedá sekvenčnému zafarbeniu grafu  $G$  s najnižším číslom farby. Položme  $FMAX :=$  počet farieb a zároveň najvyššie číslo farby použité pri sekvenčnom zafarbení grafu  $G$ .

Nech  $y$  je prvý vrchol na ceste z koreňa do vrchola definujúceho súčasné riešenie, ktorý je zafarbený farbou  $FMAX$ . Aby sme znížili farbu vrchola  $y$ , musíme zmeniť (t. j. zvýšiť) farbu niektorého z vrcholov  $x$  z množiny  $P(y)$ . Aby sme zabezpečili, že preskúmame najbližšiu nádejnú nepreskúmanú pravú vetvu stromu  $T$ , zvolíme za  $x$  posledný vrchol z  $P(y)$ . Ak farbu vrchola  $x$  nemožno zvýšiť na hodnotu menšiu než  $FMAX$ , skúsime vrchol  $x - 1$ ,  $x - 2$  atď. Ak sa pre žiadny vrchol nepodarí zvýšiť jeho farbu, máme optimálne riešenie.



Ak takýto vrchol  $x$  nájdeme, zvýšime jeho farbu na najnižšiu prípustnú hodnotu a počínajúc vrcholom  $x + 1$  postupne prefarbíme všetky vrcholy najnižšou možnou farbou (ako v algoritme sekvenčného farbenia). Ak pre niektorý vrchol použijeme farbu  $FMAX$ , ďalej farby neprideľujeme, ale ho zoberieme za nový vrchol  $y$ . Nie je totiž šanca nájsť v tejto vetve zlepšenie.

Ak sa týmto spôsobom podarí zafarbiť všetky vrcholy bez použitia farby  $FMAX$ , máme zlepšenie a nový rekord si zapamätáme. Zároveň zaktualizujeme  $FMAX$  – položíme ho číslu najvyššej použitej farby pri novom rekorde. Nájdeme  $y$  prvý vrchol v rekordnom riešení s farbou  $FMAX$  a opakujeme algoritmus.

Obrovskou výhodou tohto postupu je, že nemusíme v pamäti udržiavať strom  $T$ . Tým, že sa stále držíme v prvej nepreskúmanej vetve sprava a po jej preskúmaní vieme prejsť do jej najbližšej susednej vetvy, stačí držať v pamäti rekord a tvar súčasnej vetvy, ktorý je jednoznačne daný systematickým zafarbením grafu  $G$ . Takto celý algoritmus okrem nárokov na uloženie grafu formou množín  $P(x)$  vyžaduje len dve celočíselné polia rozmeru  $n = |V|$  a niekoľko jednoduchých pomocných premenných.

Pracovné farby vrcholov budeme držať v poli  $B[\ ] - B[x]$  je farba vrchola  $x$ , najlepšie nájsené zafarbenie v poli  $REKORD[\ ]$ . Súčasný najlepší dosiahnutý počet farieb bude v premennej  $FMAX$ .

**Algoritmus 8.4. Exaktný algoritmus na zafarbenie grafu minimálnym počtom farieb.**

$$\text{Označme } P(x) = V_x \cap \{1, 2, \dots, x - 1\}, \quad (8.8)$$

$$\mathcal{F}(x) = \min\{i \mid 1 \leq i, \forall j \in P(x) \ i \neq B[j]\} \quad (8.9)$$

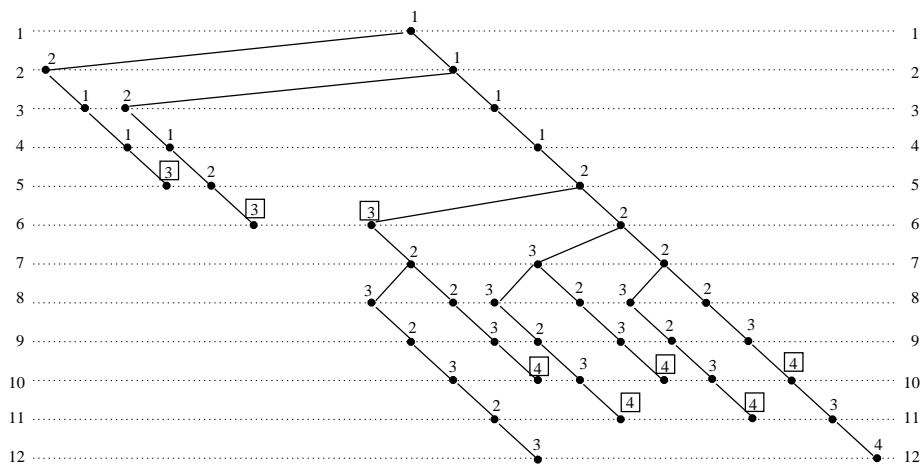
$$\mathcal{G}(x) = \min\{i \mid B[x] < i, \forall j \in P(x) \ i \neq B[j]\} \quad (8.10)$$

$\{\mathcal{F}(x)$  je najnižšie číslo farby, ktorou možno prípustne zafarbiť vrchol  $x$  v grafe  $G_x$ , ak sú vrcholy  $1, 2, \dots, (x - 1)$  zafarbené farbami porade  $B[1], B[2], \dots, B[x - 1]$ ..}

$\{\mathcal{G}(x)$  je najnižšie číslo farby väčšie ako  $B[x]$ , ktorou možno prípustne zafarbiť vrchol  $x$  v grafe  $G_x$ , ak sú vrcholy  $1, 2, \dots, (x - 1)$  zafarbené farbami porade  $B[1], B[2], \dots, B[x - 1]$ ..}

- **Krok 0.** Polož  $B[1] := 1$  a postupne pre každé  $x = 2, 3, \dots, n$   $B[x] := \mathcal{F}(x)$ .
- **Krok 1.** Polož  $FMAX := \max\{B[1], B[2], \dots, B[n]\}$ .  
Skopíruj pole  $B[\ ]$  do poľa  $REKORD[\ ]$ .

- **Krok 2.** Nájdi v poli  $B[ ]$  najmenšie  $y$  také, že  $B[y] = FMAX$ .
- **Krok 3.** Polož  $x := \max P(y)$ .
- **Krok 4.** Ak  $x = 1$ , STOP. Chromatické číslo grafu  $\chi(G) = FMAX$  a príslušné optimálne zafarbenie grafu  $G$  je v poli  $REKORD[ ]$ .
- **Krok 5.** Ak  $\mathcal{G}(x) \geq FMAX$  alebo ak  $\mathcal{G}(x) > (\max\{B[i] \mid 1 \leq i < x\} + 1)$ , polož  $x := x - 1$  a GOTO Krok 4. Ináč polož  $B[x] := \mathcal{G}(x)$ ,  $z := x + 1$ .
- **Krok 6.**  $B[z] := \mathcal{F}(z)$ . Ak  $B[z] \geq FMAX$ , polož  $y := z$  a GOTO Krok 3. Ak  $z < n$ , polož  $z := z + 1$  a opakuj Krok 6. Ak  $z = n$  máme nový rekord. GOTO Krok 1.



1

Obr. 8.8: Postup prezerania stromu riešení  $T$  pre farbenie grafu z príkladu 8.1.

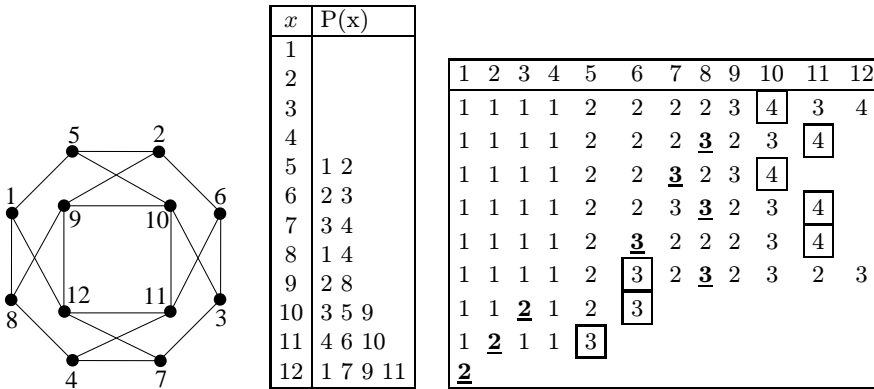
**Príklad 8.1.** Hľadáme optimálne zafarbenie grafu  $G$  daného diagramom z obrázku 8.9 vľavo pomocou algoritmu 8.4. Reprézntácia grafu  $G$  pomocou množín

$P(x)$  je v tabuľke na obrázku 8.9 v strede. Zafarbenie vrcholov budeme ukladať v poli  $B[\ ]$ , ktorého vývoj budeme zapisovať do tabuľky. Jej tvar je zrejmy z obrázku 8.9 vpravo.

Každý riadok tabuľky zodpovedá stavu poľa  $B[\ ]$  po vykonaní kroku 2. algoritmu 8.4. Značka (farba)  $FMAX$  pre najmenšie  $y$  také, že  $B[y] = FMAX$  je označená rámečkom. V nasledujúcom riadku je tučným fontom vyznačená značka prvého vrchola odzadu, ktorú možno zvýšiť v kroku 5, t. j. pre ktorý sa v kroku 5 vykonalo  $B[x] := \mathcal{G}(x)$ . Počnúc týmto vrcholom sa budú meniť značky vrcholov v poli  $B[\ ]$ .

Každý riadok tabuľky z obrázku 8.9 zodpovedá jednej vetve stromu riešeni na obrázku 8.8, pričom prvý riadok zodpovedá prvej vetve stromu  $T$  sprava, druhý riadok druhej vetve sprava atď.

Štvrtý riadok tabuľky zdola zodpovedá nájdeniu zlepšenia – graf sme po prvýkrát zafarbili troma farbami. Tento riadok si zapamätáme do poľa  $REKORD[\ ]$ . Keďže sme v ďalšom postupe nenašli zlepšenie, tento riadok obsahuje optimálne zafarbenie grafu  $G$ . Ako vo väčšine príkladov, počítač vystačí na udržiavanie aktuálnych farieb vrcholov s poľom  $B[\ ]$  a s poľom  $REKORD[\ ]$  na uchovávanie doteraz najlepšieho zafarbenia. Zápis v tabuľke je vhodný iba pre malý ručný výpočet a na ilustráciu postupu algoritmu.



Obr. 8.9: Vľavo graf  $G$ ,  
 v strede jeho reprezentácia množinami  $P(x) = V_x \cap \{1, 2, \dots, x - 1\}$ ,  
 vpravo tabuľka s výpočtom optimálneho zafarbenia grafu  $G$ .

Algoritmus 8.4 na exaktné farbenie grafu má nepolynomiálnu zložitosť. Je preto možné, že sa pri niektorých väčších grafoch nedočkáme jeho skončenia. V takomto prípade ho môžeme využiť ako heuristiku – predčasne ho ukončíme a riešenie v poli *REKORD*[ ] vezmeme ako suboptimálne riešenie. Keďže algoritmus štartuje zo sekvenčného zafarbenia grafu, jeho výsledok je lepší alebo nanajvýš rovnaký ako výsledok sekvenčného farbenia grafu.

## 8.5 Aplikácie

### 8.5.1 Priradenie registrov počítača

Počítače majú konečný počet špeciálnych registrov. Aritmetické operácie s údajmi v týchto registroch sú rýchlejšie, než operácie s údajmi v pamäti. Programátor má možnosť deklarovať niektoré premenné, ktoré sa veľmi často používajú, ako registre, čím sa program s takto definovanými premennými zrýchli. Avšak ak programátor deklaruje viac registrových premenných ako je registrov, môže sa stať, že pri výkone programu sa viac času premrhá prepisovaním obsahu pamäte medzi registrami, než sa získa použitím týchto registrov. Jedným z riešení je priradenie rôznych registrov premenným, ktoré sa súčasne používajú. Grafový model pre tento problém má za vrcholy premenné. Dve premenné sú spojené hranou práve vtedy, keď sú príslušné premenné súčasne aktívne. Chromatické číslo tohto grafu sa rovná počtu registrov potrebných na to, aby sa predišlo časovým stratám z nadmerného menenia obsahu registrov.

### 8.5.2 Priradenie rádiových frekvencií

Ak sú dva vysielacie blízko seba, nemôžu používať tú istú frekvenciu, pretože by sa navzájom rušili. Rádiové frekvencie sú však obmedzeným prírodným zdrojom. Preto nemožno každému vysielaciu priradiť inú frekvenciu. Modelom tejto situácie je graf  $G$ , ktorého vrcholy sú vysielacie a v ktorom za susedné vrcholy považujeme tie dvojice vysieláčov, ktoré by sa pri pridelení rovnakej frekvencie rušili. Úloha prideliť danej množine vysieláčov čo najmenej rôznych frekvencií tak, aby sa žiadne dva navzájom nerušili sa tak prevedie na úlohu zafarbiť vrcholy grafu  $G$  minimálnym počtom farieb (frekvencií) tak, aby žiadne dva susedné vrcholy (vysielacie, ktoré by sa rušili) nemali pridelenú tú istú farbu (frekvenciu).

Graf  $G$  na riešenie problému pridelovania frekvencií nemusí byť a ani vo väčšine prípadov nie je rovinný, hoci by tomu mohlo naznačovať rozmiestnenie vysieláčov v rovine. Rôzny výkon vysieláčov, prírodné podmienky šírenia rádiových vln, prekážky, odrazy spôsobujú, že vzájomné ovplyvňovanie sa vysieláčov je veľmi zložité, a preto sa nedá graf  $G$  čo do zložitosti porovnať s grafom pre farbenie rovinných máp.

### 8.5.3 Problém nákupných tašiek

Ideme nakupovať  $n$  produktov. Zo skúsenosti vieme, že niektoré tovary neradno nosiť spolu v jednej taške. Nová košeľa a slanina, čerstvé pečivo a prášky na pranie, sáčkové mlieko a klince by nemali byť v jednej taške. Takéto tovary nazveme nekompatibilné. Koľko nákupných tašiek najmenej potrebujeme, aby sme do nich mohli poukladať  $n$  tovarov tak, aby sa žiaden tovar nepoškodil? Problém riešime tak, že zostrojíme graf  $G$ , ktorého vrcholy budú tovary a v ktorom dva vrcholy (tovary) budú spojené hranou práve vtedy, keď ich nemožno dať do jednej nákupnej tašky. Tak sme problém minimalizácie počtu nákupných tašiek previedli na problém zafarbenia grafu  $G$  minimálnym počtom farieb. Každému vrcholu (tovaru) treba určiť farbu (tašku, do ktorej ho uložíme) tak, aby žiadne dva susedné vrcholy nemali tú istú farbu (aby žiadne dva nekompatibilné tovary neboli uložené v jednej taške) a aby bol počet použitých farieb (tašiek) minimálny.

Táto úloha má veľa variantov. Chemikálie treba uložiť do najmenšieho možného počtu kontajnerov tak, aby sa nedostala do jedného kontajnera žiadna dvojica chemikálií, ktorá by mohla spôsobiť výbuch. Biologické preparáty treba uložiť do najmenšieho možného počtu chladiacich boxov tak, aby sa žiadne dva, ktoré sa môžu ovplyvniť, nedostali do toho istého boxu. Z danej množiny pracovníkov treba vytvoriť čo najmenší počet pracovných skupín tak, aby sa v každej skupine dobre znášali.

Kontrolná otázka: Aký je minimálny počet politických strán taký, aby žiadni dvaja na seba alergickí politici neboli v jednej strane? Pretože hlavnou činnosťou politikov je byť alergický na príslušníkov iných opozičných a neskôr aj koalíčných strán, a pretože alergie vznikajú aj pri mocenskom boji vnútri strán, vidíme, že ani použitie najlepších algoritmov na farbenie grafu tu nezabráni vzniku nových a nových politických strán.

### 8.5.4 Rozvrhovanie voliteľných predmetov

Predpokladajme, že každý študent fakulty si môže zapísať niekoľko vybraných voliteľných predmetov. Fakulta venuje voliteľným predmetom niekoľko vyhradených blokov. Pretože fakulta má záujem, aby všetci študenti mohli navštevovať vybrané predmety, treba umiestniť tieto predmety do blokov tak, aby žiadnemu študentovi predmety nekolidovali. Je to vôbec možné? (Predpokladá sa dostatok učební).

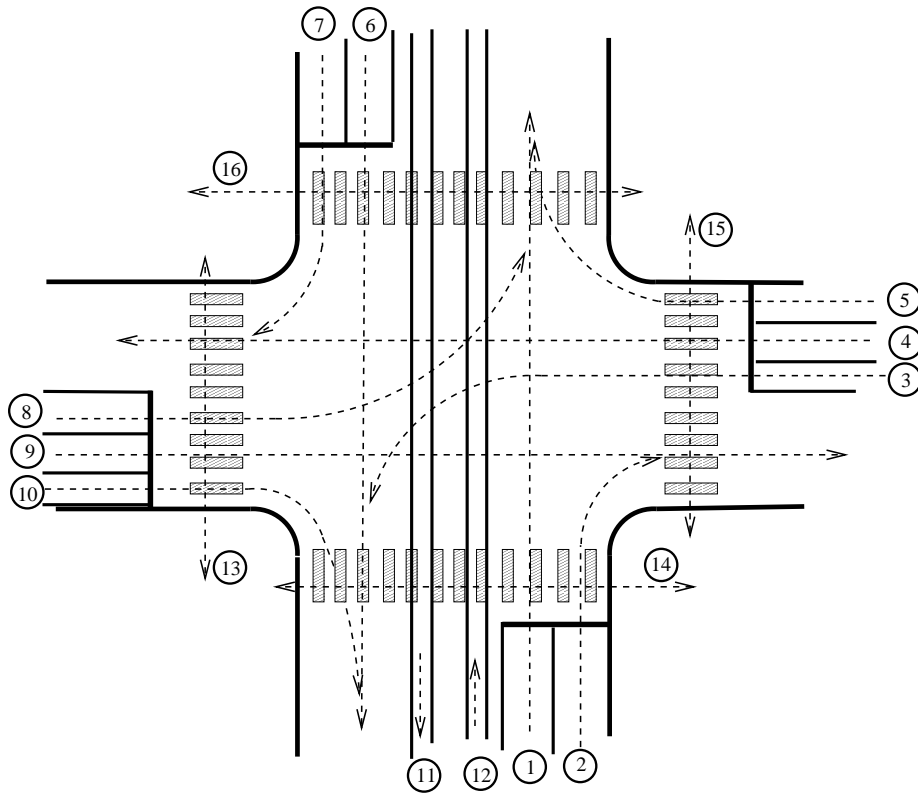
Na riešenie tejto úlohy zostrojíme graf  $G$  ktorého vrcholmi budú voliteľné predmety. Dvojica predmetov bude tvoriť hranu grafu  $G$  práve vtedy, keď aspoň jeden študent má zapísané obidva predmety. Chromatické číslo  $\chi(G)$  grafu  $G$  nám povie, aký je minimálny počet blokov potrebný na rozvrh bez konfliktov medzi jednotlivými voliteľnými predmetmi. Zafarbenie grafu  $G$  minimálnym počtom farieb (blokov) nám povie, ktoré voliteľné predmety treba zaradiť do rozvrhu v rovnakom čase.

Problém má niekoľko variácií. Pre povinné predmety sú skupiny poslucháčov disjunktné, avšak dva predmety môžu spolu kolidovať preto, lebo majú toho istého učiteľa, alebo preto, lebo vyžadujú špecializovanú učebňu či laboratórium.

### 8.5.5 Fázovanie svetelne riadenej križovatky

Konštrukcia programu pre riadenie svetelne riadenej križovatky začína geometrickým plánom križovatky. Jeden takýto plán vidíme na obrázku 8.10. Vozidlá prechádzajú križovatku v prúdoch. Na našej križovatke máme desať vozidlových prúdov  $1, 2, \dots, 10$ , dva električkové prúdy  $11, 12$  a štyri prúdy chodcov  $13, 14, 15, 16$ . Medzi prúdmi existuje relácia kolíznosti – dva prúdy  $i, j$  sú kolízne, ak ich dráhy majú spoločný bod zvaný tiež kolízny bod. Kolízne prúdy nemôžu vchádzať do križovatky naraz, pretože v kolíznych bodoch by sa vozidlá jednotlivých prúdov stretali a vznikali by tak nebezpečné situácie.

Tak napríklad prúdy  $1$  a  $2$  (sú to prúdy vozidiel) sú nekolízne, ale dvojice prúdov  $\{1, 3\}$ ,  $\{1, 4\}$ ,  $\{1, 5\}$ ,  $\{1, 8\}$ ,  $\{1, 9\}$ , sú kolízne dvojice vozidlových prúdov. Navyiac prúd  $1$  ešte koliduje s chodeckými prúdmi  $14$  a  $16$ .

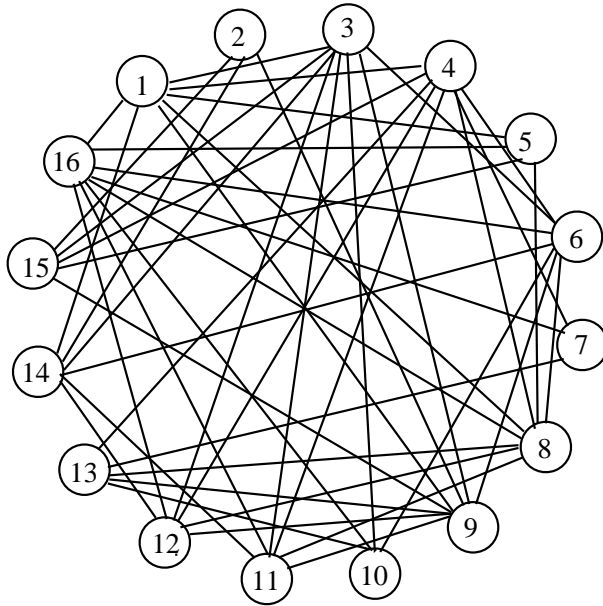


Obr. 8.10: Križovatka s vyznačením dopravných prúdov.

Matica medzičasov pre križovatku z obr. 8.10

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	-	-	5	6	7	-	-	6	5	-	-	-	-	4	-	8
2	-	-	-	-	-	-	-	-	3	-	-	-	-	3	6	-
3	6	-	-	-	-	4	-	-	8	9	6	8	-	9	3	-
4	6	-	-	-	-	8	9	6	-	-	8	7	10	-	3	-
5	3	-	-	-	-	-	-	3	-	-	-	-	-	-	3	6
6	-	-	5	4	-	-	-	8	9	3	-	-	10	-	-	5
7	-	-	3	-	-	-	-	-	-	-	-	-	6	-	-	3
8	8	-	-	6	10	8	-	-	-	-	8	6	3	-	-	10
9	10	10	8	-	-	4	-	-	-	-	7	8	3	-	10	-
10	-	-	6	-	-	3	-	-	-	-	-	-	3	6	-	-
11	-	-	10	12	-	-	-	9	8	-	-	-	-	12	-	5
12	-	-	8	10	-	-	-	8	10	-	-	-	-	5	-	12
13	-	-	-	3	-	-	8	9	9	9	-	-	-	-	-	-
14	10	10	5	-	-	4	-	-	-	6	3	10	-	-	-	-
15	-	8	9	9	9	-	-	-	3	-	-	-	-	-	-	-
16	4	-	-	-	8	9	9	4	-	-	9	4	-	-	-	-

Pre dva kolízne prúdy  $i, j$  definujeme medzičas  $m_{ij}$  ako najmenší možný čas taký, o ktorý treba oneskoriť začiatok zelenej pre prúd  $j$  od konca zelenej pre prúd  $i$  tak, aby vozidlá prúdu  $i$  stihli opustiť kolízny bod prúdov  $i, j$  skôr, než doň dorazia vozidlá prúdu  $j$ . Medzičasy  $m_{ij}$  sa usporiadajú do matice medzičasov. Ak sú prúdy  $i, j$  nekolízne, má matica medzičasov na mieste  $(i, j)$  symbol „–“.



Obr. 8.11: Graf kolíznosti pre križovatku z obr. 8.10.

Pri riadení križovatky technikou fázových skupín sa jednotlivé prúdy združujú do množín tak, že každá z nich obsahuje iba navzájom nekolízne prúdy. Takáto množina nekolíznych prúdov sa volá **fáza**. Prúdy jednej fázy budú mať v budúcom signálnom pláne súčasne zelenú, po istom čase sa zelená prideli inej fáze atď. až kým sa zelená neprideli aj poslednej fáze. Potom sa celý tento dej cyklicky opakuje. Prvou úlohou pri navrhovaní signálneho plánu riadenej križovatky je určiť fázovanie.

Aké sú požiadavky na dobré fázovanie? V prvom rade každý prúd sa musí vyskytovať v niektorej fáze. Druhou požiadavkou je, aby bol počet fáz čo najmenší. Táto druhá požiadavka vyplýva zo skutočnosti, že pri prechode signálneho plánu zo zelenej pre jednu fázu na zelenú na druhú fázu musia byť dodržané medzi-



časy medzi prúdmi prvej a druhej fázy. Počas medzičasu nesvieti zelená ani pre jeden z dvojice príslušných kolíznych prúdov, križovatka sa po tento čas len vyprázdňuje. Čím je súčet medzičasov na križovatke väčší, tým je väčšia neproduktívna časť cyklu riadenia a tým sa znižuje reálna priepustnosť križovatky. Čím menej fáz bude mať signálny plán, tým menšia bude časť cyklu premárnená v medzičasoch.

Pomocou matice medzičasov môžeme zostrojiť graf  $G = (V, H)$  kolíznosti prúdov križovatky, ktorý bude mať za množinu vrcholov všetky prúdy a za množinu hrán všetky neusporiadané dvojice kolíznych prúdov (t. j. tie neusporiadané dvojice  $\{i, j\}$  vrcholov  $i, j$ , pre ktoré je v matici medzičasov na mieste  $(i, j)$  reálne číslo). Potom problém fázovania možno v grafe  $G$  riešiť ako problém zafarbenia grafu  $G$  minimálnym počtom farieb. Každému vrcholu treba určiť farbu – fázu, tak aby žiadne dva kolízne vrcholy nemali tú istú farbu – fázu a tak, aby počet použitých farieb – fáz bol čo najmenší.

### 8.5.6 Minimalizácia počtu autobusových stanovišť

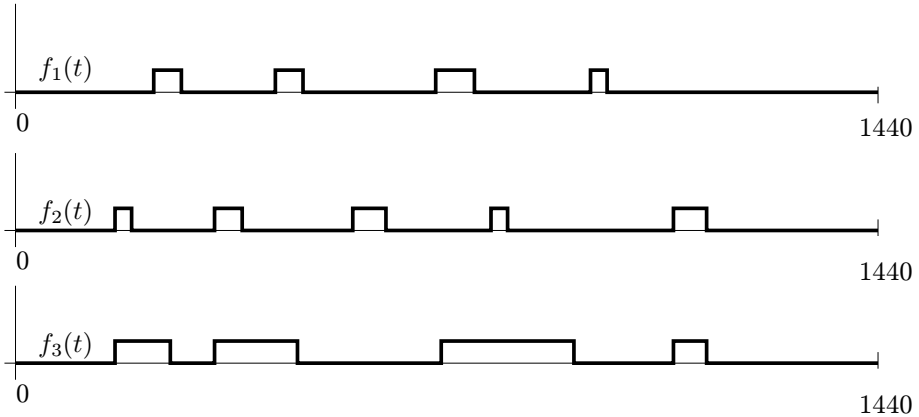
Autobusová stanica má obmedzený počet autobusových stanovišť. Na autobusovom stanovišti môže stáť len jedno vozidlo. Ideálne by bolo, keby autobusy každej linky mali svoje vlastné stanovište. To však nie je možné, pretože autobusovú stanicu nemožno rozšíriť. Preto musia niektoré linky zdieľať spoločné autobusové stanovište.

Obsadenie nástupišťa autobusmi jednej linky možno vyjadriť reálnou funkciou  $f(t)$  s definičným oborom  $\langle 0, 1440 \rangle$  (minúty jedného dňa) ktorá pre  $t \in \langle 0, 1440 \rangle$  nadobúda hodnotu 1 práve vtedy, keď je nástupište obsadené autobusom príslušnej linky, inak  $f(t) = 0$ .

Na obrázku 8.12 vidíme tri funkcie  $f_1, f_2, f_3$  obsadenia stanovišťa troch liniek  $L_1, L_2, L_3$ . Linky  $L_1, L_2$  môžu zdieľať to isté stanovište, pretože neexistuje časový okamžik, v ktorom by obe funkcie boli rovné 1 – budeme hovoriť, že linky  $L_1, L_2$  sú kompatibilné. Naopak, linka  $L_3$  nemôže zdieľať to isté stanovište so žiadnou z liniek  $L_1, L_2$ .

Pomocou funkcií  $f_1, f_2$  možno veľmi ľahko určiť, kompatibilitu liniek  $L_1, L_2$  – tie sú, kompatibilné práve vtedy, keď  $f_1(t) + f_2(t) \leq 1$  pre každé  $t \in \langle 0, 1440 \rangle$ .

Teraz môžeme zostrojiť graf  $G$ , vrcholmi ktorého budú linky a hranami ktorého budú dvojice nekompatibilných liniek. Optimálnym zafarbením grafu  $G$  určíme minimálny počet farieb (stanovišť) a každej linke priradíme farbu



Obr. 8.12: Funkcie  $f_1$ ,  $f_2$ ,  $f_3$  obsadenia nástupištia pre tri linky  $L_1$ ,  $L_2$ ,  $L_3$ .

(stanovište) tak, že žiadne dve nekompatibilné linky nemajú rovnakú farbu (stanovište).

Ak by vyšla potreba stanovišť väčšia ako disponibilný počet stanovišť, môžeme sa pokúsiť skrátiť pobyty autobusov na zastávkach, v dôsledku čoho sa „zúžia zuby“ funkcií  $f_i$  a poklesne počet nekompatibilných dvojíc liniek, čo môže mať za následok zníženie chromatického čísla príslušného grafu  $G$ .

Pri priradovaní stanovišť linkám môže prax požadovať splnenie ďalších podmienok. Napríklad je vhodné, aby diaľkové linky mali iné stanovištia ako miestne linky. Prirodzená je požiadavka, aby linky s podobnou trasou zdieľali pokiaľ možno to isté stanovište. Mnohé takéto podmienky možno modelovať pridaním ďalších hrán ku grafu  $G$ . Podmienky, ktoré sa nedajú ošetriť týmto spôsobom, môžeme skúsiť splniť nasledovne: Exaktný algoritmus možno upraviť tak, aby preskúmal všetky možné optimálne (z hľadiska počtu farieb) systematické zafarbenia aj z hľadiska splnenia dodatočných podmienok a vybral to, ktoré im najlepšie vyhovuje. Pri takomto prístupe však takmer iste nastane to, že systematických zafarbení grafu  $G$  s minimálnym počtom farieb bude veľmi veľa, a preto výpočet neskončí v rozumnom čase.

## 8.6 Cvičenia

1. Skúste nakresliť diagram grafu  $K_{3,3}$  na anuloide – „pneumatike“ tak, aby sa jeho hrany nepretínali nikde okrem vrcholov. To isté skúste na Möbiovom liste a na guľovej ploche.
2. Rovinný graf  $G$  má všetky steny okrem vonkajšej tvorené štvoruholníkmi. Na základe počtu vrcholov odhadnite počet jeho hrán.
3. Odhadnite počet hrán rovinného grafu, ktorého všetky steny okrem vonkajšej sú šesťuholníky.
4. Nájdite príklady grafov, pre ktoré algoritmus 8.1 sekvenčného farbenia grafu alebo algoritmus 8.2 paralelného farbenia grafu zafarbí graf väčším počtom farieb, než je jeho chromatické číslo.
5. Nájdite ďalšie aplikácie problému farbenia grafov.

### Počítačové cvičenia

6. Naprogramujte algoritmus 8.1 sekvenčného farbenia, algoritmus 8.2 paralelného farbenia a LDF algoritmus 8.3 farbenia grafu a porovnajte ich výsledky.
7. Naprogramujte exaktný algoritmus farbenia grafov 8.4. Porovnajte jeho výsledky s výsledkami heuristik. Skúste zistiť vplyv usporiadania vrcholov podľa ich stupňov na rýchlosť výpočtov. Do akej veľkosti grafu sa ešte dočkáte skončení programu?
8. Nájdite na internete stránky zaoberajúce sa farbením grafov. Skúste, aké výsledky dajú vaše programy na niektoré testovacie úlohy.



# Kapitola 9

## Niektoré ďalšie ťažké úlohy

V tejto kapitole sa budeme venovať niektorým ďalším NP-ťažkým úlohám. Pre tieto úlohy nemáme (a pravdepodobne ani neexistuje) polynomiálny algoritmus riešenia, avšak modely mnohých praktických úloh vedú na riešenie takýchto úloh. Našťastie prax si nevyžaduje striktné optimálne riešenie. Uspokojí sa aj s riešením dostatočne dobrým a za také pokladá každé zlepšenie súčasného stavu. Preto pre problémy tu uvedené budeme uvádzať heuristické algoritmy podobne, ako to bolo pri úlohe obchodného cestujúceho.

### 9.1 Centrá a mediány

Pri zásobovaní územia nejakým tovarom (uhlím, nábytkom, potravinami, liekmi atď.) často potrebujeme rozhodnúť, koľko skladov a v ktorých lokalitách máme postaviť. Podobný problém môžeme riešiť pri rozhodovaní koľko stredísk zdravotnej záchrannej služby a v ktorých miestach zriadiť. Tieto problémy spadajú pod tzv. lokačné problémy. Lokačné problémy sa líšia typom kritériálnej funkcie a modelom prostredia, v ktorom ich riešime. Existuje niekoľko spôsobov na modelovanie a riešenie lokačných problémov. Najdôležitejšími z nich sú metódy celočíselného lineárneho programovania a metódy teórie grafov. My sa, pochopiteľne, budeme zaoberať metódami teórie grafov v najjednoduchšom prípade, kedy je už množstvo skladov či záchraných stredísk známe. Modelom prostredia, v ktorom budeme tieto úlohy riešiť, bude súvislý hranovo a vrcholovo

ohodnotený graf  $G = (V, H, c, w)$ , v ktorom vrcholy predstavujú križovatky a dôležité body, hrany modelujú ulice, resp. priame cesty medzi vrcholmi, ohodnotenie  $c(h)$  hrany  $h \in H$  predstavuje dĺžku hrany  $h$ , ohodnotenia  $w(v)$  vrchola  $v \in V$  – váha vrchola  $v$  (predstavuje relatívnu dôležitosť vrchola  $v$ ).

Všetky vrcholy v grafe  $G = (V, H, c, w)$  potrebujú obsluhu. Ich náročnosť na obsluhu je vyjadrená ich váhou. Niektoré vrcholy v grafe  $G$  môžu navyše slúžiť ako strediská obsluhy. Poznáme dve základné funkcie stredísk obsluhy.

Prvá z nich je funkcia **zásobovania**. V tomto prípade pre stredisko obsluhy používame termín **depo**. V depe je umiestnený sklad materiálu. Každý vrchol  $v$  grafu  $G = (V, H, c, w)$  potrebuje za jednotku času  $w(v)$  jednotiek materiálu, jednotkové náklady na dovoz materiálu sú úmerné prepravovanej vzdialenosti. Tu hľadáme také umiestnenie dep, ktoré minimalizuje celkové dopravné náklady na obsluhu všetkých vrcholov grafy  $G$ .

Druhá funkcia stredísk obsluhy je **záchranná**. Takú funkciu plnia napríklad stanice pohotovostnej lekárskej služby, požiarne zbrojnice, strediská horskej služby atď. V tomto prípade pre stredisko obsluhy používame termín **havarijné stredisko**. Tu už dopravné náklady nehrajú takú dôležitú úlohu ako v predchádzajúcom prípade — kritériom je tu dostupnosť najhoršie položeného vrchola grafu  $G = (V, H, c, w)$ . Chceme nájsť umiestnenia staníc záchrannej lekárskej služby tak, aby ani ten najhoršie položený pacient neumrel pre neskorý príchod pomoci, chceme nájsť umiestnenie požiarnych zbrojníc tak, aby v prípade potreby požiarnici došli včas, aj keby požiar vznikol aj v najhoršie položenom mieste.

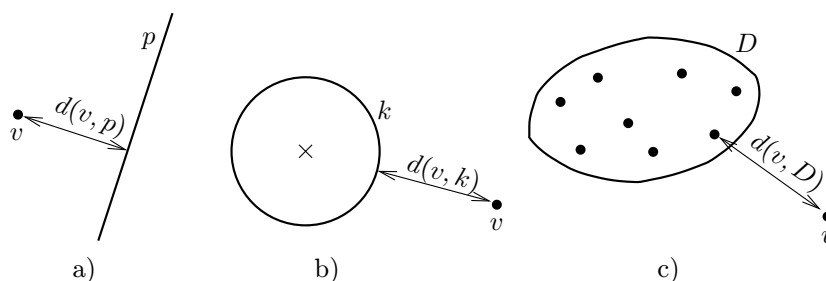
**Definícia 9.1.** Nech  $G = (V, H, c)$  je hranovo ohodnotený graf,  $D \subseteq V$  podmnožina vrcholovej množiny  $V$ ,  $v \in V$ . Potom **vzdialenosť**  $d(v, D)$  **vrchola**  $v$  **a množiny**  $D$  (resp. vzdialenosť  $d(D, v)$  množiny  $D$  a vrchola  $v$ ) definujeme nasledovne:

$$d(v, D) = d(D, v) = \min\{d(v, x) \mid x \in D\}, \quad (9.1)$$

kde  $d(v, x)$  je vzdialenosť vrcholov  $v, x$  v grafe  $G$ .

*Poznámka.* Všimnime si, že ak  $v \in D$ , potom  $d(v, D) = 0$ .

Z obrázku 9.1 je zrejma podobnosť pojmu „vzdialenosť bodu a množiny“ v geometrii a pojmu „vzdialenosť vrchola  $v$  a množiny  $D$ “. Vzdialenosť bodu  $v$  a priamky  $p$  je minimum zo vzdialeností  $d(v, x)$ , kde  $x$  prebieha všetky body ležiace na priamke  $p$ . Vzdialenosť bodu  $v$  a kružnice  $k$  je vzdialenosť  $d(v, x)$  bodu  $v$  a toho bodu  $x$  kružnice  $k$ , ktorý je najbližšie k bodu  $x$ . Analogicky je vzťahom (9.1) definovaná vzdialenosť vrchola  $v$  a množiny  $D \subseteq V$ .



Obr. 9.1: Analógia medzi vzdialenosťou a) bodu  $v$  a priamky  $p$ ,  
 b) bodu  $v$  a kružnice  $k$   
 a vrchola  $v$  a množiny vrcholov  $D$ .

**Definícia 9.2.** Nech  $G = (V, H, c, w)$  je súvislý hranovo a vrcholovo ohodnotený graf,  $D \subseteq V$ . **Súhrnná vážená vzdialenosť  $f(D)$  všetkých vrcholov grafu  $G$  od množiny  $D$**  je definovaná nasledovne:

$$f(D) = \sum_{v \in V} w(v) \cdot d(v, D) . \quad (9.2)$$

Váha  $w(v)$  vrchola  $v \in V$  môže predstavovať množstvo tovaru, ktoré do vrchola  $v$  treba dodať za jednotku času. Predpokladajme, že dopravné náklady na obsluhu vrchola  $v$  sú priamo úmerné jeho vzdialenosti od najbližšieho depa  $z D$  (z ktorého bude vrchol  $v$  zásobovaný) a množstvu  $w(v)$  - teda dopravné náklady na obsluhu vrchola  $v$  za jednotku času sa budú rovnať  $k \cdot w(v) \cdot d(v, D)$ , kde  $k$  je vhodná konštanta. Celkové dopravné náklady na obsluhu všetkých vrcholov za jednotku času budú

$$N = k \cdot \sum_{v \in V} w(v) \cdot d(v, D) = k \cdot f(D),$$

$N$  je teda priamo úmerné súhrnnej vázenej vzdialenosti všetkých vrcholov grafu  $G$  od množiny diep  $D$ . Ak chceme nájsť optimálnu  $p$ -prvkovú množinu diep, ktorá minimalizuje celkové dopravné náklady na obsluhu všetkých vrcholov grafu  $G$ , stačí nájsť takú  $p$ -prvkovú množinu diep  $D_p$ , pre ktorú je  $f(D_p)$  minimálne.

**Definícia 9.3.** Nech  $1 \leq p < |V|$ ,  $D_p$   $p$ -prvková podmnožina množiny  $V$ . Hovoríme, že  $D_p$  je **vážený  $p$ -medián** grafu  $G$ , ak pre ľubovoľnú  $p$ -prvkovú

podmnožinu  $D'_p$  množiny  $V$  platí

$$f(D_p) \leq f(D'_p),$$

t.j. ak súhrnná vážená vzdialenosť všetkých vrcholov grafu  $G$  od  $D_p$  je najmenšia medzi všetkými  $p$ -prvkovými podmnožinami množiny  $V$ . Špeciálne ak  $w(v) = 1$  pre všetky  $v \in V$ , hovoríme, že  $D_p$  je  **$p$ -medián**.

**Definícia 9.4.** Nech  $G = (V, H, c, w)$  je súvislý hranovo a vrcholovo ohodnotený graf,  $D \subseteq V$ . **Vážená excentricita  $\text{ecc}(D)$  množiny  $D$**  je definovaná nasledovne:

$$\text{ecc}(D) = \max\{w(v).d(v, D) \mid v \in V\}.$$

Vážená excentricita množiny  $D$  je vážená vzdialenosť najhoršie položeného vrchola od množiny  $D$ . Vyjadruje kvalitu množiny havarijných stredísk  $D$  z hľadiska kvality obsluhy najhoršie položeného vrchola vzhľadom na  $D$ .

**Definícia 9.5.** Nech  $1 \leq p < |V|$ ,  $D_p$   $p$ -prvková podmnožina množiny  $V$ . Hovoríme, že  $D_p$  je **vážené  $p$ -centrum** grafu  $G$ , ak pre ľubovoľnú  $p$ -prvkovú podmnožinu  $D'_p$  množiny  $V$  platí

$$\text{ecc}(D_p) \leq \text{ecc}(D'_p),$$

t. j. ak množina  $D_p$  má najmenšiu váženú excentricitu zo všetkých  $p$ -prvkových podmnožín množiny  $V$ . Špeciálne ak  $w(v) = 1$  pre všetky  $v \in V$ , hovoríme, že  $D_p$  je  **$p$ -centrum**.

**Algoritmus 9.1. Heuristický algoritmus na hľadanie váženého  $p$ -mediánu v súvislom hranovo a vrcholovo ohodnotenom grafe  $G = (V, H, c, w)$ .**

- **Krok 1.** Náhodne vyber  $p$ -prvkovú podmnožinu množiny  $V$ .  
Nech  $D_p = \{v_1, v_2, \dots, v_p\}$ ,  $V - D_p = \{u_1, u_2, \dots, u_q\}$ , kde  $q = |V| - p$ .
- **Krok 2.** Hľadaj také  $i, j$ ,  $1 \leq i \leq p$ ,  $1 \leq j \leq q$ ,  
že pre  $D'_p(i, j) = (D_p \cup \{u_j\}) - \{v_i\}$  je  $f(D'_p) < f(D_p)$ .
- **Krok 3.** Ak taká dvojica indexov  $i, j$  neexistuje, STOP.  
Inak polož  $D_p := D'_p(i, j)$  a GOTO Krok 2.





Posledný algoritmus nie je závislý na tvare funkcie  $f(D_p)$ . Ak v ňom namiesto funkcie  $f(D_p)$  použijeme funkciu  $\text{ecc}(D_p)$ , dostaneme suboptimálny algoritmus pre hľadanie váženého  $p$ -centra grafu  $G$ .

Predchádzajúci algoritmus realizuje prvú možnú výmenu depa  $v_i$  za nové depo  $u_j$  vedúcu k lepšej množine diep. Tento prístup by sme mohli modifikovať tak, že pre jedno pevné depo  $v_i \in V$  sa nájde vrchol  $u_j \in V - D_p$ , pre ktorý je  $f(D'_p(i, j))$  minimálne a zrealizuje sa až takáto najlepšia výmena. Inou možnosťou je nájsť najlepšiu dvojicu  $v_i \in D_p, u_j \in V - D_p$  z hľadiska kriteriálnej funkcie  $F(D'_p(i, j))$  a zrealizovať až túto výmenu. Žiadna zo spomínaných stratégií však nezaručuje ani optimálny, ani lepší výsledok ako tie ostatné. Heuristika skončí, ak nenájde dvojicu typu depo  $v_i$  - vrchol  $u_j$  takú, že výmenou depa  $v_i$  za vrchol  $u_j$  možno dosiahnuť novú množinu diep s menšou hodnotou kriteriálnej funkcie. V takomto prípade hovoríme, že sme našli **lokálne minimum**.

Ak by sme chceli zmenšiť riziko, že heuristika skončí v niektorom zlom „lokálnom“ minime, môžeme použiť popisovaný algoritmus niekoľkokrát, vždy s inou štartovacou množinou  $D_p$  a vybrať z výsledkov najlepšie riešenie.

Pre malé  $p$  je tu ešte možnosť prezrieť všetky riešenia, ktorých je  $\binom{n}{p}$ . Pre štyri depa a sto vrcholov je  $\binom{n}{p} = \binom{100}{4} \approx 4.10^6$  možností umiestnenia diep, čo je reálny počet na preskúmanie súčasnou výpočtovou technikou. V praktických situáciách často už pred výpočtom vieme, že mnohé lokality sú pre umiestnenie depa nevhodné z priestorových, enviromentálnych či iných dôvodov, čo ešte viac znižuje počet prehľadávaných možností.

Na záver diskusie o heuristickom algoritme na hľadanie  $p$ -mediánu resp.  $p$ -centra poznamenajme, že existujú aj zložitejšie a efektívnejšie heuristické techniky, ako tu uvádzaná zámenná heuristika.<sup>1</sup> Tu popisované i všeobecnejšie lokačné úlohy možno modelovať a úspešne riešiť tiež prostriedkami celočíselného lineárneho programovania.

Ak už máme určenú množinu diep resp. havarijných stredísk, je ešte potrebné určiť, z ktorého depa bude obsluhovaný ktorý vrchol dopravnej siete.

**Definícia 9.6.** Nech je daný súvislý hranovo a vrcholovo ohodnotený graf  $G = (V, H, c, w)$  a  $p$ -prvková množina diep  $D_p$ .

**Atrakčný obvod**  $A(v)$  depa  $v \in D_p$  je množina všetkých takých vrcholov grafu  $G$ , ktorých vzdialenosť od depa  $v$  je menšia alebo rovná ako vzdialenosť od iných diep, t.j.

$$A(v) = \{x \mid x \in V, \forall u \in D_p d(v, x) \leq d(u, x)\}$$

<sup>1</sup>Sú to metaheuristiky typu tabu search, genetické algoritmy, simulated annealing atď.

**Prvotný atrakčný obvod**  $A'(v)$  depa  $v \in D_p$  je množina všetkých takých vrcholov grafu  $G$ , ktorých vzdialenosť od depa  $v$  je menšia ako vzdialenosť od iných diep, t.j.

$$A'(v) = \{x \mid x \in V, \forall u \in D_p, u \neq v \ d(v, x) < d(u, x)\}$$

**Systém pridelených atrakčných obvodov** je systém podmnožín  $A^v(v), v \in D_p$  vrcholovej množiny  $V$  takých že

1.  $A'(v) \subseteq A^v(v) \quad \forall v \in D_p$
2.  $A^v(v) \subseteq A(v) \quad \forall v \in D_p$
3.  $A^v(u) \cap A^v(v) = \emptyset \quad \forall u, v \in D_p, u \neq v$
4.  $\bigcup_{v \in D_p} A^v(v) = V$

Atrakčný obvod  $A(v)$  depa  $v$  silne závisí od množiny diep  $D_p$ . Aby sme to zdôraznili, mali by sme dôsledne písať  $A(v, D_p)$  namiesto zjednodušeného  $A(v)$ . To isté platí aj pre ďalšie druhy atrakčných obvodov.

Dôvod pre zavedenie viacerých druhov atrakčných obvodov je nasledujúci. Nech  $x$  je vrchol, ktorý má rovnakú najmenšiu vzdialenosť od dvoch (alebo viacerých) diep. Takýto vrchol  $x$  patrí do atrakčných obvodov všetkých diep, od ktorých má vrchol  $x$  rovnakú najmenšiu vzdialenosť, zatiaľ čo  $x$  nepatrí v takomto prípade ani do jedného prvotného atrakčného obvodu. Z praktických dôvodov chceme, aby každý vrchol bol obsluhovaný práve z jedného depa. Problém rieši systém pridelených atrakčných obvodov, v ktorých je každý vrchol prvkom práve jedného atrakčného obvodu.

## 9.2 Kliky a maximálne nezávislé množiny

Pri akomkoľvek (nie nutne optimálnom) zafarbení grafu  $G$  sme dostali rozklad množiny vrcholov na triedy rovnako zafarbených vrcholov. Žiadne dva vrcholy jednej triedy nie sú susedné. Pre množiny s touto vlastnosťou máme nasledujúcu definíciu.

**Definícia 9.7.** Nech  $G = (V, H)$  je graf. Nech  $N$  je podmnožina množiny  $V$ , teda  $N \subseteq V$ . Hovoríme, že  $N$  je **nezávislá množina v grafe**  $G$ , ak žiadne dva

prvky z množiny  $N$  nie sú susedné.

Hovoríme, že  $N$  je **maximálna nezávislá množina v grafe**  $G$ , ak  $N$  je nezávislá a pritom  $N$  nie je podmnožinou žiadnej inej nezávislej množiny v  $G$ .

Hovoríme, že  $N$  je **najpočetnejšia nezávislá množina v grafe**  $G$ , ak  $N$  má najväčší počet prvkov medzi všetkými nezávislými množinami v  $G$ .

Nech  $K \subseteq V$  je podmnožina množiny  $V$ . Hovoríme, že  $K$  je **úplná množina v grafe**  $G$ , ak každé dva rôzne prvky množiny  $K$  sú susedné.  $K$  je **maximálna úplná množina** alebo **klika v grafe**  $G$ , ak  $K$  je úplná v  $G$  a  $K$  nie je podmnožinou žiadnej inej úplnej množiny v  $G$ .

Hovoríme, že  $K$  je **najpočetnejšia klika v grafe**  $G$ , ak  $K$  má najväčší počet prvkov medzi všetkými klikami v  $G$ .

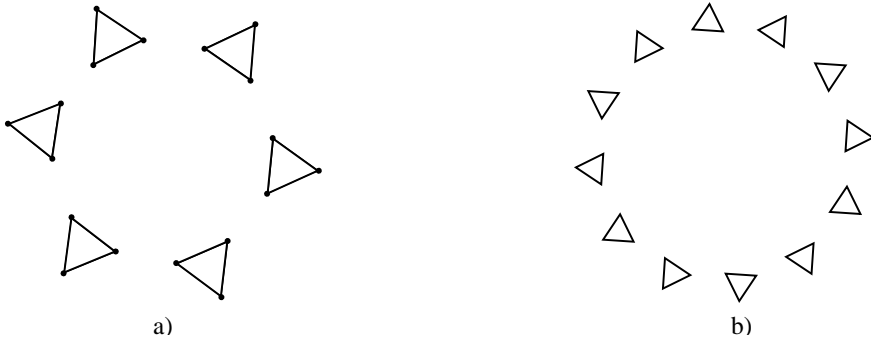
**Klikové číslo**  $\omega(G)$  **grafu**  $G$  je počet prvkov najpočetnejšej kliky v  $G$ . **Číslo maximálnej nezávislosti**  $\text{ind}(G)$  **grafu**  $G$  je počet prvkov najpočetnejšej nezávislej množiny v  $G$ .

Všimnime si, že množina  $M \subseteq V$  je nezávislá v grafe  $G$  práve vtedy, keď je úplná v komplementárnom grafe  $\overline{G}$  grafu  $G$ . Ak je  $M$  najpočetnejšia nezávislá množina v grafe  $G$ , potom  $M$  je najpočetnejšia klika v komplementárnom grafe  $\overline{G}$ . Preto platí  $\omega(G) = \text{ind}(\overline{G})$ .

Kliky a maximálne nezávislé množiny sa často vyžadujú v mnohých aplikáciách, preto má význam hľadať všetky kliky resp. všetky maximálne nezávislé množiny v grafe  $G$ . Vzhľadom na to, že nezávislosť a úplnosť sú komplementárne pojmy, stačí mať algoritmus na hľadanie všetkých klík. Hľadanie všetkých maximálnych nezávislých množín sa prevedie na hľadanie všetkých klík v komplementárnom grafe  $\overline{G}$ .

Klík resp. nezávislých množín v grafe  $G$  môže byť veľmi veľa. Majme graf  $G_{3,k}$  s  $3k$  vrcholmi pozostávajúci z  $k$  komponentov, z ktorých každý je úplný graf  $K_3$  s tromi vrcholmi. Každá maximálna nezávislá množina  $\mathcal{M}$  v  $G_{3,k}$  bude obsahovať práve jeden vrchol z každého z  $k$  komponentov typu  $K_3$ . Pretože výber vrchola do  $\mathcal{M}$  z jedného komponentu nijako neovplyvní výber vrchola z iných komponentov, máme  $3^k$  možností tvorby maximálnej nezávislej množiny. Práve toľko je klík v komplementárnom grafe  $\overline{G}_{3,k}$  grafu  $G_{3,k}$ .<sup>2</sup> Preto žiaden algoritmus na hľadanie všetkých klík grafu nemôže byť polynomiálny.

<sup>2</sup>V literatúre sa uvádza, že grafy typu  $G_{3,k}$  majú najväčší možný počet maximálnych nezávislých množín na jeden vrchol:  $\frac{3^k}{3k} = \frac{3^{k-1}}{k}$ .



Obr. 9.2: a) Graf  $G_{3,6}$ . Má 6 komponentov typu  $K_3$ .

Je  $\omega(G_{3,6}) = 3$ ,  $\text{ind}(G_{3,6}) = 6$ .

Existuje v ňom  $3^6 = 729$  maximálnych nezávislých množín.

b) Graf  $G_{3,12}$ ,  $\omega(G_{3,12}) = 3$ ,  $\text{ind}(G_{3,12}) = 12$ .

Počet maximálnych nezávislých množín v  $G_{3,12}$  je  $3^{12} = 531441$ .

Iným problémom, na ktorý vedie mnoho praktických úloh, je hľadanie najpočetnejšej kliky resp. najpočetnejšej nezávislej množiny v grafe  $G$ . Tento problém sa ukázal byť NP-ťažký.

Metóda hľadania všetkých klík spočíva vo vyskúšaní všetkých možností, ktoré prichádzajú do úvahy. V každej fáze výpočtu pole  $K$  bude obsahovať  $k$ -prvkovú úplnú podmnožinu vrcholov  $\mathcal{K}_i = \{K[1], K[2], \dots, K[i]\}$ . Mohutnosť množiny  $\mathcal{K}_i$  – t.j. číslo  $i$  sa bude v priebehu výpočtu meniť s tým, ako budeme do množiny  $\mathcal{K}_i$  pridávať alebo z nej uberať vrcholy. V priebehu výpočtu budeme pre všetky vrcholy  $i = 1, 2, \dots, k$  udržiavať dva systémy množín vrcholov  $\mathcal{N}(i)$  a  $\mathcal{C}(i)$  také, že bude platiť:

1. Všetky vrcholy  $v$  z  $\mathcal{C}(i) \cup \mathcal{N}(i)$  sú pridane k množine  $\mathcal{K}_i = \{K[1], K[2], \dots, K[i]\}$ , t.j. pre každé  $v \in \mathcal{C}(i) \cup \mathcal{N}(i)$  je  $\mathcal{K}_i \cup \{v\}$  úplnou množinou.
2.  $\mathcal{C}(i)$  – množina všetkých vrcholov, ktoré doteraz ešte neboli pridané k množine  $\mathcal{K}_i$ , t.j. žiadna úplná množina typu  $\mathcal{K}_i \cup \{v\}$  ešte nebola preskúmaná pre žiadne  $v \in \mathcal{C}(i)$ .
3.  $\mathcal{N}(i)$  množina všetkých vrcholov  $v$ , ktorých pridanie ku  $\mathcal{K}_i$  už bolo vyskúšané, t.j. všetky kliky obsahujúce  $\mathcal{K}_i \cup \{v\}$ , kde  $v \in \mathcal{N}(i)$ , už boli nájdené.

4.  $(\mathcal{C}(i) \cup \mathcal{N}(i)) \cap \mathcal{K}_i = \emptyset$ , t.j. ani  $\mathcal{C}(i)$  ani  $\mathcal{N}(i)$  neobsahujú žiaden vrchol zaradený v úplnej množine  $\mathcal{K}_i$ .

Algoritmus sa bude snažiť rozšíriť množinu  $\mathcal{K}_k$  tak, že vyberie vrchol  $x \in \mathcal{C}(k)$  a položí  $K[k+1] := x$ . Zároveň definuje nové množiny  $\mathcal{C}(k+1) := \mathcal{C}(k) \cap V_x$ ,  $\mathcal{N}(k+1) := \mathcal{N}(k) \cap V_x$ , kde  $V_x$  je množina všetkých vrcholov susedných s vrcholom  $x$  a zvýši  $k$ , t. j. položí  $k := k + 1$ . Takto postupuje dotedy, kým  $\mathcal{C}(k) \neq \emptyset$ .

Ak  $\mathcal{C}(k) = \emptyset$  a súčasne  $\mathcal{N}(k) = \emptyset$ , objavili sme novú kliku. Ak  $\mathcal{C}(k) = \emptyset$ , ale  $\mathcal{N}(k) \neq \emptyset$ , úplná množina  $\mathcal{K}_k$  ešte nie je klikou, lebo k nej možno priradiť niektorý z vrcholov z  $\mathcal{N}(k)$ , ale tieto prípady už boli preskúmané a ďalšie rozširovanie by mohlo viesť len k známej klikke.

Ak teda nemožno množinu  $\mathcal{K}_k$  ďalej rozšíriť (po nájdení kliky alebo po zistení, že už boli jej možné rozšírenia preskúmané), treba sa vrátiť pred jej posledné rozšírenie o vrchol  $x = K[k]$  – t.j. zmenšiť ju a skúsiť ju rozšíriť o iný vrchol. To urobíme tak, že položíme  $k := k - 1$  a preradíme vrchol  $x$  z množiny  $\mathcal{C}(k)$  kandidátov na rozšírenie  $\mathcal{K}_k$  do množiny preverených rozšírení  $\mathcal{N}(k)$ .

Je možné vylepšenie popísaného postupu, ktorým môžeme včas zistiť, že nemá význam ďalej množinu  $\mathcal{K}_k$  rozširovať a že sa treba o krok vrátiť. Nech existuje  $p \in \mathcal{N}(k)$  také, že  $p$  je susedný vrchol zo všetkými vrcholmi v  $\mathcal{C}(k)$ . Pre taký vrchol platí  $p \in \bigcap_{x \in \mathcal{C}(k)} V_x$ , a teda nikdy nedostaneme množinu  $\mathcal{N}()$  prázdnu. Po nájdení takého vrchola nemá význam úplnú množinu  $\mathcal{K}_k$  ďalej rozširovať a treba sa vrátiť o krok späť.

**Algoritmus 9.2. Exaktný algoritmus na hľadanie všetkých klík grafu  $G = (V, H)$ .**

Predpokladajme, že  $V = \{1, 2, \dots, n\}$ .

- **Krok 0. Inicializácia.**  
Polož  $k := 0$ ,  $\mathcal{N}(0) := \emptyset$ ,  $\mathcal{C}(0) := V$ .
- **Krok 1. Rozšírenie úplnej množiny.**  
Polož  $x := \min \mathcal{C}(k)$ . Polož

$$\begin{aligned} K[k+1] &:= x \\ \mathcal{N}(k+1) &:= \mathcal{N}(k) \cap V_x \\ \mathcal{C}(k+1) &:= \mathcal{C}(k) \cap V_x \\ k &:= k + 1 \end{aligned}$$

- **Krok 2. Test maximálnosti úplnej množiny.**  
Ak  $\mathcal{C}(k) = \emptyset$  a súčasne  $\mathcal{N}(k) = \emptyset$ , ulož alebo vytlač kliku  $\mathcal{K}_k = \{K[1], K[2], \dots, K[k]\}$  a GOTO Krok 5.
- **Krok 3. Test možnosti zväčšovania úplnej množiny.**  
Ak  $\mathcal{C}(k) = \emptyset$ , GOTO Krok 5.
- **Krok 4. Test na predčasné ukončenie rozširovania množiny  $\mathcal{K}_k$ .**  
Ak existuje vrchol  $p \in \mathcal{N}(k)$ , ktorý je susedný so všetkými vrcholmi množiny  $\mathcal{C}(k)$ , GOTO Krok 5.  
Inak GOTO Krok 1.
- **Krok 5. Návrat pred prídanie posledného vrchola.**  
Ak  $k = 0$ , STOP. Boli nájdené všetky kliky grafu.  
Inak polož  $x := K[k]$ ,  $k := k - 1$ ,  $\mathcal{C}(k) := \mathcal{C}(k) - \{x\}$ ,  $\mathcal{N}(k) := \mathcal{N}(k) \cup \{x\}$  a GOTO Krok 3.



Na záver tejto časti poznamenajme, že vo vrcholovo ohodnotenom grafe  $G = (V, H, w)$ , kde  $w(v) > 0$  pre  $v \in V$  možno formulovať úlohu hľadať najcennejšiu nezávislú množinu alebo najcennejšiu úplnú množinu ako nezávislú resp. úplnú množinu s najväčším súčtom ohodnotení vrcholov. Vzhľadom na kladné ocenenie vrcholov najcennejšia nezávislá resp. úplná množina bude zároveň maximálnou nezávislou množinou resp. klikou.

### 9.3 Dominujúce množiny

**Definícia 9.8.** Nech  $G = (V, H)$  je graf,  $P \subseteq V$ . Hovoríme, že  $P$  je **dominujúca množina grafu**  $G$ , ak pre každý vrchol  $v \in V$  grafu  $G$  je buď  $v \in P$ , alebo existuje  $w \in P$  taký, že  $\{w, v\} \in H$ . Hovoríme, že  $P \subseteq V$  je **minimálna dominujúca množina grafu**  $G$ , ak  $P$  je dominujúca množina grafu  $G$  a žiadna jej pravá podmnožina nemá túto vlastnosť. Hovoríme, že  $P$  je **dominujúca množina grafu**  $G$  s **najmenšou mohutnosťou** alebo **najmenej početná dominujúca množina**, ak neexistuje dominujúca množina grafu  $G$  s menším počtom prvkov.

Problém nájsť najmenej početnú dominujúcu množinu grafu je NP-ťažký. Jeho riešenie vedie priamo na známy pokrývací problém (Set Covering Problem)

bivalentného lineárneho programovania, ktorý je teoreticky i prakticky dobre prepracovaný.

Nech  $\mathbf{C} = (c_{ij})$  je matica príľahlosti grafu  $G$ , t.j.  $c_{ij} = 1$  práve vtedy, keď  $\{i, j\} \in H$ , inak  $c_{ij} = 0$ . Predefinujme všetky diagonálne prvky matice  $\mathbf{C}$  na jedničky, t.j.  $c_{ii} := 1$ , pre  $i = 1, 2, \dots, n = |V|$ . Teraz nájsť dominujúcu množinu s najmenšou mohutnosťou znamená nájsť takú najmenej početnú množinu stĺpcov  $\mathcal{S}$ , že pre každý riadok sa nájde aspoň jeden stĺpec obsahujúci v tomto riadku jednotku.

Nech  $x_i$  je bivalentná premenná,  $x_i = 1$  práve vtedy, keď vyberieme stĺpec  $i$  do množiny  $\mathcal{S}$ , inak  $x_i = 0$ . Skúmame súčet

$$\sum_{j=1}^n c_{ij}x_j = c_{i1}x_1 + c_{i2}x_2 + \dots + c_{in}x_n.$$

Sčítanec  $c_{ij}x_j$  sa rovná 1 len ak vo vybranom stĺpci  $j$  je v riadku  $i$  číslo 1. Výraz (9.3) sa rovná počtu vybraných stĺpcov, ktoré majú v riadku  $i$  číslo 1. Ak chceme, aby  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  vyjadrovalo prípustné riešenie, musí byť výraz (9.3) väčší alebo rovný 1 pre každé  $i \in \{1, 2, \dots, n\}$ . Počet vybraných stĺpcov je rovný  $\sum_{i=1}^n x_i$ . Teraz už môžeme našu úlohu formulovať nasledovne:

Pre danú maticu  $\mathbf{C}$  príľahlosti grafu  $G$  nájsť takú  $n$ -ticu reálnych čísel

$$\mathbf{x} = (x_1, x_2, \dots, x_n) \tag{9.3}$$

pre ktorú je 
$$\sum_{i=1}^n x_i \text{ minimálne} \tag{9.4}$$

za predpokladov: 
$$\sum_{j=1}^n c_{ij}x_j \geq 1 \text{ pre všetky } i \in \{1, 2, \dots, n\} \tag{9.5}$$

$$x_{ij} \in \{0, 1\} \text{ pre všetky } i, j \in \{1, 2, \dots, n\} \tag{9.6}$$

Úlohy menšieho rozsahu možno exaktne riešiť v tabuľkovom procesore EXCEL s využitím nástroja *Řešitel -- Solver*, väčšie úlohy zvládne študentská verzia programu XPRESS, pre veľké úlohy treba použiť jeho komerčnú verziu.

## 9.4 Aplikácie

### 9.4.1 Znovu fázovanie svetelne riadenej križovatky

V časti 8.5.5 na str. 238 sme definovali problém optimálneho fázovania svetelne riadenej križovatky ako problém farbenia grafu. Bolo to pre prípad, že každý prúd má dostať zelenú počas cyklu práve raz.

Ak dovolíme, aby niektorý prúd dostal zelenú počas cyklu aj viackrát, potom môžeme hľadať fázy ako maximálne nezávislé množiny prúdov. Maximalita tu vyplýva z prirodzeného stanoviska: prečo by mal nejaký prúd stáť, keď môže mať zelenú. Problém optimálneho fázovania teraz možno preformulovať ako problém pokrytia grafu najmenším počtom maximálnych nezávislých množín.

Úloha sa potom rieši v dvoch krokoch. V prvom kroku sa nájdú všetky maximálne nezávislé množiny (dá sa tu použiť algoritmus na hľadanie všetkých klík v komplementárnom grafe).

V druhom kroku treba vybrať z množiny všetkých maximálnych nezávislých množín najmenší možný počet takých, ktoré pokrývajú všetky vrcholy grafu kolíznosti. Zostrojme maticu  $\mathbf{C} = (c_{ij})$  typu  $n \times q$ , kde  $n$  je počet prúdov križovatky a  $q$  je počet maximálnych nezávislých množín v grafe kolíznosti. Prvky  $c_{ij}$  matice  $\mathbf{C}$  definujeme nasledovne:  $c_{ij} = 1$  práve vtedy, keď  $j$ -tá maximálna nezávislá množina obsahuje vrchol  $i$ . Našou úlohou je vybrať minimálny počet stĺpcov matice  $\mathbf{C}$  tak, aby pre každý riadok  $i$  existoval aspoň jeden stĺpec z vybratej množiny obsahujúci na  $i$ -tom mieste 1. Táto druhá úloha je typickým pokrývacím problémom matematického programovania ako bol formulovaný v časti 9.3 formulami (9.3), (9.4), (9.5) a (9.6).

Poznamenajme ešte, že úlohu pokrytia grafu  $G$  najmenším počtom maximálnych nezávislých množín môžeme riešiť aj tak, že graf  $G$  najprv zafarbíme najmenším počtom farieb. Množiny rovnako zafarbených vrcholov sú nezávislé množiny, ktoré v druhom kroku ľahko rozšírime na maximálne nezávislé množiny jednoduchým pridávaním vrcholov dovtedy, dokiaľ pridaním ďalšieho vrchola neporušíme ich nezávislosť.

### 9.4.2 Úlohy o koalíciách

Do parlamentu sa dostalo niekoľko strán. Niektoré dvojice strán sú obojstranne ochotné spolupracovať, iné dvojice zase majú tak odlišný program, že



spolupracovať nechcú. Akú najväčšiu koalíciu (vzhľadom na počet strán) možno v parlamente vytvoriť?

Úlohu možno modelovať v grafe  $G = (V, H)$ , v ktorom množinu vrcholov tvoria strany a množinu hrán neusporiadané dvojice strán ochotné spolupracovať. Potom nájsť najpočetnejšiu (čo do počtu strán) koalíciu znamená nájsť najpočetnejšiu kliku v grafe  $G$ .

Pozmeňme teraz úlohu smerom bližšie k realite. Treba vytvoriť najväčšiu koalíciu nie z hľadiska počtu zúčastnených strán, ale z hľadiska počtu mandátov. Na riešenie tejto úlohy možno použiť graf  $G$  zostrojený pre predchádzajúcu úlohu, ku ktorému pridáme ohodnotenie vrcholov – strán počtom mandátov. Dostaneme tak vrcholovo ohodnotený graf  $G = (V, H, w)$ , v ktorom úlohu o koalícii s najväčším počtom mandátov riešime ako hľadanie najcennejšej kliky.

Podobného charakteru sú úlohy o vytvorení najpočetnejšej skupiny ľudí, ktorí sa navzájom dobre znášajú alebo úlohy o jednej nákupnej taške: Koľko najviac zlučiteľných tovarov môžeme uložiť do jednej tašky z hľadiska počtu, váhy alebo ceny.

### 9.4.3 Ešte raz o havarijných strediskách

Pôvodná úloha o rozmiestnení havarijných stredísk bola táto: Máme možnosť umiestniť v súvislom hranovo a vrcholovo ohodnotenom grafe  $G = (V, H, c, w)$   $k$  havarijných stredísk. Treba určiť  $k$ -prvkovú množinu vrcholov  $D_k$  ako množinu havarijných stredísk tak, aby vzdialenosť najhoršie umiestneného vrchola od množiny  $D_k$  bola minimálna.

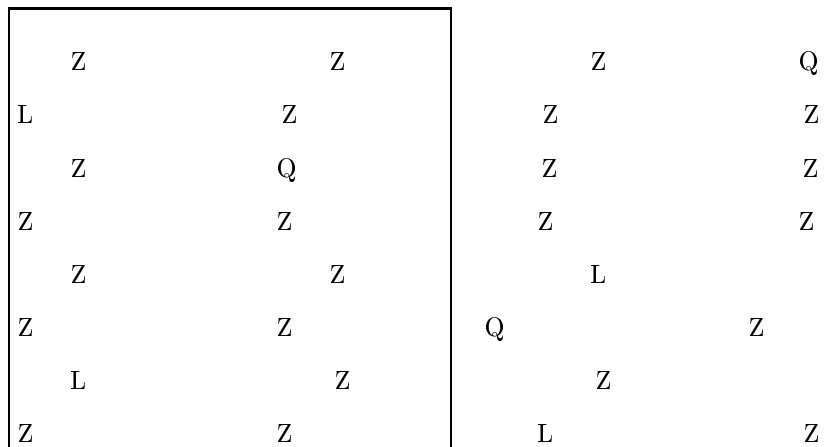
Pozmeňme trochu úlohu nasledovne: V grafe  $G$  máme určiť množinu havarijných stredísk  $D$  tak, aby každý vrchol grafu mal havarijné stredisko vo vzdialenosti do 10 km. Akým najmenším počtom havarijných stredísk sa to dá urobiť a kde majú byť tieto strediská umiestnené?

Nech  $G = (V, H, c, w)$  je súvislý hranovo a vrcholovo ohodnotený graf, v ktorej hľadáme umiestnenie minimálneho počtu havarijných stredísk. Ku grafu  $G$  zostrojme pomocný graf  $G' = (V, H')$  s rovnakou množinou vrcholov. Neusporiadaná dvojica  $\{u, v\}$  rôznych vrcholov z  $V$  bude tvoriť hranu množiny  $H'$  práve vtedy, keď vzdialenosť vrcholov  $u, v$  v pôvodnom grafe  $G$  bude menšia alebo rovná než 10 km. Problém umiestnenia minimálneho počtu havarijných stredísk s dostupnosťou do 10 km sa teraz dá formulovať ako problém hľadania dominujúcej množiny v grafe  $G'$  s najmenšou mohutnosťou.

#### 9.4.4 Dámy na šachovnici

Koľko dám možno rozostaviť na štandardnú šachovnicu o  $8 \times 8$  políčkach tak, aby sa navzájom neohrozovali? Táto úloha je dlho známa a jej riešenie bolo publikované v berlínskom časopise Schachspiele už v roku 1854.

Ako model pre túto úlohu môžeme zostrojiť graf  $G = (V, H)$  so 64 vrcholmi prislúchajúcimi políčkam šachovnice. Dva vrcholy budú susedné práve vtedy, keď sa dámy postavené na týchto políčkach ohrozujú. Nájst' postavenie maximálneho počtu neohrozujúcich sa dám na šachovnici znamená nájst' najpočetnejšiu nezávislú množinu v grafe  $G$ . Úlohu vyriešime algoritmom pre hľadanie všetkých klík v komplementárnom grafe  $\overline{G}$ . Z nájdených klík vyberieme najpočetnejšiu, ktorá bude definovať hľadané riešenie.



Obr. 9.3: Jedno možné riešenie problému dám.

#### 9.4.5 Ústredne v komunikačnej sieti

Komunikačná sieť pozostáva z  $n$  lokalít, medzi niektorými z nich sú vybudované priame telefónne (alebo dátové) vedenia. Aby bolo možné aj iné spojenie ako priame, treba v mieste niektorých lokalít vybudovať prepojovacie ústredne. Na ústredňu je napojiteľná lokalita v mieste ústredne a lokalita spojená vedením

s miestom ústredne. Aký najmenší počet ústrední treba v sieti vybudovať, aby každá lokalita bola napojiteľná aspoň na jednu ústredňu?

Situáciu možno modelovať grafom  $G = (V, H)$ , ktorého množinu vrcholov  $V$  tvoria všetky lokality. Dvojica lokalít tvorí hranu množiny  $H$  práve vtedy, keď sú tieto lokality spojené priamym vedením. Úlohu o umiestnení ústrední teraz možno formulovať ako úlohu hľadania dominujúcej množiny s minimálnou mohutnosťou v grafe  $G$ .

Podobný problém vzniká pre spoločnosť šíriacu televízny program káblovými rozvodmi. Televízny signál sa prijíma zo satelitných vysielačov špeciálnymi prijímačmi a potom sa rozvádza káblami. Úlohou je zistiť, aký minimálny počet prijímacích staníc a kde má spoločnosť vybudovať, aby všetci zákazníci obsluhovaného regiónu boli pripojiteľní na niektorú z prijímacích staníc.

## 9.5 Cvičenia

1. Ukážte, že  $\chi(G) \leq \omega(G)$ , t.j. chromatické číslo grafu  $G$  je menšie alebo rovné než klikové číslo grafu  $G$
2. Ukážte, že  $\chi(G) \cdot \text{ind}(G) \geq n$ , t.j. súčin chromatického čísla grafu  $G$  a čísla maximálnej nezávislosti grafu  $G$  je väčší než počet vrcholov grafu  $G$ .  
Návod: Optimálne zafarbenie grafu  $G$  definuje rozklad vrcholovej množiny  $V$  na  $\chi(G)$  nezávislých podmnožín.
3. Ukážte, že  $\chi(G) + \text{ind}(G) \leq n + 1$ . Návod: Ofarbíme najpočetnejšiu nezávislú množinu farbou 1. K ofarbeniu  $n - \text{ind}(G)$  vrcholov stačí  $n - \text{ind}(G)$  farieb.

### Počítačové cvičenia

4. Naprogramujte algoritmus 9.1 na hľadanie váženého  $p$ -mediánu. Funkciu pre výpočet  $f(D_p)$  naprogramujte tak, aby sa dala jednoducho zameniť za funkciu  $\text{ecc}(D_p)$ .  
Experimentujte s rôznymi štartovacími riešeniami a rôznymi stratégiami výberu nového lepšieho riešenia tak, ako sú popísané na str. 248 pri rozbere algoritmu 9.1. Skúste, do akej veľkosti  $n = |V|$  a  $p$  ste schopní nájsť vážený  $p$ -medián prejdением všetkých možností. Porovnajzte takto získané exaktné výsledky s výsledkami heuristik.

5. Naprogramujte rekurzívnu a nerekurzívnu verziu algoritmu 9.2.

# Dodatok A

## Anglicko – slovenský slovníček

### A

**acyclic graph** acyklický graf  
**adjacent edges** príľahlé (susedné)  
hrany  
**adjacent vertices** susedné vrcholy  
**adjacency matrix** matica príľahlosti  
(susednosti)  
**algorithm** algoritmus  
**ancestor of a vertex** predchodca  
vrchola (v koreňových stromoch)  
**AOA (activity-on-arc) network**  
sieťový digraf, v ktorom hrana  
modeluje činnosť  
**AON (activity-on-node) network**  
precedenčný digraf  
**arc** orientovaná hrana  
**assignment problem** priraďovacia  
úloha

### B

**binary tree** binárny strom  
**bipartite graph** bipartitný graf  
**Breadth-First Search**  
prehľadávanie (grafu) do šírky

### C

**center of a graph** centrum grafu  
**child of a vertex** bezprostredný  
následník vrchola (v koreňovom  
strome)  
**Chinese Postman Problem** úloha  
čínskeho poštára  
**chromatic number of a graph**  
chromatické číslo grafu  
**clique in a graph** klika v grafe  
(úplný podgraf grafu)  
**clique number of a graph** klikové  
číslo grafu  
**closed walk (trail)** uzavretý sled  
(ťah)

**coloring of a graph** farbenie grafu

**$k$ -colorable graph**  $k$ -zafarbiteľný graf

**complement of a graph**

komplement grafu

**comparable elements** porovnateľné

prvky (v súvislosti s reláciou precedencie pri sieťovom plánovaní)

**component of a graph** komponent grafu

**complete bipartite graph** úplný

bipartitný graf

**complete graph** úplný (kompletný) graf

**computational complexity**

výpočtová zložitosť

**concentration of walks** zretáženie

sledov

**connected graph** súvislý graf

**critical activity** kritická činnosť

(v metóde CPM)

**critical path** kritická cesta

**cycle** cyklus

## D

**degree of a vertex** stupeň vrchola

**degree sequence** valenčná

postupnosť

**Depth-First Search** prehľadávanie

(grafu) do hĺbky

**depth of a vertex** úroveň vrchola

(v koreňovom strome)

**descendant of a vertex** následník vrchola

**diameter of a graph** priemer grafu

**digraph, directed graph** digraf,

orientovaný graf

**directed distance** orientovaná

vzdialenosť – dĺžka najkratšej

orientovanej  $u$ - $v$  cesty

**directed edge** orientovaná hrana

**directed walk (trail, path)**

orientovaný sled (ťah, cesta)

**directed tree** orientovaný strom

**distance** vzdialenosť

## E

**earliest event time** najskôr možný

začiatok činnosti (pri časovom

plánovaní)

**edge** hrana

**edge-weight** ohodnotenie hrany

**endpoint of an edge** koncový vrchol

hrany

**eulerian graph** eulerovský graf

**eulerian tour** eulerovský ťah

## F

**feasible flow in a network**

pripustný tok v sieti (napr.

intervalovo ohodnotenej)

**flow in a network** tok v sieti

**forest** les, acyklický graf

## G

**graph** graf (pseudomigraf)

**H**

**hamiltonian cycle** hamiltonovský cyklus

**hamiltonian graph** hamiltonovský graf

**hamiltonian path** hamiltonovská cesta

**head of an arc** koncový (druhý) vrchol orientovanej hrany

**heap** halda (reprezentácia prioritného stromu)

**height of a rooted tree** výška koreňového stromu

**Huffman code** Huffmanov kód

**I**

**immediate predecessor** bezprostredný predchodca

**immediate successor** bezprostredný následník

**incidence** incidencia (vzťah vrchola a hrany)

**incidence matrix of a graph** incidenčná matica grafu

**indegree of a vertex** vstupný stupeň vrchola

**induced subgraph on an edge-set** graf indukovaný množinou hrán

**induced subgraph on an vertex-set** graf indukovaný množinou vrcholov

**isomorphic graphs, digraphs** izomorfné grafy, digrafy

**isomorphism of graphs**

izomorfizmus grafov

**L**

**latest event time** najneskôr nutný koniec činnosti (pri časovom plánovaní)

**leaf in a rooted tree** koncový vrchol v koreňovom strome

**length of a walk** dĺžka sledu

**level of a vertex** úroveň vrchola (v koreňovom strome)

**linear programming** lineárne programovanie

**M**

**matching in a graph** párenie v grafe

**maximum capacity route** cesta maximálnej kapacity

**maximum cardinality matching** najpočetnejšie párenie

**maximum flow** maximálny tok (tok s najväčšou veľkosťou)

**maximum matching** maximálne párenie

**maximum weight matching** najcennejšie párenie

**median of a graph** medián grafu

**minimum cost maximum flow** maximálny tok s minimálnou cenou

**minimum cut of a network** rez s minimálnou kapacitou (v sieti)

**migraph, mixed graph, partially directed graph** migraf  
**multi-edge** množina násobných hrán  
**multi-arc** množina násobných orientovaných hrán

## N

**negative cycle** cyklus so zápornou cenou  
**neighbor of a vertex** susedný vrchol  
**neighborhood of a vertex** okolie vrchola  
**network** sieť  
**normalized drawing of a graph** digram grafu  
**node** vrchol (často len v digrafe)

## O

**odd cycle** cyklus s nepárnym počtom hrán, nepárny cyklus  
**open walk** otvorený sled  
**outdegree of a vertex** výstupný stupeň vrchola

## P

**partial order** čiastočné usporiadanie  
**partially ordered set (poset)** čiastočne usporiadaná množina  
**path** cesta  
**parent of a vertex** otec vrchola, bezprostredný predchodca  
**perfect matching** úplné párenie  
**planar graph** rovinný graf

**priority tree** prioritný strom  
**proper subgraph** pravý podgraf, podgraf rôzny od pôvodného grafu

## R

**radius of a graph** polomer grafu  
**reachability relation** dosiahnuteľnosť, relácia dosiahnuteľnosti  
**root of a tree** koreň stromu  
**rooted tree** koreňový strom

## S

**self-loop** slučka, hrana typu  $\{v, v\}$ ,  $(v, v)$   
**simple graph, digraph** graf resp. digraf; „simple“ zdôrazňuje neexistenciu násobných hrán a slučiek  
**single source – single sink network** sieť s jedným zdrojom a jedným ústím  
**spanning subgraph of a graph** faktorový podgraf grafu

**spanning tree** kostra grafu  
**strongly connected digraph** silne súvislý digraf  
**subdigraph** podgraf digrafu  
**subgraph** podgraf  
**subwalk** podsled (súvislá podpostupnosť sledu)

## T

**transitivity** tranzitívnosť



**transitive digraph** tranzitívny  
digraf

**transportation problem** dopravná  
úloha

**tail of an arc** začiatočný vrchol  
orientovanej hrany

**trail** ťah

**Travelling Salesman Problem**  
úloha obchodného cestujúceho

**tree** strom

**trivial graph** triviálny graf (graf  
s jednoprvkovou množinou vrcholov)

## V, W

**valency** stupeň vrchola

**value of a flow** veľkosť toku

**vertex** vrchol

**walk** sled

**weighted graph (digraph)** graf  
(digraf) s ohodnotením hrán alebo  
vrcholov

**weighted  $p$ -center of a graph**  
vážené  $p$ -centrum grafu

**weighted  $p$ -median of a graph**  
vážený  $p$ -medián grafu



# Register

- algoritmus, 45
  - $\rho$ -aproximačný, 55
  - Dijkstrov, 80
  - Edmondsov, 169
  - Euklidov, 46
  - Fleuryho, 163
  - Floydov, 84
  - Fordov, 79
  - Fordov–Fulkersonov, 199
  - genetický, 57
  - Huffmanov, 125
  - kostry a párenia, 175
  - Kruskalov, 116
  - label correct, 88
  - label set, 88
  - labyrintový, 164
  - na hľadanie cesty max. priepustnosti, 121
  - na monotónne očíslovanie acyklického digrafu, 136
  - polynomiálny, 47
  - prehľadávania okolí, 177
  - suboptimálny, 173
  - Tarryho, 66
  - základný, 72
  - zdvojenia kostry, 174
  - zložitosti  $O(f(n))$ , 47
- artikulácia, 63
- atrakčný obvod depa, 249
  - prvotný, 250
- breadth-first search, 112
- cena
  - hrany, 30
  - kostry, 113
  - párenia, 168
  - polocyklu, 206
  - polosledu, 71
  - sledu, 71
  - toku, 205
  - vrchola, 30
- centrum grafu, 83
- cesta, 59
  - kritická, 146
  - maximálnej spoľahlivosti, 94
  - najdlhšia, 139
  - najkratšia, 71
  - orientovaná, 60
- cyklus, 62
  - orientovaný, 62
  - zápornej ceny, 90
- časová rezerva činnosti, 146
- činnosť
  - elementárna, 141
  - fiktívna, 154
  - kritická, 146
- číslo

- chromatické - grafu, 225
- klikové - grafu, 251
- maximálnej nezávislosti grafu, 251
- depo, 246
- depth-first search, 112
- diagram grafu, 19
  - rovinný, 19
- diameter grafu, 83
- digraf, 18
  - acyklický, 131
  - bezprostrednej precedencie, 143
  - hranovo ohodnotený, 30
  - jednostranne súvislý, 65
  - neorientovane súvislý, 64
  - nesúvislý, 65
  - orientovane súvislý, 65
  - precedencie, 143
  - precedenčný, 143
  - sieťový, 152
  - silne súvislý, 65
  - slabo súvislý, 64
  - tranzitívny, 140
  - úplný, 22
  - vrcholovo ohodnotený, 30
- digrafy
  - komplementárne, 29
- dĺžka
  - polosledu, 71
  - sledu, 71
- dvojica
  - neusporiadaná, 16
  - usporiadaná, 13
- ekvivalencia, 15
- excentricita množiny
  - vážená, 248
- excentricita vrchola, 83
- graf, 18
  - $k$ -zafarbiteľný, 225
  - acyklický, 105
  - bipartitný, 29
  - eulerovský, 162
  - hamiltonovský, 171
  - hranovo ohodnotený, 30
  - hranový, 30
  - indukovaný sledom, 63
  - kompletný, 22
  - nesúvislý, 63
  - planárny, 19
  - pravidelný - stupňa  $k$ , 29
  - rovinný, 19, 220
  - súvislý, 63
  - triviálny, 105
  - úplný, 22
  - úplný bipartitný, 29
  - vrcholovo ohodnotený, 30
- grafy
  - homeomorfné, 224
  - komplementárne, 29
- greedy algorithm, 57
- halda, 156
- heuristika, 173
  - vytvárajúca, 56
  - zlepšujúca, 56
- hrana
  - hraničná, 111
  - nasýtená, 191
  - orientovaná, 18
- hrana grafu, 18
- hrany
  - prilahlé, 23
  - susedné, 23
- hviezda vrchola
  - výstupná - -, 24
- hviezda vrchola
  - vstupná - -, 24

- incidencia, 23
- invarianty izomorfizmu, 31
- izomorfizmus digrafo, 31
- izomorfizmus grafov, 31
- izomorfné digrafy, 31
- izomorfné grafy, 31
  
- kapacita hrany, 119
- karteziánsky súčin množín, 13
- klika, 251
  - najpočetnejšia, 251
- kocka  $n$ -rozmerná, 185
- kód, 123
  - Grayov, 185
  - Huffmanov, 125
  - prefixový, 123
- komponent digrafu, 65
- komponent grafu, 63
- koniec vykonávania projektu, 152
- koreň, 109
- kostra
  - najdrahšia, 113
  - najlacnejšia, 113
- kostra grafu, 113
- kostry
  - susedné, 115
- kružnica, 63
  
- list, 110
- lokálne minimum, 249
  
- matica
  - incidenčná, 39
  - ohodnotení hrán digrafu, 36
  - ohodnotení hrán grafu, 36
  - príľahlosti, 35
- metóda
  - pažravá, 57
  - prehľadávania okolí, 57
  - vetiev a hraníc, 57
- metrika, 83
- množina
  - čiasťočne usporiadaná, 17
  - dominujúca, 254
    - minimálna, 254
  - hranová, 18
  - lineárne usporiadaná, 17
  - nezávislá, 250
    - maximálna, 251
    - najpočetnejšia, 251
  - rezová, 194
  - úplná, 251
    - maximálna, 251
  - vrcholová, 18
- monotónne očíslovanie, 135
- most, 63
- multidigraf, 20
- multigraf, 20
- multimigraf, 20
  
- $n$ -tica usporiadaná, 13
- najneskôr nutný koniec
  - elementárnej činnosti, 146
- najskôr možný začiatok
  - elementárnej činnosti, 146
- následník činnosti, 142
  - bezprostredný, 142
- neighborhood search, 57
  
- $O(f(n))$  – algoritmus, 47
- oblúk, 18
- ohodnotenie hrany, 30
- ohodnotenie vrchola, 30
- okolie vrchola, 24
  
- $p$ -centrum, 248
  - vážené, 248
- $p$ -medián, 248

- vážený, 247
- párenie
  - maximálne, 168
  - najpočetnejšie, 168
  - úplné, 168
- párenie v grafe, 168
- pestovanie stromu, 111
- podgraf digrafu, 22
  - faktorový, 22
  - indukovaný množinou hrán, 23
  - indukovaný množinou vrcholov, 23
- podgraf grafu, 22
  - faktorový, 22
  - indukovaný množinou hrán, 23
  - indukovaný množinou vrcholov, 23
- polocesta, 60
  - rezervná, 191
  - zväčšujúca, 192
- polocyklus, 62
  - rezervný, 206
- polomer grafu, 83
- posled, 60
  - otvorený, 62
  - uzavretý, 62
- poloťah, 60
- postupnosť
  - de-Bruijnovská, 181
  - spätná, 164
  - valenčná - grafu, 27
- prameň, 134
- predchodca činnosti, 142
  - bezprostredný, 142
- prehľadávanie do hĺbky, 112
- prehľadávanie do šírky, 112
- priemer grafu, 83
- priepustnosť hrany, 119
- priepustnosť rezovej množiny, 194
- priepustnosť sledu, 120
- problém
  - NP-ekvivalentný, 55
  - NP-ľahký, 55
  - NP-ťažký, 55
- problémy
  - polynomiálne ekvivalentné, 53
- prvky
  - neporovnateľné, 17
  - porovnateľné, 17
- pseudodigraf, 20
  - deBruijnovský, 182
- pseudograf, 20
- pseudomigraf, 20
- rádus grafu, 83
- redukcia polynomiálna, 53
- relácia
  - antireflexívna, 14
  - antisymetrická, 14
  - binárna, 14
  - ekvivalencie, 15
  - precedencie, 142
  - precedenčná, 142
  - reflexívna, 14
  - symetrická, 14
  - tranzitívna, 14
- reprezentant triedy ekvivalencie, 15
- reťazec, 17
- rezerva
  - hrany, 191
  - polocesty, 191
- rozklad množiny, 15
- rozpolenie hrany, 224
- rozvrh
  - prípustný, 145
  - úlohy časového plánovania, 145
- sieť, 189

- intervalovo ohodnotená, 202
- simulated annealing, 57
- sled, 59
  - eulerovský, 162
  - hamiltonovský, 171
  - orientovaný, 60
  - otvorený, 62
  - Tarryho, 66
  - triviálny, 59
  - uzavretý, 62
- slovo
  - kódové, 123
  - nekódové, 123
- slučka, 18
- spoľahlivosť cesty, 94
- stena
  - rovinného diagramu, 220
  - vnútorná, 220
  - vonkajšia, 220
- stok, 134
- stredisko havarijné, 246
- strom, 105
  - binárny koreňový, 110, 134
  - koreňový, 109, 134
  - orientovaný, 131
  - prioritný, 156
- stupeň vrchola, 25
  - vstupný, 25
  - výstupný, 25
- systém pridelených
  - atrakčných obvodov, 250
- šíp, 18
- tabu search, 57
- technologická tabuľka projektu, 143
- tok hranou, 191
- tok v sieti, 191
  - maximálny, 191
  - najlacnejší, 206
  - prípustný, 202
- transformácia polynomiálna, 52
- tranzitívna redukcia, 140
- tranzitívny uzáver, 140
- trieda ekvivalencie, 15
- trieda rozkladu, 15
- trvanie projektu, 145
- trvanie rozvrhu, 145
- ťah, 59
  - eulerovský, 162
  - eulerovský orientovaný, 163
  - orientovaný, 60
  - otvorený, 62
  - uzavretý, 62
- úloha
  - časového plánovania, 143
  - čínskeho pošťára, 167
  - dopravná, 213
    - s medziskladmi, 214
  - lineárneho programovania, 49
    - bivalentného (binárneho), 50
    - celočíselného, 49
  - NP-ťažká, 53
  - obchodného cestujúceho, 171
  - priradovacia, 213
- úroveň vrchola, 109
- usporiadanie, 17
  - lineárne, 17
- ústie, 189
- veľkosť toku, 191
- vrchol
  - centrálny - grafu, 83
  - dosiahnuteľný z vrchola  $u$ , 62
  - koncový - stromu, 110
  - voľný, 111

- zaradený, 111
- vrchol grafu, 18
- vrchol hrany
  - koncový, 23
  - začiatočný, 23
- vrcholy
  - príľahlé, 23
  - susedné, 23
- výška koreňového stromu, 109
- vzdialenosť vrchola a množiny, 246
- vzdialenosť vrcholov, 83
  
- začiatok vykonávania projektu, 152
- zafarbenie
  - grafu, 225
    - prípustné, 225
    - systematické, 231
- zdroj, 189
- zložitosť algoritmov, 45
- zložitosť problému, 47
- zreťazenie sledov, 62



# Literatúra

- [1] CONWAY, W.,R., MAXWELL, W.,L.,MILLER, L.,W.: *Theory of Scheduling*. Addison-Wesley Publishing Company, (1967)
- [2] Nicos Christofides: *Graph theory – an Algorithmic Approach*, Academic Press, London (1975). Ruský preklad, Moskva (1978)
- [3] Jiří Demel: *Grafy*, SNTL, Praha (1989)
- [4] Jiří Demel: *Teorie grafů*, ČVUT Praha (1984)
- [5] James R. Evans, Edward Minieka: *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, Inc. (1992), second edition, ISBN 0-8247-8602-5
- [6] Martin Fronc: *Teória grafov*, VŠDS Žilina (1993), ISBN 80-7100-119-8
- [7] Jana Galanová, Peter Kaprálik: *Diskrétna matematika*, STU, Bratislava (1997), ISBN 80-227-0942-5
- [8] Gliviak, F., Vadkerti, P.: *Sieťová analýza a manažment projektov*, skriptá, UCM Trnava (2001)
- [9] Johnatan Gross, Jay Yellen: *Graph Theory and its Applications*, CRC Press (1999), ISBN 0-8493-3982-0
- [10] J. Nešetřil: *Teorie grafů*, Matematický seminář, SNTL, Praha (1979)
- [11] Luděk Kučera: *Kombinatorické algoritmy*, Matematický seminář SNTL, Praha (1983)

- 
- [12] PINEDO, M.: *Scheduling: Theory, Algorithms, and Systems (2nd Edition)*, Prentice Hall; 2 edition (August 8, 2001), ISBN: 0130281387
- [13] Ján Plesník: *Grafové algoritmy*, Veda, Bratislava (1983)
- [14] *Modern Heuristic Techniques for Combinatorial Problems* McGRAW-HILL BOOK COMPANY, London (1995) ISBN 0-07-709239-2
- [15] Kenneth H. Rosen: *Handbook of Discrete and Combinatorial Mathematics*, CRC Press (2000), ISBN 0-8493-0149-1
- [16] J. Sedláček: *Úvod do teorie grafů*, Academia Praha, (1981)
- [17] Unčovský, L. a kolektív: *Modely sieťovej analýzy*, Alfa Bratislava, (1991) ISBN 80-05-00812-0

Autor:	Doc. RNDr. Stanislav Palúch, CSc.
Názov:	<b>ALGORITMICKÁ TEÓRIA GRAFOV</b>
Vydala:	Žilinská univerzita v EDIS-vydavateľstve ŽU v xxxxx 2008 ako svoju XXX. publikáciu
Zodpovedný redaktor:	Xxxxxx Xxxxxxx
Technický redaktor:	Xxxxxxxxxx XXxxxx
Určené:	pre študentov Fakulty riadenia a informatiky
Vydanie:	prvé
Náklad:	XXX výtlačkov
AH/VH:	XX/XX
Druh tlače:	ofset
Typografický systém:	L <sup>A</sup> T <sub>E</sub> X pod o. s. Linux
ISBN 80-XXXX-XX-X	