Acyclic digraphs

Stanislav Palúch

Fakulta riadenia a informatiky, Žilinská univerzita

28. apríla 2016

Acyclic digraph, directed tree

Definition

An acyclic digraph *is a digraph in which there is no directed cycle.*

Definition

A directed tree *is a weekly connected digraph in which is no quasi-cycle.*

Remark

If $\overrightarrow{G} = (V, H)$ *as an acyclic digraph then it can not contain both arcs* (u, v) *and* (v, u) *simultaneously, since, in this case, it would contain also following cycle* $(u, (u, v), v, (v, u), u)$.

Remark

$(u, (u, v), v, (u, v), u)$ *is not a quasi-cycle, since it contains the same arc* (u, v) *twice.*

Acyclic digraph, directed tree

Definition

An acyclic digraph *is a digraph in which there is no directed cycle.*

Definition

A directed tree *is a weekly connected digraph in which is no quasi-cycle.*

Remark

If $\overrightarrow{G} = (V, H)$ *as an acyclic digraph then it can not contain both arcs* (u, v) *and* (v, u) *simultaneously, since, in this case, it would contain also following cycle* $(u, (u, v), v, (v, u), u)$.

Remark

$(u, (u, v), v, (u, v), u)$ *is not a quasi-cycle, since it contains the same arc* (u, v) *twice.*

Acyclic digraph, directed tree

Definition

An acyclic digraph *is a digraph in which there is no directed cycle.*

Definition

A directed tree *is a weekly connected digraph in which is no quasi-cycle.*

Remark

If $\overrightarrow{G} = (V, H)$ *as an acyclic digraph then it can not contain both arcs* (u, v) *and* (v, u) *simultaneously, since, in this case, it would contain also following cycle* $(u, (u, v), v, (v, u), u)$.

Remark

$(u, (u, v), v, (u, v), u)$ *is not a quasi-cycle, since it contains the same arc* (u, v) *twice.*

Acyclic digraph, directed tree

Definition

An acyclic digraph *is a digraph in which there is no directed cycle.*

Definition

A directed tree *is a weekly connected digraph in which is no quasi-cycle.*

Remark

If $\overrightarrow{G} = (V, H)$ *as an acyclic digraph then it can not contain both arcs* (u, v) *and* (v, u) *simultaneously, since, in this case, it would contain also following cycle* $(u, (u, v), v, (v, u), u)$.

Remark

$(u, (u, v), v, (u, v), u)$ *is not a quasi-cycle, since it contains the same arc* (u, v) *twice.*

Acyclic digraph

We can create for every acyclic digraph $\overrightarrow{G} = (V, H)$ a (non directed) graph $G' = (V, H')$ with the same vertex set V and with edge set H' defined as

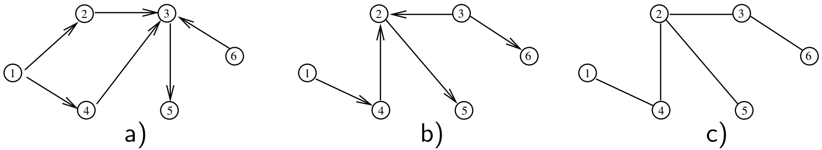
$$H' = \{\{u, v\} \mid (u, v) \in H\} \ . \tag{1}$$

Edge set H' is the arc set H in which we „forget" direction.

Since an acyclic digraph can contain at most one arc from (u, v), (v, u) for every pair of vertices $u \in V$, $v \in V$, it bholds

$$|H'| = |H|.$$

Acyclic digraph



Obr.: a) Weakly connected acyclic digraph,
which is not a directed tree.

b) Directed tree $\overrightarrow{G}$.

c) Non directed tree corresponding to $\overrightarrow{G}$.

Properties of directed trees

Theorem

Following assertions are equivalent:

a) *Digraph* $\overrightarrow{G} = (V, H)$ *je a directed tree.*

b) *There exists exactly one u–v quasi-path in digraph* $\overrightarrow{G} = (V, H)$ *for every* $u, v \in V$.

c) *Digraph* $\overrightarrow{G} = (V, H)$ *is weakly connected and every arc of arc set* H *is a bridge in* $\overrightarrow{G} = (V, H)$.

(A bridge in a digraph is such an arc, after removing it the number of components rises.)

d) *Digraph* $\overrightarrow{G} = (V, H)$ *is weakly connected and* $|H| = |V| - 1$.

e) *Digraph* $\overrightarrow{G} = (V, H)$ *does not contain a quasi-cycle and it holds* $|H| = |V| - 1$.

Theorem

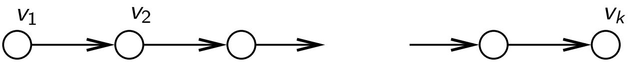
Let $\overrightarrow{G} = (V, H)$ be an acyclic digraph.
Then V contains at least one vertex z such that $\mathrm{ideg}(z) = 0$ and at least one vertex u such that $\mathrm{odeg}(u) = 0$.

PROOF.

Let

$$\boldsymbol{\mu}(v_1, v_k) = (v_1, (v_1, v_2), v_2, \ldots, (v_{k-1}, v_k), v_k)$$

be a directed path in digraph $\overrightarrow{G}$ with maximum number of arcs.
We show that $\mathrm{odeg}(v_k) = 0$.



If $\mathrm{odeg}(v_k) > 0$,
then there exists at least on arc (dashed) outgoing from v_k,
which extends path $\boldsymbol{\mu}(v_1, v_k)$ (contradiction with path havin maximu number of arcs)
or closes a cycle (contradiction with acyclicity of $\overrightarrow{G}$)

Properties of acyclic digraphs

Theorem

Let $\overrightarrow{G} = (V, H)$ be an acyclic digraph.
Then V contains at least one vertex z such that $\mathrm{ideg}(z) = 0$
and at least one vertex u such that $\mathrm{odeg}(u) = 0$.

Proof.

Let
$$\boldsymbol{\mu}(v_1, v_k) = (v_1, (v_1, v_2), v_2, \dots, (v_{k-1}, v_k), v_k)$$
be a directed path in digraph $\overrightarrow{G}$ with maximum number of arcs.
We show that $\mathrm{odeg}(v_k) = 0$.

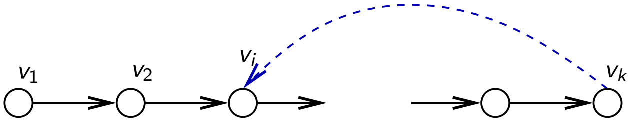
If $\mathrm{odeg}(v_k) > 0$,
then there exists at least on arc (dashed) outgoing from v_k,
which extends path $\boldsymbol{\mu}(v_1, v_k)$ (contradiction with path havin maximu number of arcs)
or closes a cycle (contradiction with acyclicity of $\overrightarrow{G}$)

Theorem

Let $\overrightarrow{G} = (V, H)$ be an acyclic digraph.
Then V contains at least one vertex z such that $\mathrm{ideg}(z) = 0$ and at least one vertex u such that $\mathrm{odeg}(u) = 0$.

PROOF.

Let
$$\boldsymbol{\mu}(v_1, v_k) = (v_1, (v_1, v_2), v_2, \ldots, (v_{k-1}, v_k), v_k)$$
be a directed path in digraph $\overrightarrow{G}$ with maximum number of arcs.
We show that $\mathrm{odeg}(v_k) = 0$.



If $\mathrm{odeg}(v_k) > 0$,
then there exists at least on arc (dashed) outgoing from v_k,
which extends path $\boldsymbol{\mu}(v_1, v_k)$ (contradiction with path havin maximu number of arcs)
or closes a cycle (contradiction with acyclicity of $\overrightarrow{G}$)

Topological ordering of acyclic digraph

Theorem

A digraph $\overrightarrow{G} = (V, H)$ is acyclic if and only if its vertex set V can be ordered into sequence

$$v_1, v_2, \dots, v_n \tag{2}$$

so that it holds:

$$\textit{If } (v_i, v_k) \in H \quad \textit{then} \quad i < k. \tag{3}$$

Definition

Numbering of vertices $v_1, v_2, \dots, v_n$ of an acyclic digraph $\overrightarrow{G} = (V, H)$ for which it holds:

$$\textit{if } (v_i, v_k) \in H, \ \textit{then } i < k,$$

is called **topological ordering** *of vertices of acyclic digraph $\overrightarrow{G}$.*

Topological ordering of acyclic digraph

Theorem

A digraph $\overrightarrow{G} = (V, H)$ is acyclic if and only if its vertex set V can be ordered into sequence

$$v_1, v_2, \dots, v_n \tag{2}$$

so that it holds:

$$\text{If } (v_i, v_k) \in H \quad \text{then} \quad i < k. \tag{3}$$

Definition

Numbering of vertices $v_1, v_2, \dots, v_n$ of an acyclic digraph $\overrightarrow{G} = (V, H)$ for which it holds:

$$\text{if } (v_i, v_k) \in H, \text{ then } i < k,$$

is called **topological ordering** *of vertices of acyclic digraph $\overrightarrow{G}$.*

Algorithm

Algorithm I. for topological ordering of acyclic digraph $\overrightarrow{G} = (V, H)$.

- **Step 1.** *Set* $i := 1$.
- **Step 2.** $\{$*Digraph* $\overrightarrow{G} = (V, H)$ *contains at least one vertex such that* $v \in V$, *že* $\mathrm{ideg}(v) = 0.\}$
  Take such vertex $v \in V$ *for which* $\mathrm{ideg}(v) = 0$ *and set* $v_i := v$.
- **Step 3.** *If* $V - \{v\} = \emptyset$ *STOP,*
  otherwise $\overrightarrow{G} := \overrightarrow{G} - \{v\}$, $i := i + 1$ *and Goto Step 2.*

♣

Algorithm

Algorithm I. for topological ordering of acyclic digraph $\overrightarrow{G} = (V, H)$.

- **Step 1.** *Set* $i := 1$.
- **Step 2.** $\{$*Digraph* $\overrightarrow{G} = (V, H)$ *contains at least one vertex such that* $v \in V$, *že* $\mathrm{ideg}(v) = 0.\}$
  Take such vertex $v \in V$ *for which* $\mathrm{ideg}(v) = 0$ *and set* $v_i := v$.
- **Step 3.** *If* $V - \{v\} = \emptyset$ *STOP,*
  otherwise $\overrightarrow{G} := \overrightarrow{G} - \{v\}$, $i := i + 1$ *and Goto Step 2.*

♣

Algorithm

Algorithm I. for topological ordering of acyclic digraph $\overrightarrow{G} = (V, H)$.

- **Step 1.** *Set* $i := 1$.
- **Step 2.** $\{$*Digraph* $\overrightarrow{G} = (V, H)$ *contains at least one vertex such that* $v \in V$, *že* $\mathrm{ideg}(v) = 0.\}$
  Take such vertex $v \in V$ *for which* $\mathrm{ideg}(v) = 0$ *and set* $v_i := v$.
- **Step 3.** *If* $V - \{v\} = \emptyset$ *STOP,*
  otherwise $\overrightarrow{G} := \overrightarrow{G} - \{v\}$, $i := i + 1$ *and Goto Step 2.*

♣

Algorithm

Algorithm II. for topological ordering of acyclic digraph $\overrightarrow{G} = (V, H)$.

- **Step 1.** *Assign a label* $d(v) := \mathrm{ideg}(v)$ *for every vertex* $v \in V$.

  Determine the subset $V_0 \subseteq V$ *of vertex set* V *containing all vertices with zero label* $d(\ )$, *i. e.*

  $$V_0 = \{v \mid v \in V,\ d(v) = 0\}.$$

  Set $k := |V_0|$ *and order the elements of* V_0 *into arbitrary sequence* $\mathcal{P} = v_1, v_2, \dots, v_k$.

  Set $i := 1$. *Set* $r := i$.

- **Step 2.** *For all vertices* $w \in V^+(r)$ *do:*
  $d(w) := d(w) - 1$. *If* $d(w) = 0$ *then set* $k := k + 1$, $v_k := w$.

- **Step 3.** *If* $k = n = |V|$ *then STOP. Otherwise set* $i := i + 1$, $r := v_i$ *and GOTO Step 2.*

Algorithm

Algorithm II. for topological ordering of acyclic digraph $\overrightarrow{G} = (V, H)$.

- **Step 1.** *Assign a label* $d(v) := \mathrm{ideg}(v)$ *for every vertex* $v \in V$.

  Determine the subset $V_0 \subseteq V$ *of vertex set* V *containing all vertices with zero label* $d(\ )$, *i. e.*

  $$V_0 = \{v \mid v \in V,\ d(v) = 0\}.$$

  Set $k := |V_0|$ *and order the elements of* V_0 *into arbitrary sequence* $\mathcal{P} = v_1, v_2, \dots, v_k$.

  Set $i := 1$. *Set* $r := i$.

- **Step 2.** *For all vertices* $w \in V^+(r)$ *do:*
  $d(w) := d(w) - 1$. *If* $d(w) = 0$ *then set* $k := k + 1$, $v_k := w$.

- **Step 3.** *If* $k = n = |V|$ *then STOP. Otherwise set* $i := i + 1$, $r := v_i$ *and GOTO Step 2.*

Algorithm

Algorithm II. for topological ordering of acyclic digraph $\overrightarrow{G} = (V, H)$.

- **Step 1.** *Assign a label* $d(v) := \mathrm{ideg}(v)$ *for every vertex* $v \in V$.

  Determine the subset $V_0 \subseteq V$ *of vertex set* V *containing all vertices with zero label* $d(\ )$, *i. e.*

  $$V_0 = \{v \mid v \in V,\ d(v) = 0\}.$$

  Set $k := |V_0|$ *and order the elements of* V_0 *into arbitrary sequence* $\mathcal{P} = v_1, v_2, \dots, v_k$.

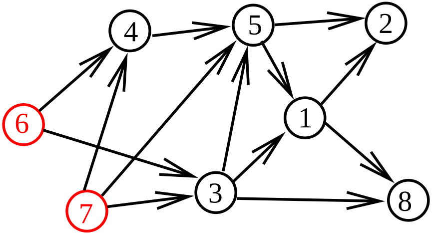
  Set $i := 1$. *Set* $r := i$.

- **Step 2.** *For all vertices* $w \in V^+(r)$ *do:*
  $d(w) := d(w) - 1$. *If* $d(w) = 0$ *then set* $k := k + 1$, $v_k := w$.

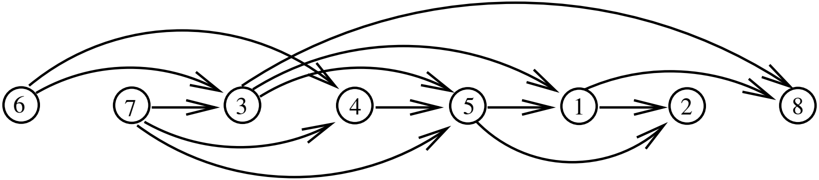
- **Step 3.** *If* $k = n = |V|$ *then STOP. Otherwise set* $i := i + 1$, $r := v_i$ *and GOTO Step 2.*

♣

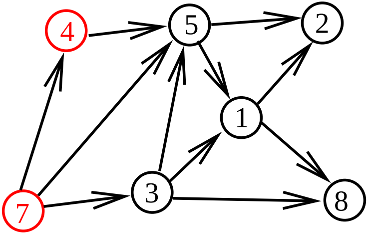
Topological ordering of acyclic digraph



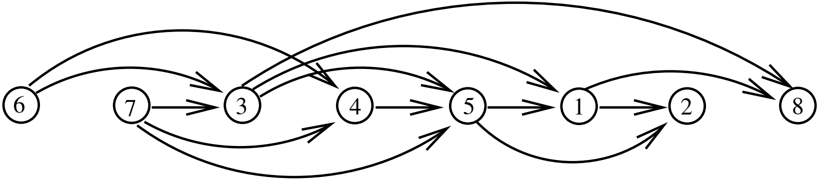
i	$v(i)$	1	2	3	4	5	6	7	8
		$d(v)$							
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



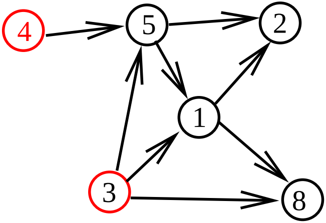
Topological ordering of acyclic digraph



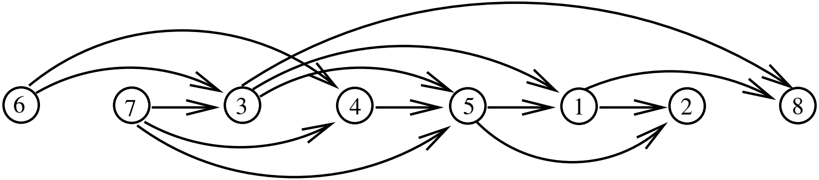
i	$v(i)$	1	2	3	4	5	6	7	8
		$d(v)$							
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



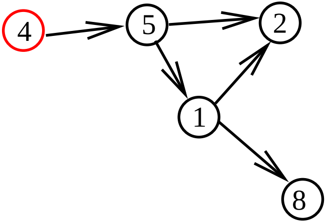
Topological ordering of acyclic digraph



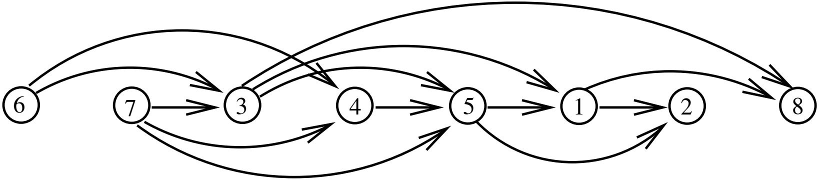
i	$v(i)$	1	2	3	4	5	6	7	8
		$d(v)$							
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



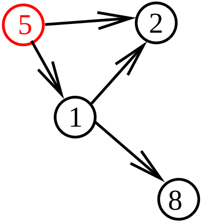
Topological ordering of acyclic digraph



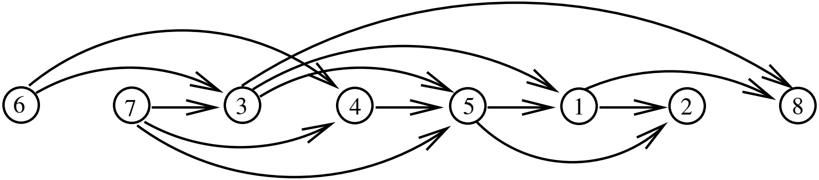
i	$v(i)$	1	2	3	4	5	6	7	8
		$d(v)$							
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



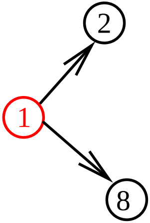
Topological ordering of acyclic digraph



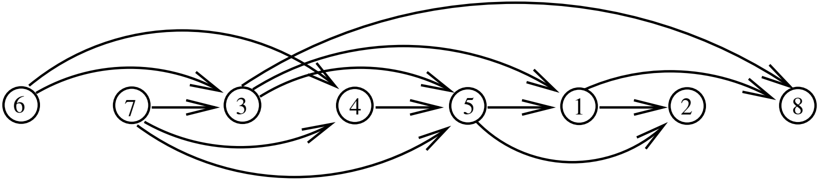
i	$v(i)$	1	2	3	4	5	6	7	8
		$d(v)$							
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



Topological ordering of acyclic digraph

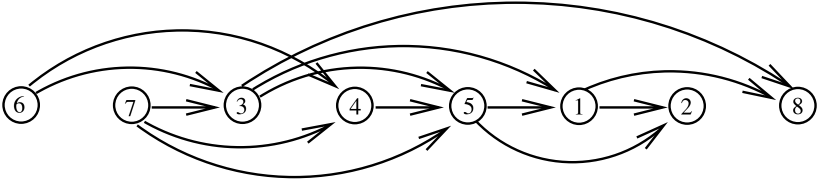


i	$v(i)$	1	2	3	4	5	6	7	8
		$d(v)$							
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



Topological ordering of acyclic digraph

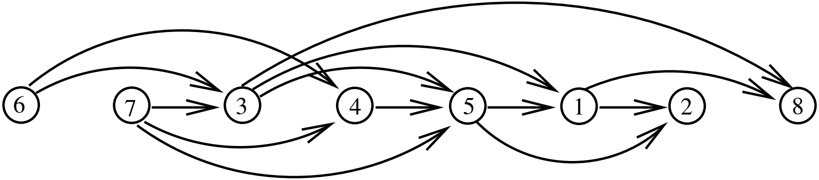
	i	$v(i)$	1	2	3	4	5	6	7	8
			$d(v)$							
	-	-	2	2	2	2	3	0	0	2
2	1	6	2	2	**1**	**1**			0	2
	2	7	2	2	**0**	**0**	**2**			2
	3	3	**1**	2			**1**			**1**
	4	4	1	2			**0**			1
	5	5	**0**	**1**						1
	6	1		**0**						**0**
8	7	2								
	8	8								



Topological ordering of acyclic digraph

8

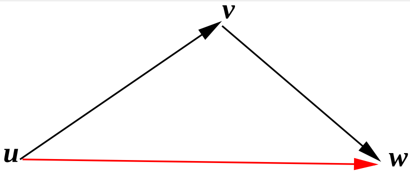
i	$v(i)$	1	2	3	4	5	6	7	8
					$d(v)$				
-	-	2	2	2	2	3	0	0	2
1	6	2	2	**1**	**1**			0	2
2	7	2	2	**0**	**0**	**2**			2
3	3	**1**	2			**1**			**1**
4	4	1	2			**0**			1
5	5	**0**	**1**						1
6	1		**0**						**0**
7	2								
8	8								



Transitieve digraph

Definition

An acyclic digraph $\overrightarrow{G} = (V, H)$ is **transitive***, if for every two arcs $(u, v) \in H$, $(v, w) \in H$ there exists arc $(u, w) \in H$.*



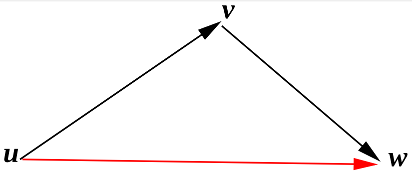
In a transitive digraph there is a direct arc (v, w) for every pair of arcs (u, v), (v, w)

Theorem

An acyclic digraph $\overrightarrow{G}$ is transitive if and only if there exists an arc $(u, v) \in H$ for every directed u–v path in $(u, v) \in H$.

Definition

An acyclic digraph $\overrightarrow{G} = (V, H)$ is **transitive**, *if for every two arcs $(u, v) \in H$, $(v, w) \in H$ there exists arc $(u, w) \in H$.*



In a transitive digraph there is a direct arc (v, w) for every pair of arcs (u, v), (v, w)

Theorem

An acyclic digraph $\overrightarrow{G}$ is transitive if and only if there exists an arc $(u, v) \in H$ for every directed u–v path in $(u, v) \in H$.

Definition

A tranzitive closure $\overrightarrow{G}_T$ *of a digraph* $\overrightarrow{G}$, *is minimal transitive digraph containing as a subgraph digraph* $\overrightarrow{G}$.

A transitive reduction $\overrightarrow{G}_R$ *of a digraph* $\overrightarrow{G}$ *is minimal spanning subgraph of digraph* $\overrightarrow{G}$ *with the same reachebility of vertices as in digraph* $\overrightarrow{G}$.

a) Digraph $\overrightarrow{G}$. b) Transitive closure $\overrightarrow{G}_T$ of digraph $\overrightarrow{G}$.
c) Transitive reduction $\overrightarrow{G}_R$ of digraph $\overrightarrow{G}$.

Definition

A tranzitive closure $\overrightarrow{G}_T$ *of a digraph* $\overrightarrow{G}$*, is minimal transitive digraph containing as a subgraph digraph* $\overrightarrow{G}$*.*

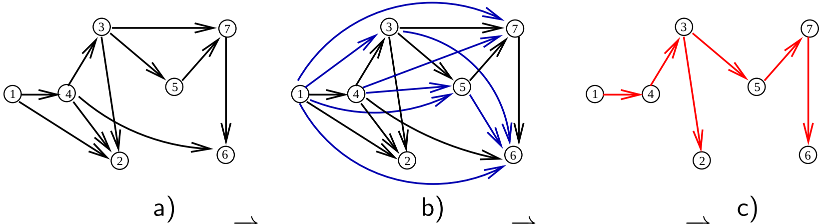
A transitive reduction $\overrightarrow{G}_R$ *of a digraph* $\overrightarrow{G}$ *is minimal spanning subgraph of digraph* $\overrightarrow{G}$ *with the same reachebility of vertices as in digraph* $\overrightarrow{G}$*.*

a) Digraph $\overrightarrow{G}$. b) Transitive closure $\overrightarrow{G}_T$ of digraph $\overrightarrow{G}$.
c) Transitive reduction $\overrightarrow{G}_R$ of digraph $\overrightarrow{G}$.

Definition

A tranzitive closure $\overrightarrow{G}_T$ *of a digraph* $\overrightarrow{G}$, *is minimal transitive digraph containing as a subgraph digraph* $\overrightarrow{G}$.

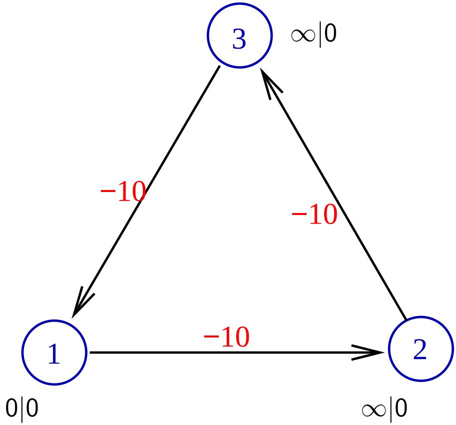
A transitive reduction $\overrightarrow{G}_R$ *of a digraph* $\overrightarrow{G}$ *is minimal spanning subgraph of digraph* $\overrightarrow{G}$ *with the same reachebility of vertices as in digraph* $\overrightarrow{G}$.



a) Digraph $\overrightarrow{G}$. b) Transitive closure $\overrightarrow{G}_T$ of digraph $\overrightarrow{G}$.
c) Transitive reduction $\overrightarrow{G}_R$ of digraph $\overrightarrow{G}$.

Shortest path problem in the general case of arc weights

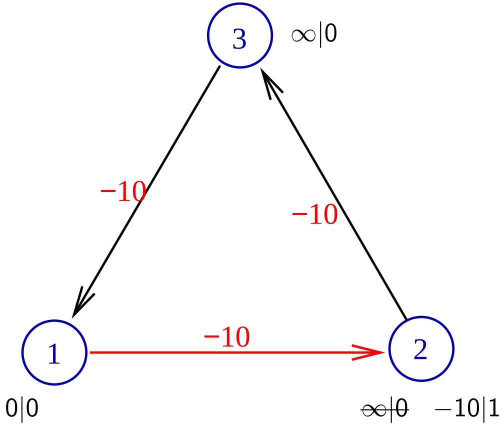
If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path problem in the general case of arc weights

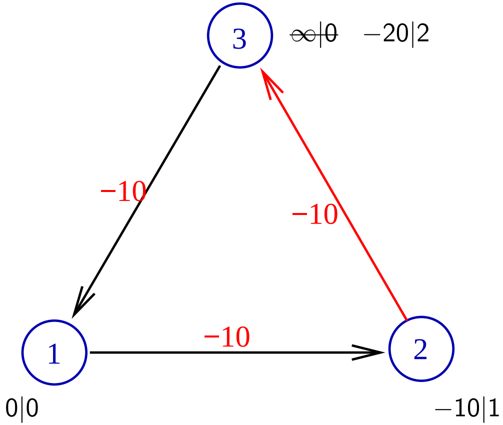
If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path problem in the general case of arc weights

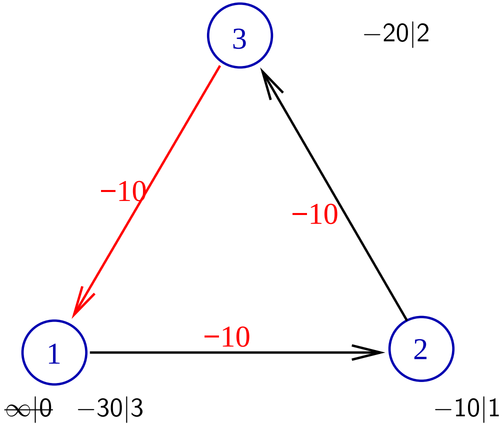
If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path problem in the general case of arc weights

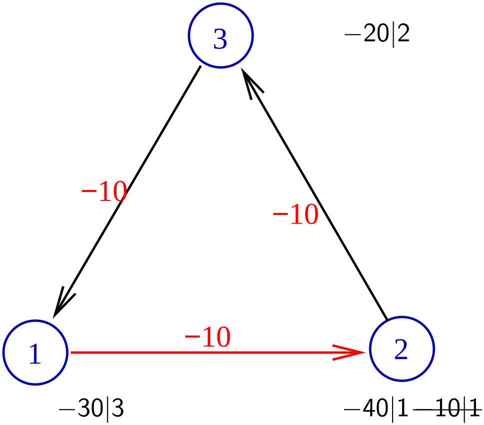
If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path problem in the general case of arc weights

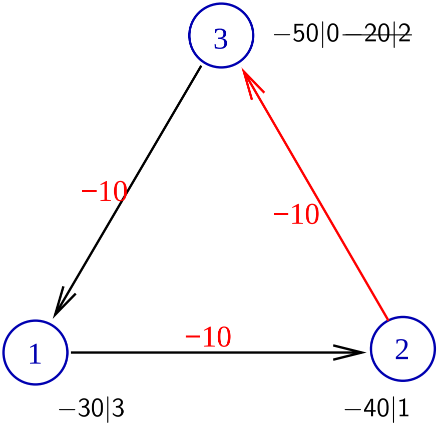
If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path problem in the general case of arc weights

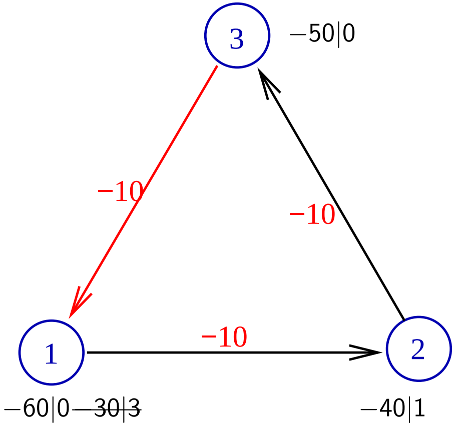
If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path problem in the general case of arc weights

If there exists a negative cycle in a digraph $\overrightarrow{G} = (V, h, c)$ then all shortest path algorithms fail.



However, shortest path problem is polynomialy solvable in acyclic graps even in the case of general arc weights.

Shortest path in an acyclic digraph

Algorithm

Shortest path algorithm for an acyclic digraph. *This algorithm will find all shortest u–v directed paths from a fixed vertex $u \in V$ into all reachable vertices $v \in V$ in an edge weighted digraph $\overrightarrow{G} = (V, H, c)$ with general edge weight $c(h)$.*

- **Step 1.** *Arrange all vertices of digraphu $\overrightarrow{G}$ in topological order into sequence $\mathcal{P} = v_1, v_2, \dots, v_n$.*

  Find index of vertex u in sequence $\mathcal{P}$. Let i be index such that $u = v_i$.

- **Step 2.** *Assign two labels $t(v)$, $x(v)$ for every vertex $v \in V$.*

  Set $t(u) := 0$, $t(j) := \infty$ for all $j \neq u$, $j \in V$.

  Set $x(j) := 0$ for all $j \in V$.

- **Step 3.** *For all veriteces $w \in V^+(v_i)$ do:*
  If $t(w) > t(v_i) + c(v_i, w)$,

  then $t(w) = t(v_i) + c(v_i, w)$, a $x(w) := v_i$.

- **Step 4.** *$i := i + 1$. If $i = n$ STOP, otherwise GOTO Step 3.*

♣

Algorithm

Shortest path algorithm for an acyclic digraph. *This algorithm will find all shortest u–v directed paths from a fixed vertex $u \in V$ into all reachable vertices $v \in V$ in an edge weighted digraph $\overrightarrow{G} = (V, H, c)$ with general edge weight $c(h)$.*

- **Step 1.** *Arrange all vertices of digraphu $\overrightarrow{G}$ in topological order into sequence $\mathcal{P} = v_1, v_2, \dots, v_n$.*

  Find index of vertex u in sequence $\mathcal{P}$. Let i be index such that $u = v_i$.

- **Step 2.** *Assign two labels $t(v)$, $x(v)$ for every vertex $v \in V$.*

  Set $t(u) := 0$, $t(j) := \infty$ for all $j \neq u$, $j \in V$.

  Set $x(j) := 0$ for all $j \in V$.

- **Step 3.** *For all veriteces $w \in V^+(v_i)$ do:*
  If $t(w) > t(v_i) + c(v_i, w)$,

  then $t(w) = t(v_i) + c(v_i, w)$, a $x(w) := v_i$.

- **Step 4.** *$i := i + 1$. If $i = n$ STOP, otherwise GOTO Step 3.*

♣

Shortest path in an acyclic digraph

Algorithm

Shortest path algorithm for an acyclic digraph. *This algorithm will find all shortest u–v directed paths from a fixed vertex $u \in V$ into all reachable vertices $v \in V$ in an edge weighted digraph $\overrightarrow{G} = (V, H, c)$ with general edge weight $c(h)$.*

- **Step 1.** *Arrange all vertices of digraphu $\overrightarrow{G}$ in topological order into sequence $\mathcal{P} = v_1, v_2, \dots, v_n$.*

  Find index of vertex u in sequence $\mathcal{P}$. Let i be index such that $u = v_i$.

- **Step 2.** *Assign two labels $t(v)$, $x(v)$ for every vertex $v \in V$.*

  Set $t(u) := 0$, $t(j) := \infty$ for all $j \neq u$, $j \in V$.

  Set $x(j) := 0$ for all $j \in V$.

- **Step 3.** *For all veriteces $w \in V^+(v_i)$ do:*
  If $t(w) > t(v_i) + c(v_i, w)$,

  then $t(w) = t(v_i) + c(v_i, w)$, a $x(w) := v_i$.

- **Step 4.** *$i := i + 1$. If $i = n$ STOP, otherwise GOTO Step 3.*

♣

Algorithm

Shortest path algorithm for an acyclic digraph. *This algorithm will find all shortest u–v directed paths from a fixed vertex $u \in V$ into all reachable vertices $v \in V$ in an edge weighted digraph $\overrightarrow{G} = (V, H, c)$ with general edge weight $c(h)$.*

- **Step 1.** *Arrange all vertices of digraphu $\overrightarrow{G}$ in topological order into sequence $\mathcal{P} = v_1, v_2, \dots, v_n$.*

  Find index of vertex u in sequence $\mathcal{P}$. Let i be index such that $u = v_i$.

- **Step 2.** *Assign two labels $t(v)$, $x(v)$ for every vertex $v \in V$.*

  Set $t(u) := 0$, $t(j) := \infty$ for all $j \neq u$, $j \in V$.

  Set $x(j) := 0$ for all $j \in V$.

- **Step 3.** *For all veriteces $w \in V^+(v_i)$ do:*
  If $t(w) > t(v_i) + c(v_i, w)$,

  then $t(w) = t(v_i) + c(v_i, w)$, a $x(w) := v_i$.

- **Step 4.** *$i := i + 1$. If $i = n$ STOP, otherwise GOTO Step 3.* ♣

Longest path in a digraph

Definition

Let $\overrightarrow{G} = (V, H, c)$ *be an arc weighted digraph, let* $u \in V$, $v \in V$.
Longest directed u–v **path** *in digraph* $\overrightarrow{G} = (V, H, c)$ *is that directed* u–v *path which has largest length of all directed* u–v *paths.*

Remark

Longest path in an edge weighted graph $G = (V, H, c)$ *can be defined by the same way.*

Longest path in a digraph

Remark

- *Shortest path problem in an arc weighted digraph $\overrightarrow{G} = (V, H, c)$ with nonnegative arc weights (in which $c(h) \geq 0 \ \forall h \in H$) is polynomialy solvable.*
- *Shortest path problem in an arc weighted digraph $\overrightarrow{G} = (V, H, c)$ in which arc weights take general (and also negative) values is in general hard – there is no polynomial algorithm for it.*
- *Longest path problem in a digraph $\overrightarrow{G} = (V, H, c)$ can be transformed to shortest path in digraph $\overrightarrow{G}' = (V, H, \overline{c})$, where $\overline{c}(h) = -c(h)$.*
  This is in general hard problem.

Remark

- Shortest path problem in an arc weighted digraph $\overrightarrow{G} = (V, H, c)$ with nonnegative arc weights (in which $c(h) \geq 0 \ \forall h \in H$) is polynomialy solvable.
- Shortest path problem in an arc weighted digraph $\overrightarrow{G} = (V, H, c)$ in which arc weights take general (and also negative) values is in general hard – there is no polynomial algorithm for it.
- Longest path problem in a digraph $\overrightarrow{G} = (V, H, c)$ can be transformed to shortest path in digraph $\overrightarrow{G}' = (V, H, \overline{c})$, where $\overline{c}(h) = -c(h)$.
  This is in general hard problem.

Remark

- *Shortest path problem in an arc weighted digraph $\overrightarrow{G} = (V, H, c)$ with nonnegative arc weights (in which $c(h) \geq 0 \ \forall h \in H$) is polynomialy solvable.*
- *Shortest path problem in an arc weighted digraph $\overrightarrow{G} = (V, H, c)$ in which arc weights take general (and also negative) values is in general hard – there is no polynomial algorithm for it.*
- *Longest path problem in a digraph $\overrightarrow{G} = (V, H, c)$ can be transformed to shortest path in digraph $\overrightarrow{G}' = (V, H, \overline{c})$, where $\overline{c}(h) = -c(h)$.*
  This is in general hard problem.

Projekt planning methods

- A project is composed from acitivities
- An activity is an elementary amount of work which is from our point of view indivisible.
- Every activity is determined by its fixed processing time which can be different from activity to activity. Activities can differ by processing time.
- Several activities can be performed simlutanously, but execution of some activities can start only after some another acivities are finished.

Definition

We will say that activity A **precedes** *activity B and write $A \prec B$, if activity B can start only after activity A ends.*
If $A \prec B$, we will say tha activity A is **predecessor** *of activity B, or activity B is* **successor** *of activity A.*
$A \prec B$ is a binary relation on the set of all given activities and it will be called **a precedence relation**.

Projekt planning methods

- A project is composed from acitivities
- An activity is an elementary amount of work which is from our point of view indivisible.
- Every activity is determined by its fixed processing time which can be different from activity to activity. Activities can differ by processing time.
- Several activities can be performed simlutanously, but execution of some activities can start only after some another acivities are finished.

Definition

We will say that activity A **precedes** *activity B and write $A \prec B$, if activity B can start only after activity A ends.*
If $A \prec B$, we will say tha activity A is **predecessor** *of activity B, or activity B is* **successor** *of activity A.*
$A \prec B$ is a binary relation on the set of all given activities and it will be called **a precedence relation**.

Projekt planning methods

- A project is composed from acitivities
- An activity is an elementary amount of work which is from our point of view indivisible.
- Every activity is determined by its fixed processing time which can be different from activity to activity. Activities can differ by processing time.
- Several activities can be performed simlutanously, but execution of some activities can start only after some another acivities are finished.

Definition

We will say that activity A **precedes** *activity B and write $A \prec B$, if activity B can start only after activity A ends.*
If $A \prec B$, we will say tha activity A is **predecessor** *of activity B, or activity B is* **successor** *of activity A.*
$A \prec B$ is a binary relation on the set of all given activities and it will be called **a precedence relation**.

Projekt planning methods

- A project is composed from acitivities
- An activity is an elementary amount of work which is from our point of view indivisible.
- Every activity is determined by its fixed processing time which can be different from activity to activity. Activities can differ by processing time.
- Several activities can be performed simlutanously, but execution of some activities can start only after some another acivities are finished.

Definition

We will say that activity A **precedes** *activity B and write $A \prec B$, if activity B can start only after activity A ends.*
If $A \prec B$, we will say tha activity A is **predecessor** *of activity B, or activity B is* **successor** *of activity A.*
$A \prec B$ is a binary relation on the set of all given activities and it will be called **a precedence relation**.

Projekt planning methods

- A project is composed from acitivities
- An activity is an elementary amount of work which is from our point of view indivisible.
- Every activity is determined by its fixed processing time which can be different from activity to activity. Activities can differ by processing time.
- Several activities can be performed simlutanously, but execution of some activities can start only after some another acivities are finished.

Definition

We will say that activity A **precedes** *activity B and write $A \prec B$, if activity B can start only after activity A ends.*
If $A \prec B$, we will say tha activity A is **predecessor** *of activity B, or activity B is* **successor** *of activity A.*
$A \prec B$ is a binary relation on the set of all given activities and it will be called **a precedence relation**.

Remark

Precedence relation $\prec$ je **transitive**, *e. g. it holds:*

$$\text{If } A \prec B,\ B \prec C, \text{ then } A \prec C.$$

If activity B has to wait for the end of activity A and activity C has to wait for the end of activity B, then activity C can not start sooner than activity A ends.

Precedence relation $\prec$ is **antireflexive**, *i. e.:*

$$\text{For no } A \in \mathcal{E} \text{ it holds } A \prec A,$$

otherwise start of activity A should wait for its own end what is thenological nonsence.
Colorary: There does no exist a sequence of activities $A_1, A_2, \ldots, A_n$ such that

$$A_1 \prec A_2 \prec \cdots \prec A_n \prec A_1,$$

otherwise transitivity implies: $A_1 \prec A_1$.

Remark

Precedence relation $\prec$ je **transitive**, *e. g. it holds:*

If $A \prec B$, $B \prec C$, then $A \prec C$.

If activity B has to wait for the end of activity A and activity C has to wait for the end of activity B, then activity C can not start sooner than activity A ends.

Precedence relation $\prec$ is **antireflexive**, *i. e.:*

For no $A \in \mathcal{E}$ it holds $A \prec A$,

otherwise start of activity A should wait for its own end what is thenological nonsence.
Colorary: There does no exist a sequence of activities $A_1, A_2, \ldots, A_n$ such that

$$A_1 \prec A_2 \prec \cdots \prec A_n \prec A_1,$$

otherwise transitivity implies: $A_1 \prec A_1$.

Definition

We will say that activity A **immediately precedes** *activity B and write $A \prec\!\!\prec B$, if $A \prec B$ and there does not exist activity C such that $A \prec C$ and simlutaneously $C \prec B$.*

If $A \prec\!\!\prec B$ we will say also that
activity A is **immediate predecessor** *of activity B,*
or
activity B ise **immediate successor** *of activity A.*

Definition

Project planning problem *$\mathcal{U}$ is given by the set of activites $\mathcal{A}$, precedence relation $\prec$ on the sest $\mathcal{A}$ and by real function $p : \mathcal{A} \to \mathbb{R}$ assigning to every activity $A \in \mathcal{A}$ its processing time $p(A)$.*
(p – processing time).

Immediate precedence. Project planning problem.

Definition

We will say that activity A **immediately precedes** *activity B and write $A \prec\!\!\prec B$, if $A \prec B$ and there does not exist activity C such that $A \prec C$ and simlutaneously $C \prec B$.*

If $A \prec\!\!\prec B$ we will say also that
activity A is **immediate predecessor** *of activity B,*
or
activity B ise **immediate successor** *of activity A.*

Definition

Project planning problem *$\mathcal{U}$ is given by the set of activites $\mathcal{A}$, precedence relation $\prec$ on the sest $\mathcal{A}$ and by real function $p : \mathcal{A} \to \mathbb{R}$ assigning to every activity $A \in \mathcal{A}$ its processing time $p(A)$.*
(p – processing time).

Digraph of precedence

Definition

A digraph of precedence $\prec$ *or* **a precedence digraph** *for corresponding project planning problem* $\mathcal{U}$ *is a vertex weighted digraph*

$$\overrightarrow{\mathbb{G}}_{\prec} = (V, H_{\prec}, p),$$

whose vertex set is the set of all activities, i.e. $V = \mathcal{A}$, *vertex weight* $p(v) > 0$ *represents processing time of vertex – activity* $v \in V$ *and arc set of* $\overrightarrow{\mathbb{G}}_{\prec}$ *is*

$$H_{\prec} = \{(A, B)|\ A, B \in V,\ A \prec B\}.$$

Definition

A digraph of immediate precedence $\prec\!\!\prec$ *for project planning problem* $\mathcal{U}$ *is a vertex weighted digraph*

$$\overrightarrow{\mathbb{G}}_{\prec\!\prec} = (V, H_{\prec\!\prec}, p),$$

with vertex set $V = \mathcal{A}$, *wertex weight* $p(v) > 0$ *representing processing time of* $v \in V$ *and arc set*

$$H_{\prec\!\prec} = \{(A, B)|\ A, B \in V,\ A \prec\!\!\prec B\}.$$

Digraph of precedence

Definition

A digraph of precedence $\prec$ *or* **a precedence digraph** *for corresponding project planning problem* $\mathcal{U}$ *is a vertex weighted digraph*

$$\overrightarrow{\mathbb{G}}_{\prec} = (V, H_{\prec}, p),$$

whose vertex set is the set of all activities, i.e. $V = \mathcal{A}$, *vertex weight* $p(v) > 0$ *represents processing time of vertex – activity* $v \in V$ *and arc set of* $\overrightarrow{\mathbb{G}}_{\prec}$ *is*

$$H_{\prec} = \{(A, B) |\ A, B \in V,\ A \prec B\}.$$

Definition

A digraph of immediate precedence $\prec\!\!\prec$ *for project planning problem* $\mathcal{U}$ *is a vertex weighted digraph*

$$\overrightarrow{\mathbb{G}}_{\prec\!\prec} = (V, H_{\prec\!\prec}, p),$$

with vertex set $V = \mathcal{A}$, *wertex weight* $p(v) > 0$ *representing processing time of* $v \in V$ *and arc set*

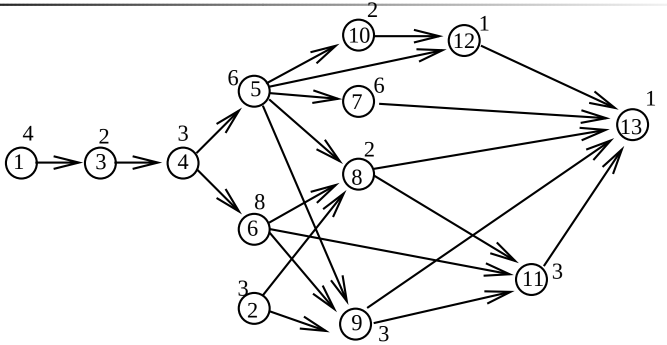
$$H_{\prec\!\prec} = \{(A, B) |\ A, B \in V,\ A \prec\!\!\prec B\}.$$

Technological table of project

Technological table of project

Activity	No	Processing time	Succesor activities
Foundation excavations	1	4	3
Engineer networks	2	3	8 9
Concrete forming of foundations	3	2	4
Concreting of foundations	4	3	5 6
Outer walls	5	6	7 8 9 10 12
Inner partition walls	6	8	9 11
Roof	7	6	13
Electric instalations	8	2	11 13
Wate instalations	9	3	11 13
External rendering	10	2	12
Inner rendering	11	3	13
Windows, doors	12	1	13
Final building approval	13	1	-

Precedence digraph corresponding to technological table



Technological table of project

Activity	No	Processing time	Succesor activities
Foundation excavations	1	4	3
Engineer networks	2	3	8 9
Concrete forming of foundations	3	2	4
Concreting of foundations	4	3	5 6
Outer walls	5	6	7 8 9 10 12
Inner partition walls	6	8	9 11
Roof	7	6	13
Electric instalations	8	2	11 13
Wate instalations	9	3	11 13
External rendering	10	2	12
Inner rendering	11	3	13
Windows, doors	12	1	13
Final building approval	13	1	

Schedule

To create **a schedule** for given project planning problem $\mathcal{U}$ means to assing a time interval $\langle b_A, c_A)$, $b_A < c_A$ for every activity A in which activity A will be processed.

- b_A – beginning time of activity A
- c_A – completion time of actiity A

A feasible schedule of project $\mathcal{U}$ is a schedule for project $\mathcal{U}$, where it holds for arbitrary two activities A, B:

$$
\begin{aligned}
&1. \quad c_A - b_A = p(A) \\
&2. \quad \text{if } A \prec B, \text{ then } b_A < c_A \leq b_B < c_B
\end{aligned}
$$

Remark

Remember that it suffices (based on property 1. of fesible schedule) to determine for every activity A only its beginning b_A. Completion time can be then calculated as $c_A = b_A + p(A)$.

Schedule

To create **a schedule** for given project planning problem $\mathcal{U}$ means to assing a time interval $\langle b_A, c_A)$, $b_A < c_A$ for every activity A in which activity A will be processed.

- b_A – beginning time of activity A
- c_A – completion time of actiity A

A feasible schedule of project $\mathcal{U}$ is a schedule for project $\mathcal{U}$, where it holds for arbitrary two activities A, B:

1. $c_A - b_A = p(A)$
2. if $A \prec B$, then $b_A < c_A \le b_B < c_B$

Remark

Remember that it suffices (based on property 1. of fesible schedule) to determine for every activity A only its beginning b_A. Completion time can be then calculated as $c_A = b_A + p(A)$.

Minimum completion time T

- There is a lot of feasible schedules for given project. However, we are interested in a feasible schedule wich is optimal from certain point of view.
- We take very often $C_{\max}$ – completion time of last activity as objective function.

$$C_{\max} = \max\{c_A \mid A \in \mathcal{A}\},$$

whereas we assume that project starts in time 0.

Value $C_{\max}$ is called **makespan**.

- The goal of our project planning problem is to determine a feasible schedule for the given project $\mathcal{U}$ with minimal makespan $C_{\max}$.
- Denote by T minimum of all completion times of all feasible schedules.

- Set start of project to the time 0.
- **Earliest possible start** $z(A)$ of activity A is the least time moment measured from the beginning of project in which it is possible to start execute activity A whereby precedence relation $\prec$ is kept.
- If earliest possible start is determined for all $A \in \mathcal{A}$ then the minimum completion time T of project can be determined as:

$$T = \max\{z(A) + p(A) \mid A \in \mathcal{E}\}$$

- Suppose that the minimum completion time T of project is determined.

  Latest necessary completion time $k(A)$ of activity A is defined as the largest time moment measured from the beginning of project to which end of execution of activity A can be delayed without encrease of the minimum completion time T.
- **Time reserve** $R(A)$ **of activity** A is:

$$R(A) = k(A) - z(A) - p(A).$$

Earliest possible start, Latest necessary completion time, Time rese

- Set start of project to the time 0.
- **Earliest possible start** $z(A)$ of activity A is the least time moment measured from the beginning of project in which it is possible to start execute activity A whereby precedence relation $\prec$ is kept.
- If earliest possible start is determined for all $A \in \mathcal{A}$ then the minimum completion time T of project can be determined as:

$$T = \max\{z(A) + p(A) \mid A \in \mathcal{E}\}$$

- Suppose that the minimum completion time T of project is determined.

  Latest necessary completion time $k(A)$ of activity A is defined as the largest time moment measured from the beginning of project to which end of execution of activity A can be delayed without encrease of the minimum completion time T.
- **Time reserve** $R(A)$ **of activity** A is:

$$R(A) = k(A) - z(A) - p(A).$$

Earliest possible start, Latest necessary completion time, Time rese

- Set start of project to the time 0.
- **Earliest possible start** $z(A)$ of activity A is the least time moment measured from the beginning of project in which it is possible to start execute activity A whereby precedence relation $\prec$ is kept.
- If earliest possible start is determined for all $A \in \mathcal{A}$ then the minimum completion time T of project can be determined as:

$$T = \max\{z(A) + p(A) \mid A \in \mathcal{E}\}$$

- Suppose that the minimum completion time T of project is determined.

  Latest necessary completion time $k(A)$ of activity A is defined as the largest time moment measured from the beginning of project to which end of execution of activity A can be delayed without encrease of the minimum completion time T.
- **Time reserve** $R(A)$ **of activity** A is:

$$R(A) = k(A) - z(A) - p(A).$$

- Set start of project to the time 0.
- **Earliest possible start** $z(A)$ of activity A is the least time moment measured from the beginning of project in which it is possible to start execute activity A whereby precedence relation $\prec$ is kept.
- If earliest possible start is determined for all $A \in \mathcal{A}$ then the minimum completion time T of project can be determined as:

$$T = \max\{z(A) + p(A) \mid A \in \mathcal{E}\}$$

- Suppose that the minimum completion time T of project is determined.

  Latest necessary completion time $k(A)$ of activity A is defined as the largest time moment measured from the beginning of project to which end of execution of activity A can be delayed without encrease of the minimum completion time T.
- **Time reserve** $R(A)$ **of activity** A is:

$$R(A) = k(A) - z(A) - p(A).$$

Earliest possible start, Latest necessary completion time, Time rese

- Set start of project to the time 0.
- **Earliest possible start** $z(A)$ of activity A is the least time moment measured from the beginning of project in which it is possible to start execute activity A whereby precedence relation $\prec$ is kept.
- If earliest possible start is determined for all $A \in \mathcal{A}$ then the minimum completion time T of project can be determined as:

$$T = \max\{z(A) + p(A) \mid A \in \mathcal{E}\}$$

- Suppose that the minimum completion time T of project is determined.

  Latest necessary completion time $k(A)$ of activity A is defined as the largest time moment measured from the beginning of project to which end of execution of activity A can be delayed without encrease of the minimum completion time T.
- **Time reserve** $R(A)$ **of activity** A is:

$$R(A) = k(A) - z(A) - p(A).$$

- **A critical activity** A is an activity with $R(A) = 0$.
- **Critical path** in digraph $\overrightarrow{\mathbb{G}}_{\prec\!\prec}$ is such directed path which has maximal sum of vertex weights.

Remark

It can be shown that

- *A critical path in $\overrightarrow{\mathbb{G}}_{\prec\!\prec}$ contains only critical activities.*
- *The sum of vertex weights of arbitrary critical path in $\overrightarrow{\mathbb{G}}_{\prec\!\prec}$ is equal to the minimum completion time T of project.*

Critical activities, critical path

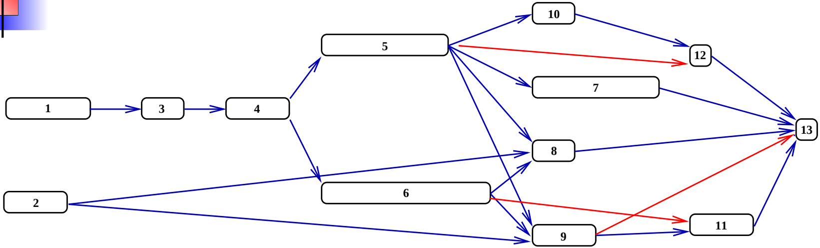
- **A critical activity** A is an activity with $R(A) = 0$.
- **Critical path** in digraph $\overrightarrow{\mathbb{G}}_{\ll}$ is such directed path which has maximal sum of vertex weights.

Remark

It can be shown that

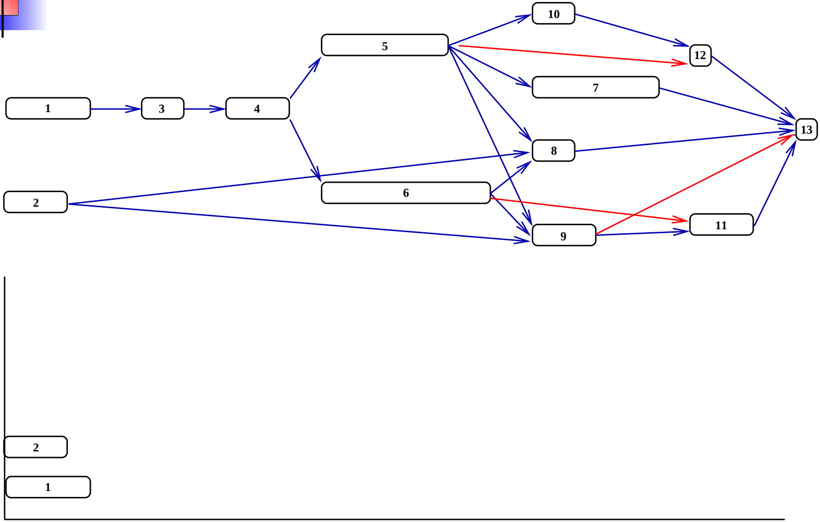
- *A critical path in $\overrightarrow{\mathbb{G}}_{\ll}$ contains only critical activities.*
- *The sum of vertex weights of arbitrary critical path in $\overrightarrow{\mathbb{G}}_{\ll}$ is equal to the minimum completion time T of project.*

Determining erliest possible starts of activities



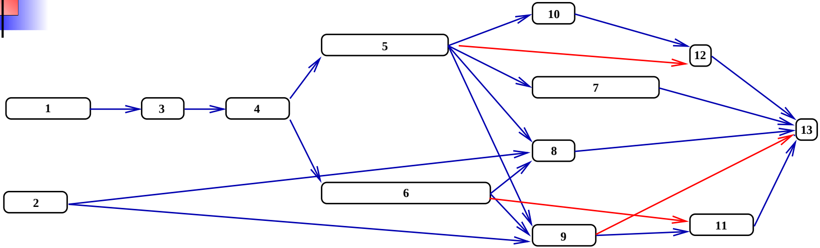
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

Determining erliest possible starts of activities



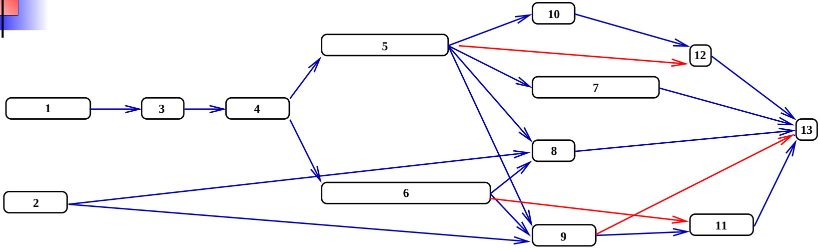
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

Determining erliest possible starts of activities



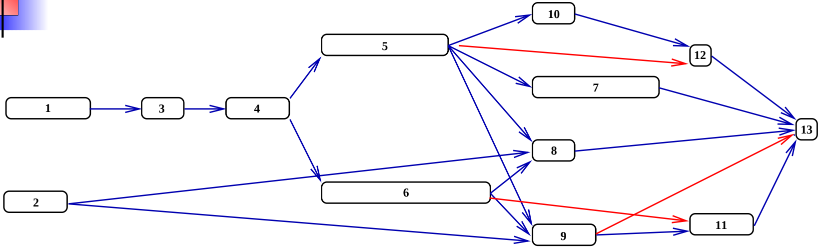
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

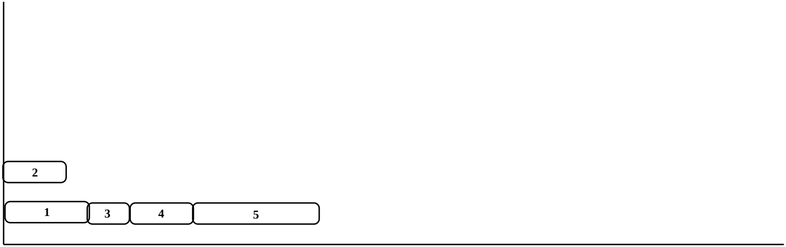
Determining erliest possible starts of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

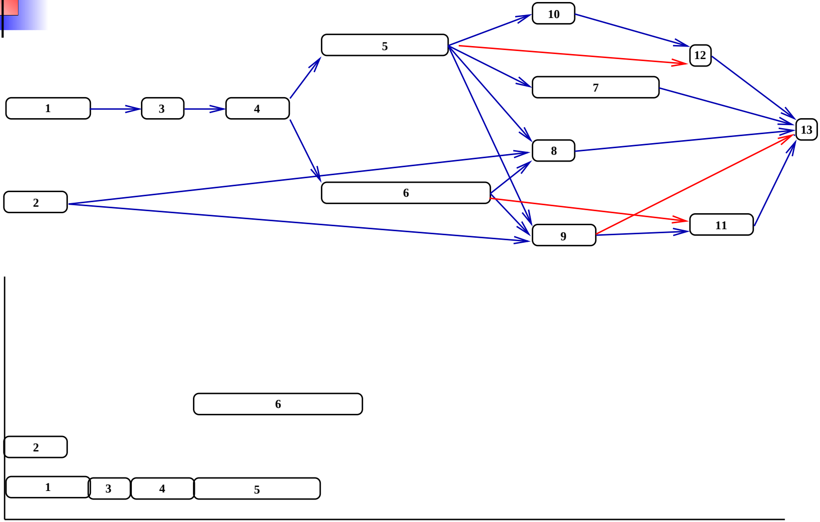
Determining erliest possible starts of activities





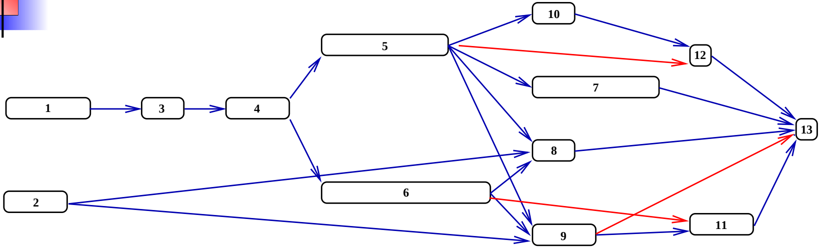
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

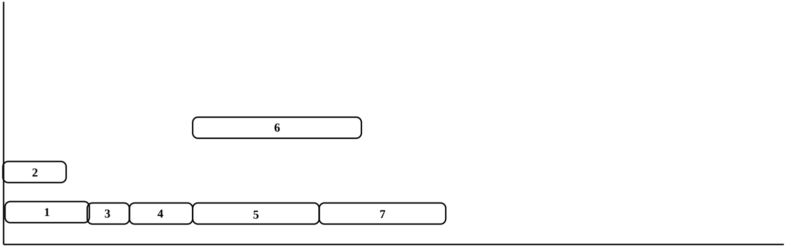
Determining erliest possible starts of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

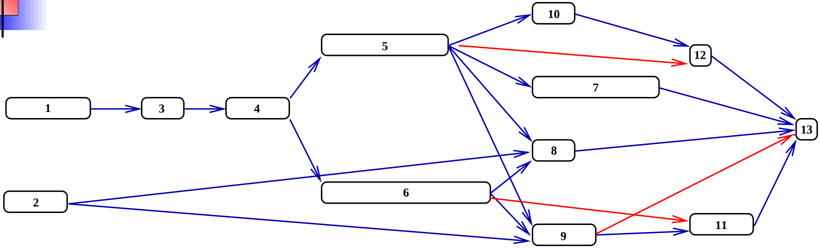
Determining erliest possible starts of activities

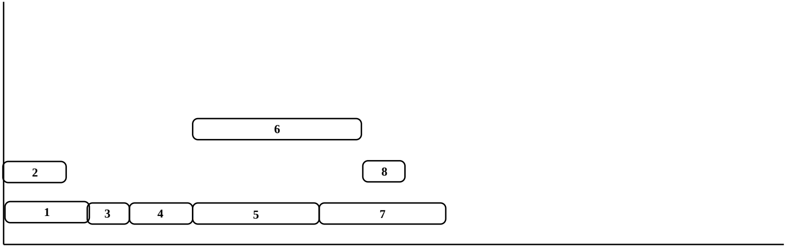




Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

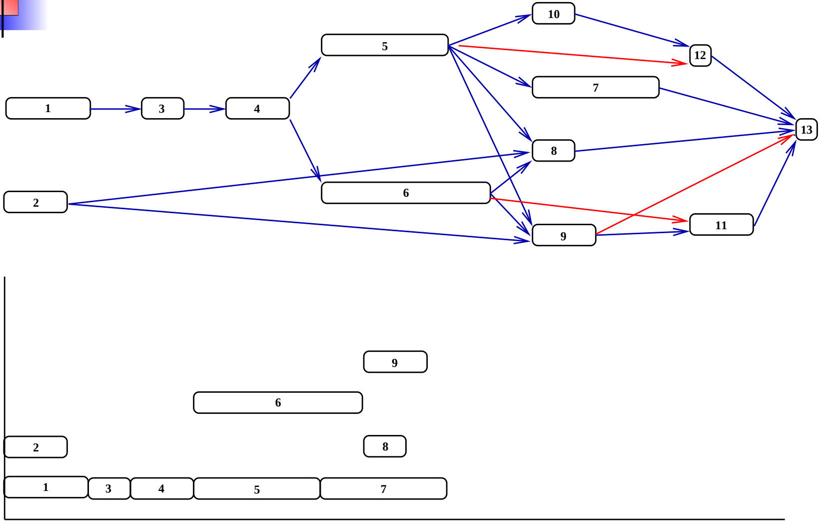
Determining erliest possible starts of activities





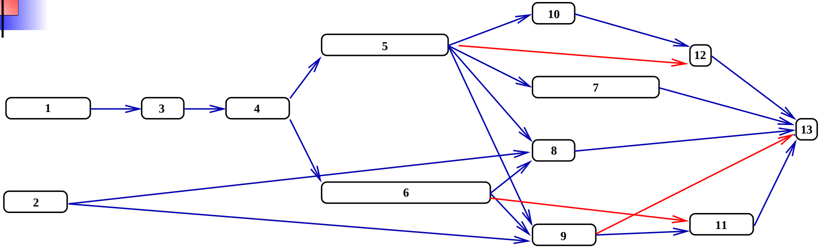
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

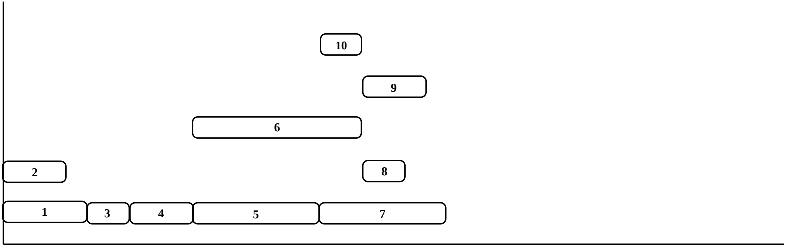
Determining erliest possible starts of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

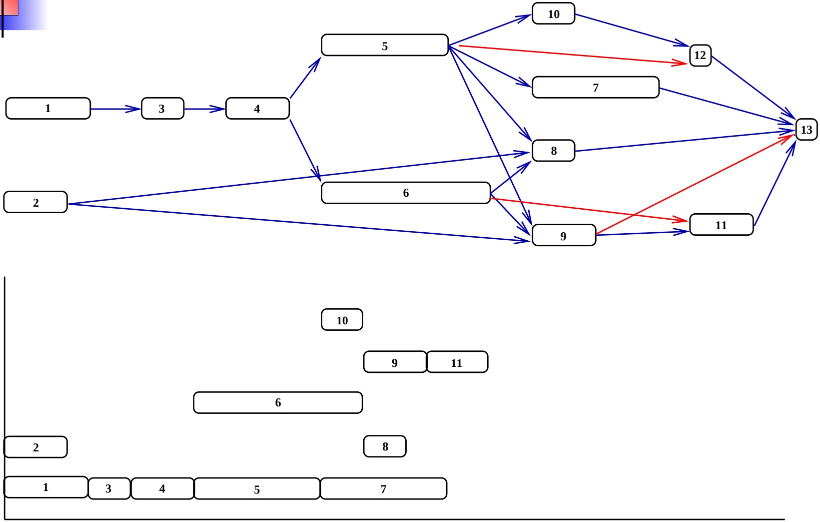
Determining erliest possible starts of activities





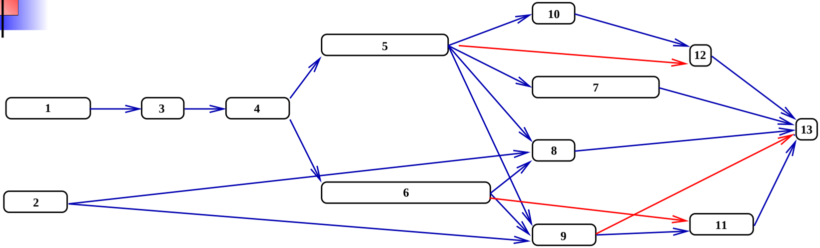
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

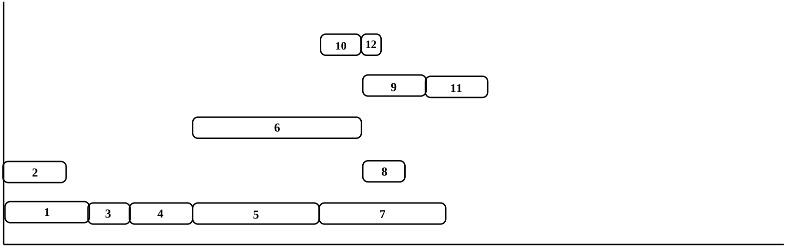
Determining erliest possible starts of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

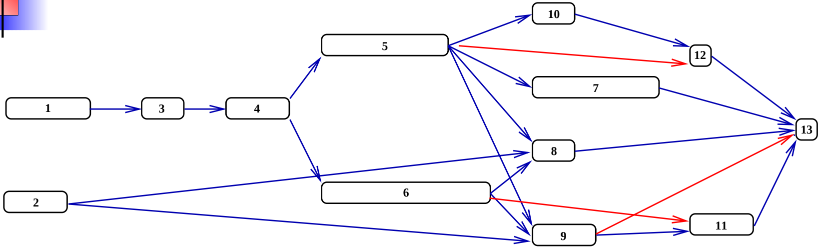
Determining erliest possible starts of activities

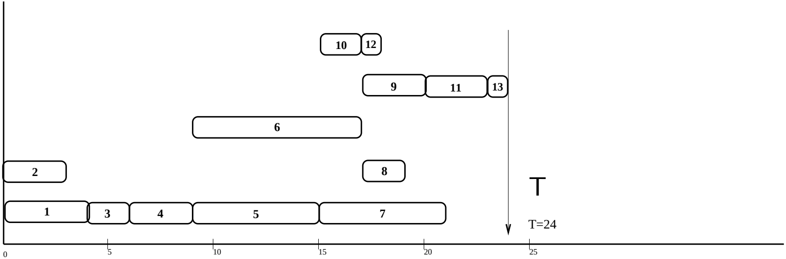




Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

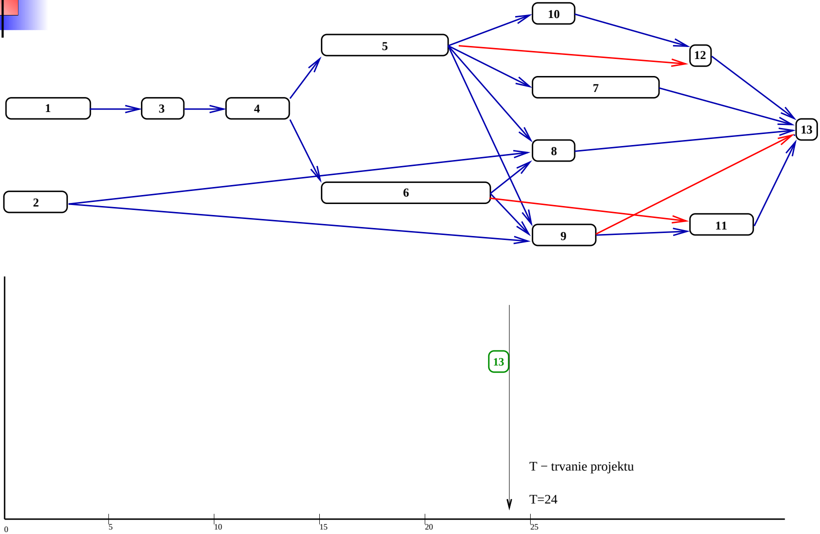
Determining erliest possible starts of activities





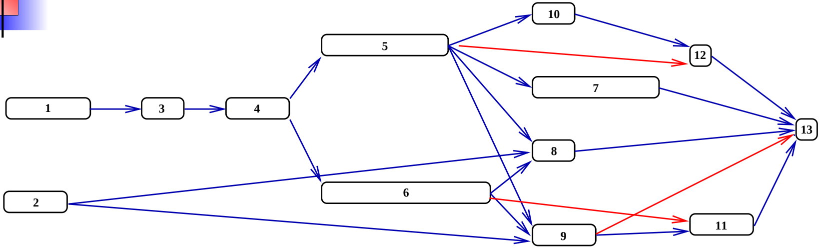
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

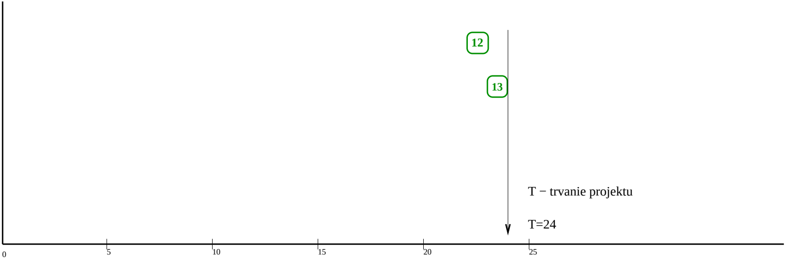
Determining latest necessary positions of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

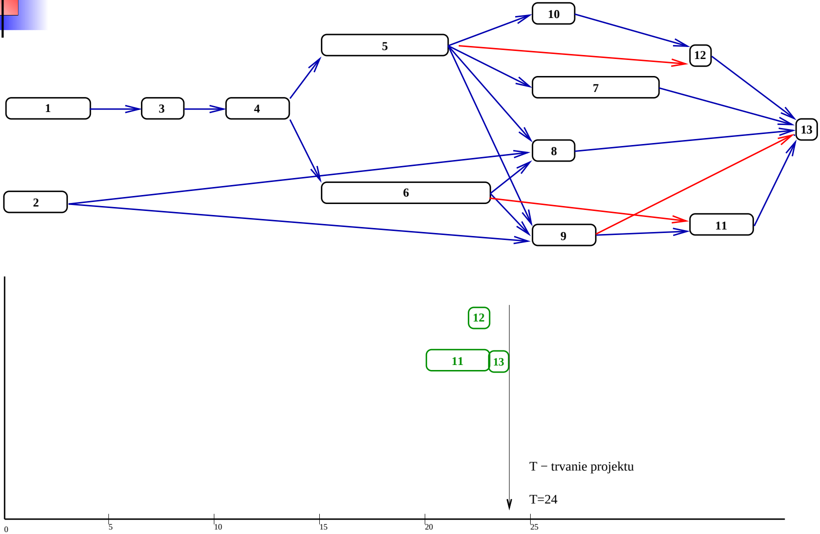
Determining latest necessary positions of activities





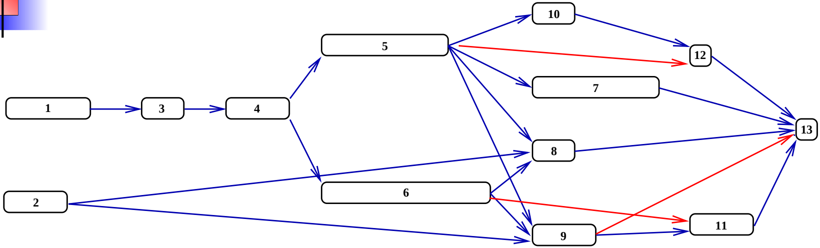
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

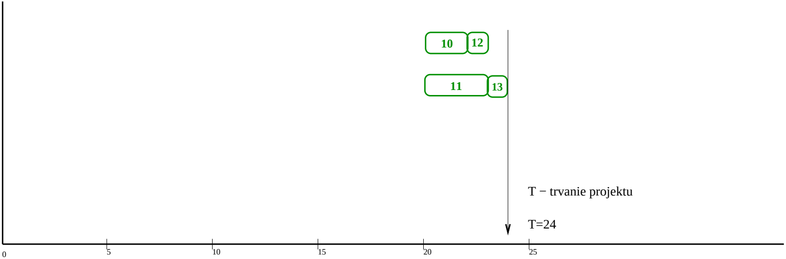
Determining latest necessary positions of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

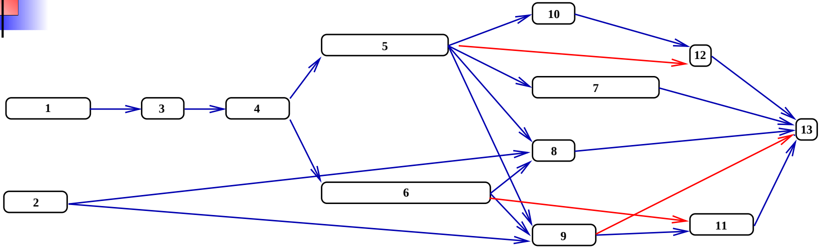
Determining latest necessary positions of activities

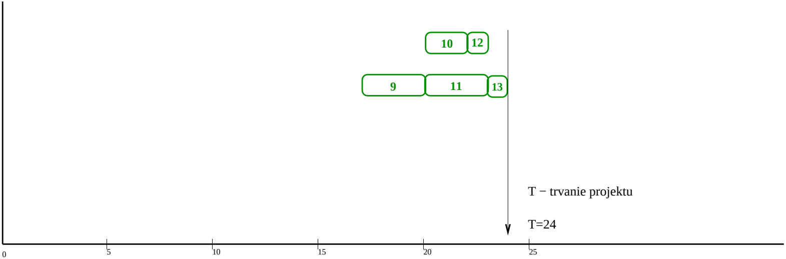




Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

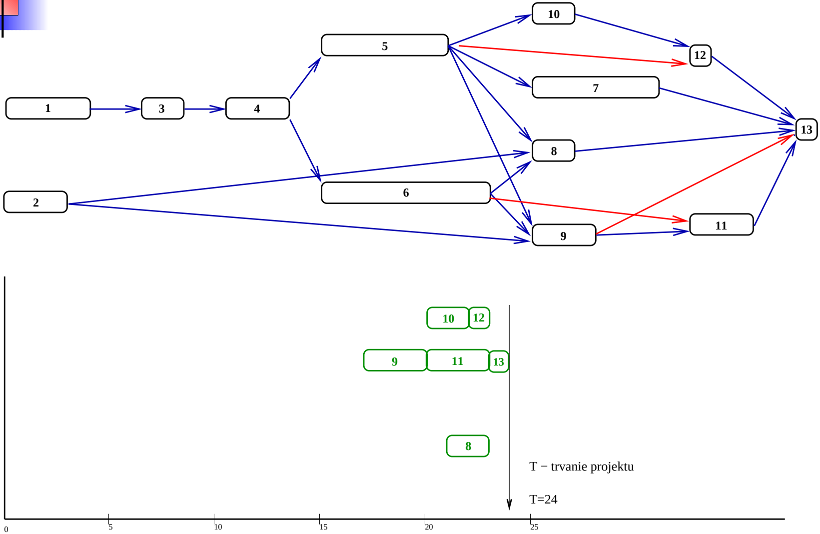
Determining latest necessary positions of activities





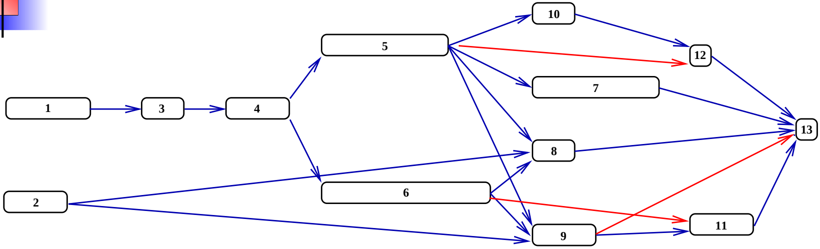
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

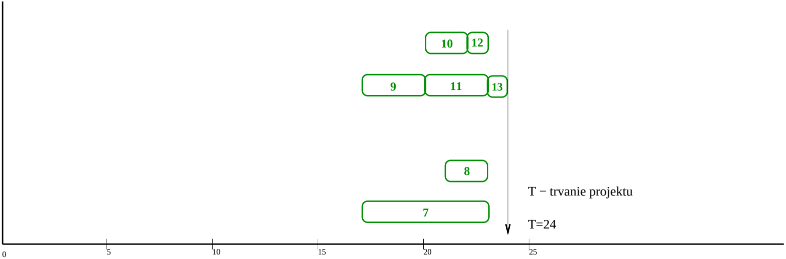
Determining latest necessary positions of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

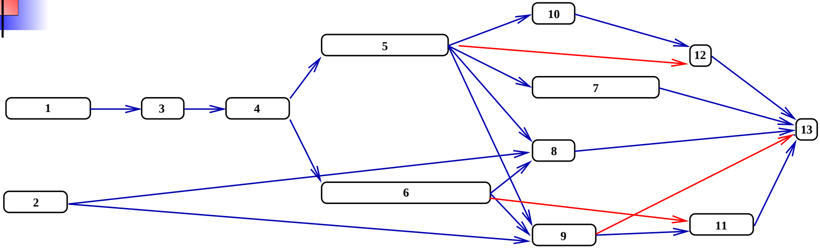
Determining latest necessary positions of activities

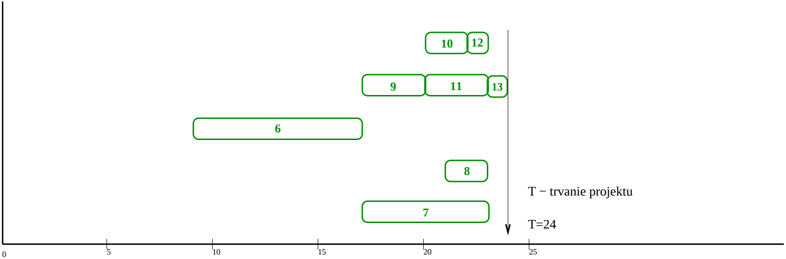




Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

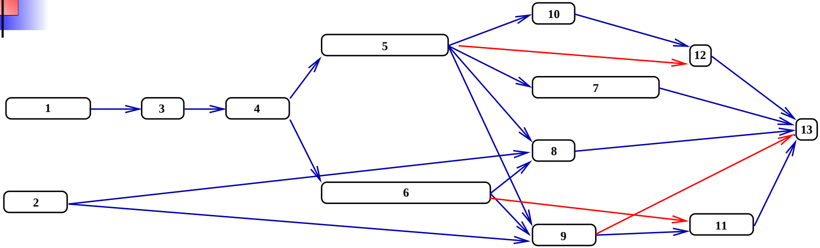
Determining latest necessary positions of activities

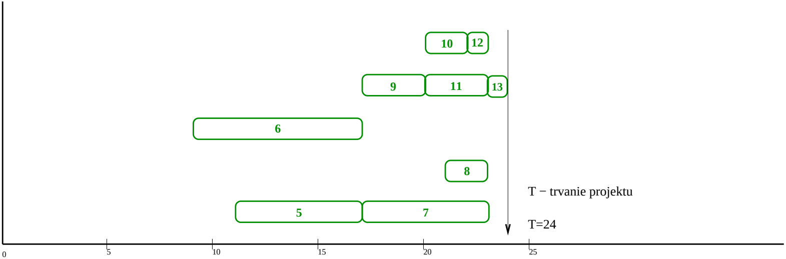




Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

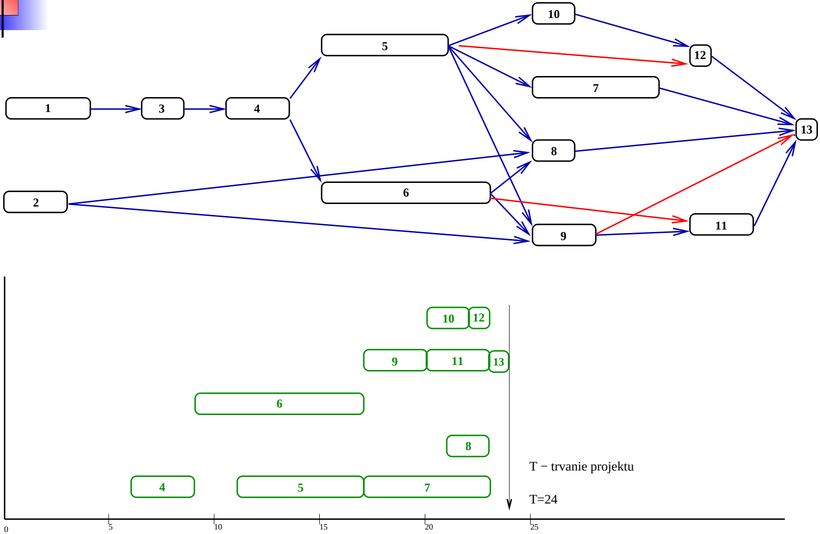
Determining latest necessary positions of activities





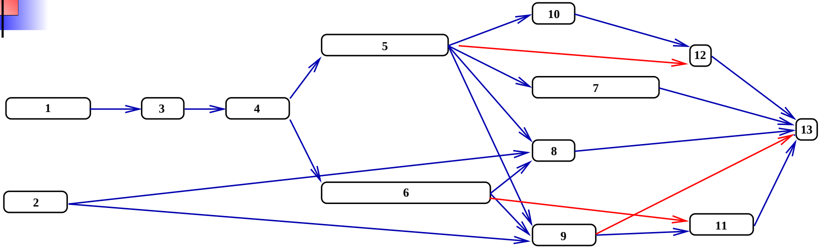
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

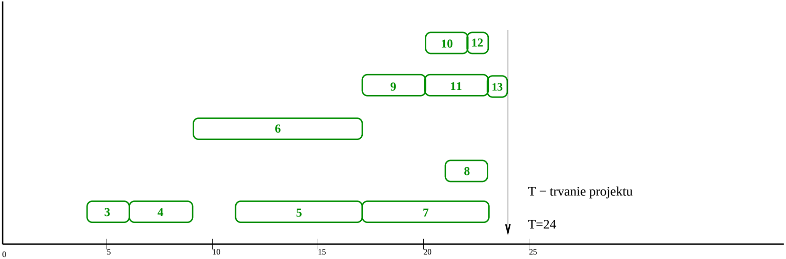
Determining latest necessary positions of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

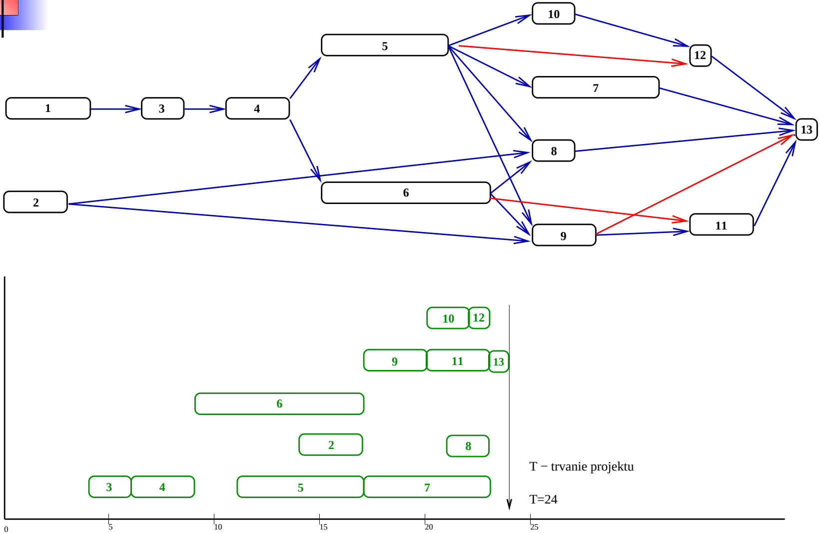
Determining latest necessary positions of activities





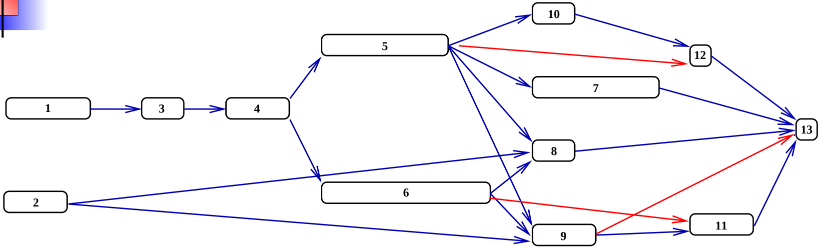
Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

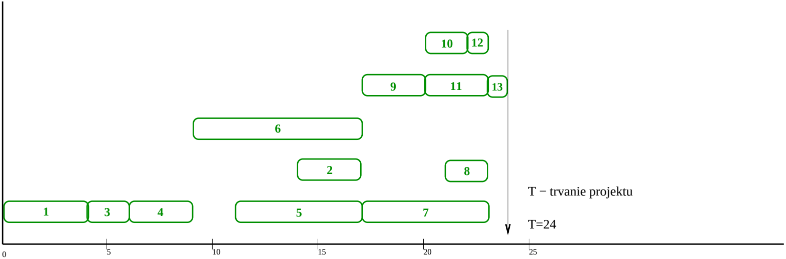
Determining latest necessary positions of activities



Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

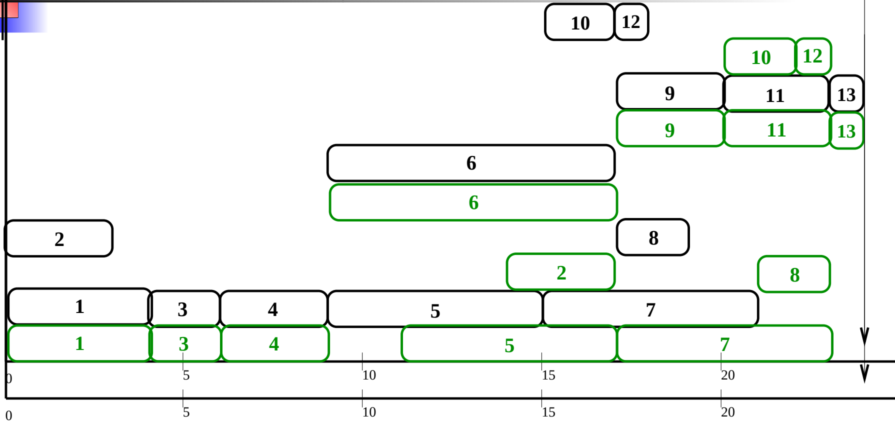
Determining latest necessary positions of activities





Only blue arcs of immediate precedence are necessary for correct project analysis. Red arcs are not arcs of immediate precedence. They have no influence on result, but they can slightly extend computing time.

Comparison



Black – earliest possible positions of activities

Green – latest necessary positions of activities

Algorithm

Algorithm II. to determine earliest beginnings $z(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\prec\!\prec} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\prec\!\prec}$.*
- **Step 2.** *Assign two labels $z(v)$, $x(v)$ to every vertex $v \in V$. Set $x(v) := 0$, $z(v) := 0$ for every $v \in V$.*
- **Step 3.** *For $k = 1, 2, \dots, n-1$ do:*

  For all vertices $w \in V^+(v_k)$ do:
  If $z(w) < z(v_k) + p(v_k)$,
  then $z(w) := z(v_k) + p(v_k)$ and $x(w) := v_k$.
- **Step 4.** *Compute the minimum completion time of project:*

$$T := \max\{z(w) + p(w) \mid w \in V,\ \mathrm{odeg}(w) = 0\}$$

♣

Algorithm

Algorithm II. to determine earliest beginnings $z(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\twoheadleftarrow} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\twoheadleftarrow}$.*
- **Step 2.** *Assign two labels $z(v)$, $x(v)$ to every vertex $v \in V$. Set $x(v) := 0$, $z(v) := 0$ for every $v \in V$.*
- **Step 3.** *For $k = 1, 2, \dots, n-1$ do:*

  For all vertices $w \in V^+(v_k)$ do:
  If $z(w) < z(v_k) + p(v_k)$,
  then $z(w) := z(v_k) + p(v_k)$ and $x(w) := v_k$.
- **Step 4.** *Compute the minimum completion time of project:*

$$T := \max\{z(w) + p(w) \mid w \in V,\ \mathrm{odeg}(w) = 0\}$$

♣

Algorithm

Algorithm II. to determine earliest beginnings $z(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\ll} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\ll}$.*
- **Step 2.** *Assign two labels $z(v)$, $x(v)$ to every vertex $v \in V$.*
  Set $x(v) := 0$, $z(v) := 0$ for every $v \in V$.
- **Step 3.** *For $k = 1, 2, \dots, n-1$ do:*

  For all vertices $w \in V^+(v_k)$ do:
  If $z(w) < z(v_k) + p(v_k)$,
  then $z(w) := z(v_k) + p(v_k)$ and $x(w) := v_k$.
- **Step 4.** *Compute the minimum completion time of project:*

$$T := \max\{z(w) + p(w) \mid w \in V,\ \mathrm{odeg}(w) = 0\}$$

♣

Algorithm

Algorithm II. to determine earliest beginnings $z(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\ll} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\ll}$.*
- **Step 2.** *Assign two labels $z(v)$, $x(v)$ to every vertex $v \in V$.*
  Set $x(v) := 0$, $z(v) := 0$ for every $v \in V$.
- **Step 3.** *For $k = 1, 2, \dots, n-1$ do:*

  For all vertices $w \in V^+(v_k)$ do:
  If $z(w) < z(v_k) + p(v_k)$,
  then $z(w) := z(v_k) + p(v_k)$ and $x(w) := v_k$.
- **Step 4.** *Compute the minimum completion time of project:*

$$T := \max\{z(w) + p(w) \mid w \in V,\ \mathrm{odeg}(w) = 0\}$$

♣

Algorithm

Algorithm II. to determine latest necessary completion times $k(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\lll} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\lll}$.*
- **Step 2.** *Assign two labels $k(v)$, $y(v)$ to every vertex $v \in V$. Let T be the minimum completion time of the project.*
  Set $k(v) := T$, $y(v) := 0$ for every $v \in V$.
- **Step 3.** *For $i = n-1, n-2, \dots, 1$ do:*

  For all vertices $w \in V^+(v_i)$ do:
  If $k(v_i) > k(w) - p(w)$,
  then $k(v_i) := k(w) - p(w)$ a $y(v_i) := w$.

♣

Algorithm

Algorithm II. to determine latest necessary completion times $k(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\ll} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\ll}$.*
- **Step 2.** *Assign two labels $k(v)$, $y(v)$ to every vertex $v \in V$. Let T be the minimum completion time of the project.*
  Set $k(v) := T$, $y(v) := 0$ for every $v \in V$.
- **Step 3.** *For $i = n-1, n-2, \dots, 1$ do:*

  For all vertices $w \in V^+(v_i)$ do:
  If $k(v_i) > k(w) - p(w)$,
  then $k(v_i) := k(w) - p(w)$ a $y(v_i) := w$.

♣

Algorithm

Algorithm II. to determine latest necessary completion times $k(v)$ of activities in digraph $\overrightarrow{\mathbb{G}}_{\lll} = (V, H, p)$.

- **Step 1.** *Create topological ordering $v_1, v_2, \dots, v_n$ of vertex set of digraph $\overrightarrow{\mathbb{G}}_{\lll}$.*
- **Step 2.** *Assign two labels $k(v)$, $y(v)$ to every vertex $v \in V$. Let T be the minimum completion time of the project.*
  Set $k(v) := T$, $y(v) := 0$ for every $v \in V$.
- **Step 3.** *For $i = n-1, n-2, \dots, 1$ do:*

  For all vertices $w \in V^+(v_i)$ do:
  If $k(v_i) > k(w) - p(w)$,
  then $k(v_i) := k(w) - p(w)$ a $y(v_i) := w$.

♣

Computation of earliest possible starts

Forward stars — Table for computation of earliest possible starts of activities

v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars			Table for computation of earliest possible starts of activities															
v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars | Table for computation of earliest possible starts of activities

v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	$z(i)$: 1	2	3	4	5	6	7	8	9	10	11	12	13
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars | Table for computation of earliest possible starts of activities

v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
												$z(i)$						
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars — Table for computation of earliest possible starts of activities

v	$p(v)$	$V^{+}(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
												$z(i)$						
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars — Table for computation of earliest possible starts of activities

v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars			Table for computation of earliest possible starts of activities															
v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars — Table for computation of earliest possible starts of activities

v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars			Table for computation of earliest possible starts of activities															
v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars

v	$p(v)$	$V^+(v)$
1	4	3
2	3	8 9
3	2	4
4	3	5 6
5	6	7 8 9 10 12
6	8	8 9 11
7	6	13
8	2	11 13
9	3	11 13
10	2	12
11	3	13
12	1	13
13	1	-

Table for computation of earliest possible starts of activities

v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
			$z(i)$												
-		-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	4	0			4										
2	3	0								3	3				
3	2	4				6									
4	3	6					9	9							
5	6	9							15	15	15	15		15	
6	8	9								17	17		17		
7	6	15													21
8	2	17											19		
9	3	17											20		
10	2	15												17	
11	3	20													23
12	1	17													
13	1	23													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars			Table for computation of earliest possible starts of activities															
v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars — Table for computation of earliest possible starts of activities

v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
												$z(i)$						
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars			Table for computation of earliest possible starts of activities															
v	$p(v)$	$V^+(v)$	v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$z(i)$												
			-		-	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													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of earliest possible starts

Forward stars

v	$p(v)$	$V^+(v)$
1	4	3
2	3	8 9
3	2	4
4	3	5 6
5	6	7 8 9 10 12
6	8	8 9 11
7	6	13
8	2	11 13
9	3	11 13
10	2	12
11	3	13
12	1	13
13	1	-

Table for computation of earliest possible starts of activities

v	$p(v)$	$z(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
									$z(i)$						
-		-	0	0	0	0	0	0	0	0	0	0	0	0	0
1	4	0			4										
2	3	0								3	3				
3	2	4				6									
4	3	6					9	9							
5	6	9							15	15	15	15		15	
6	8	9								17	17		17		
7	6	15													21
8	2	17											19		
9	3	17											20		
10	2	15												17	
11	3	20													23
12	1	17													
13	1	23													

$$T = \max\{z(v) + p(v) \mid v \in V\} = 24.$$

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v)=\min\{k(i)-p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v)=\min\{k(i)-p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars   Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v)=\min\{k(i)-p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v)=\min\{k(i)-p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v)=\min\{k(i)-p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Computation of latest necessary completion times

Forward stars   Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v) = \min\{k(i) - p(i) \mid i \in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

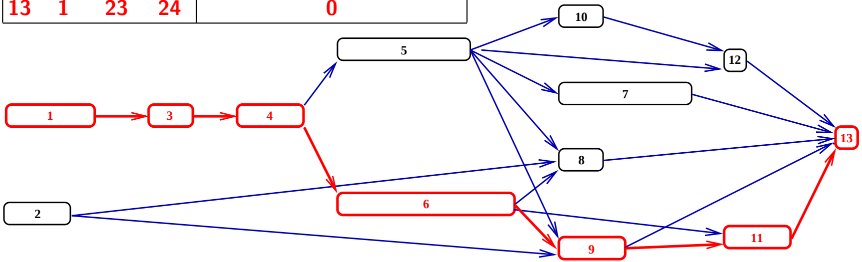
Computation of latest necessary completion times

Forward stars Table for computation of latest necessary completion times of activities

v	$V^+(v)$	v	$p(v)$	$k(v)-p(v)$	$k(v)$	1	2	3	4	5	6	7	8	9	10	11	12	13
						$k(v)=\min\{k(i)-p(i)\mid i\in V^+(v)\}$												
		-		-	-	24	24	24	24	24	24	24	24	24	24	24	24	24
13	-	13	1	23	24													24
12	13	12	1	22	23												23	
11	13	11	3	20	23											23		
10	12	10	2	20	22										22			
9	11 13	9	3	17	20									20				
8	11 13	8	2	18	20								20					
7	13	7	6	17	23							23						
6	8 9 11	6	8	9	17						17							
5	7 8 9 10 12	5	6	11	17					17								
4	5 6	4	3	6	9				9									
3	4	3	2	4	6			6										
2	8 9	2	3	14	17		17											
1	3	1	4	0	4	4												

Critical activities, critical path

v	$p(v)$	$z(v)$	$k(v)$	$R(v) = k(v) - z(v) - p(v)$
1	**4**	**0**	**4**	**0**
2	3	0	17	14
3	**2**	**4**	**6**	**0**
4	**3**	**6**	**9**	**0**
5	6	9	17	2
6	**8**	**9**	**17**	**0**
7	6	15	23	2
8	2	17	20	1
9	**3**	**17**	**20**	**0**
10	2	15	22	3
11	**3**	**20**	**23**	**0**
12	1	17	23	5
13	**1**	**23**	**24**	**0**



Classical interpretation of CPM method

Let $\mathcal{U}$ be a project planning problem given by an activity set $\mathcal{A}$, precedence relation $\prec$ on the set $\mathcal{A}$ and by a real function $c : \mathcal{A} \to \mathbb{R}$ assigning to every activity $A \in \mathcal{A}$ its processing time $p(A)$.

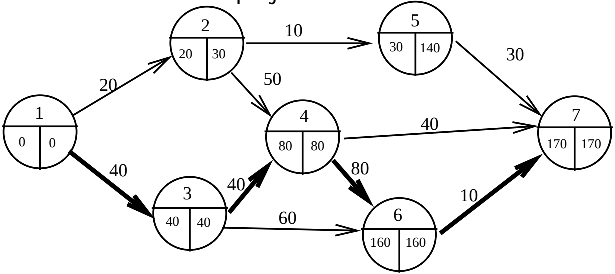
AOA (activity on arc) network is a weakly connected acyclic edge weighted digraph $\overrightarrow{G} = (V, H, p)$ containg exactly one vertex s – start of the project and exactly one vertex f – finish of the project with following properties:
Every vertex v of V is reachable from the start s and
the finish f is reachable from every vertex v of V.

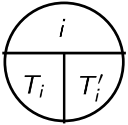
Arcs of AOA network represents activities – for every activity $A \in \mathcal{A}$ is assigned exatly one arc having arc weight equal to processing time $p(A)$ of A.

Vertices represent events of beginnings and completions of activities.
If $A \prec\!\!\prec B$ – i.e. activity A immediately precedes activity B then the arc B has its head identical with the tail of the arc A.

Classical interpretation of CPM method

Suppose that $V = \{1, 2, \ldots n\}$ and that $s = 1$ – start ot hte project, $f = n$ – finish of the project.





T_i – Earliest possible beginning time of activities outgoing from vertex i

T'_i – Latest necessary completion time of activities incommning into vertex i

Diagram of an AOA network as can be found in many textbooks.

However, author(s) tacticaly keep silent about how to construct it for a technological table without so called dummy activites with zero processing time

Classical interpretation of CPM method

Denote by $d_{\max}(x, y)$ the length of the longest x–y directed path in AOA network $\overrightarrow{G}$. Remember that vertex 1 is the start and vertex n is the finish of corresponding project.

The earliest possible time T_i of activities outgoing from every vertex $i \in V$ is calculated as

$$T_i = d_{\max}(1, i)$$

The latest possible completion time T'_i of activities incomming into vertex $i \in V$ is calculated as

$$T'_i = T_n - d_{\max}(i, n)$$

The minimum completion time T of the project is

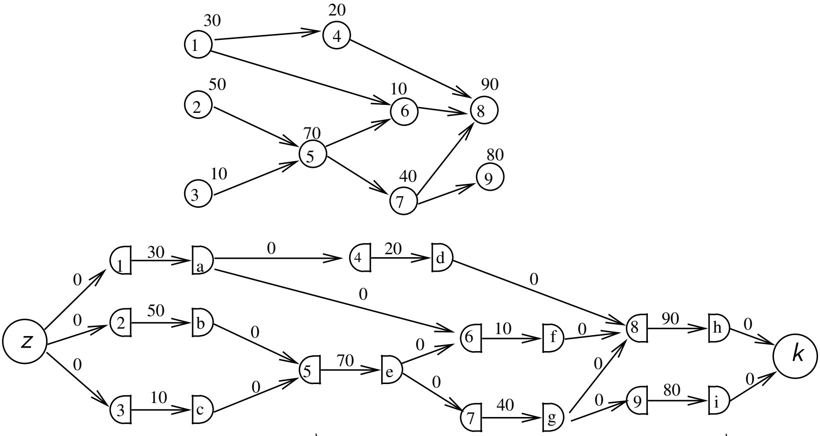
$$T = T_n.$$

Every directed 1 - n path having the length equal to T is called **a critical path**. (There can exist mor ctitical paths).

Activities belonging to a critical path are called **critical activities**.

Time reserve R_i **ina vertex** i is $R_i = T'_i - T_i$.

Construction of AOA network



Construction of AOA network $\overrightarrow{G}_S$ (below) from precedence digrap $\overrightarrow{\mathbb{G}}_{\ll}$ (above).

Construction of AOA network

1. Create digraph of immediate precedence $\overrightarrow{\mathbb{G}}_{\lll}$.
2. Declare all arcs of $\overrightarrow{\mathbb{G}}_{\lll}$ as dummy arcs with processing times equal to 0.
3. Add two vertices z and k.
4. Add arcs of the type (z, v) for all vertices v such that $\mathrm{ideg}(v) = 0$. Consider these arcs as dummy arcs with processing times equal to 0.
5. Add arcs of the type (v, k) for all vertices v such that $\mathrm{odeg}(v) = 0$.These arcs consider as dummy arcs with processing times equal to 0.
6. Split every vertex representing an activity into input and output part. Add an arc with head equal to input part and tail equal to output part of every splitted vertex and set weight of this arc equal to processing time of corresponding activity.