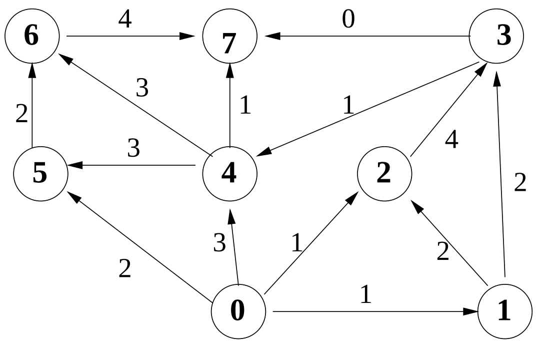
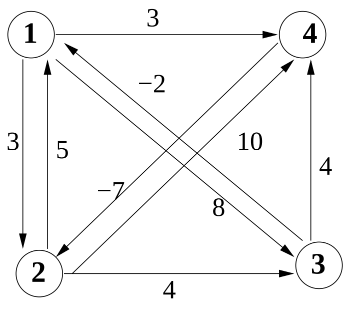
2.5 Shortest paths with nonnegative costs

Given the following directed graph, find a set of shortest paths from node 0 to all the other nodes, using Dijkstra's algorithm. Can we solve the problem with Dynamic Programming? If yes, do so.



2.6 Shortest paths with negative costs and ill-posedness

Given the following directed graph, find the shortest paths between all pairs of nodes, or show that the problem is ill-posed by exhibiting a circuit of total negative cost.



2.7 An application of Dynamic Programming to machine renewal

A company must buy a new machine and then determine a renewal (maintenance-replacement) plan for the next 5 years, making sure that, at any point in time, the available machine works properly. At the beginning of each year of the planning horizon, the company must decide whether to keep the old machine or to substitute it with a new machine.

The maintenance costs and the expected revenue (when the machine is sold) clearly depend on how old the machine is and they are indicated in the following table.

years	maintenance (kEuro)	revenue when sold (kEuro)
0	2	-
1	4	7
2	5	6
3	9	2
4	12	1

To avoid high maintenance costs of an old machine, the machine can be sold at the beginning of the second, third, fourth, and fifth year, and an new one can be bought. For the sake of simplicity, we suppose that a new machine always costs 12K Euro.

Show that the problem of determing a machine renewal plan of minimum total net cost (total cost for buying/rebuying the machine + maintenance costs - total revenue) can be solved via Dynamic Programming by finding a shortest path in an ad hoc directed acyclic graph. Find an optimal renewal plan. Is it unique?

2.8 Project planning

The preparation of the apple pie has long been a tradition at Rossi's family. First the weight of the ingredients has to be determined: flour, sugar, butter, eggs, apples, cream. The butter must then be melted down, and added to a mixture of flour, sugar, and eggs. Apples must be added to this new mixture, once they have been peeled and cut into thin slices. The mixture can then be cooked, in the already heated oven. It is advisable to whip the cream only after the apple slices have been added to the mixture. Once the cake is cooked, the cream is used to garnish it.

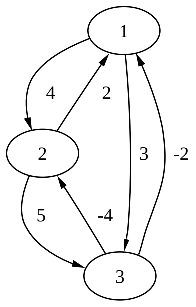
The following table reports the time needed for each activity.

Activity		Time (minutes)
A	Weight the ingredients	5
B	Melt the butter	3
C	Mix flour, eggs and sugar	5
D	Peel the apples and cut them into slices	10
E	Heat the oven	20
F	Add butter to the mixture	8
G	Add apples to the mixture	4
H	Cook the mixture in the oven	40
I	Whip the cream	10
L	Garnish	5

Draw the graph (with activities associated to arcs) which represents the project precedence relations. Determine the minimum total completion time of the project as well as the earliest times and latest times associated to each node. Identify the critical activities and draw Gantt's chart at earliest.

2.9 Shortest paths with negative costs

Given the following directed graph, find the shortest paths between all pairs of nodes, or show that the problem is ill-posed, by exhibiting a circuit of total negative cost.


1
2
3
4
2
3
-2
5
-4

SOLUTION

2.5 Shortest paths with nonnegative costs

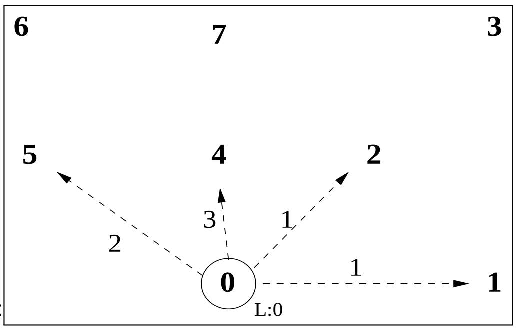
Dijkstra's algorithm determines a set of shortest paths from a given node s to all the other nodes in the graph. It can be applied to any (directed) graph with nonnegative arc costs.

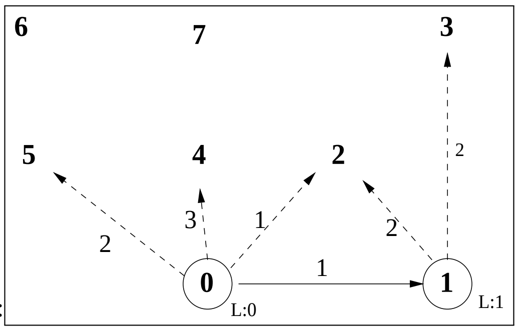
- *Data structures*: subset $S \subseteq V$ of nodes with fixed labels (initialization $S = \{s\}$); n-dimensional vector L where $L(i)$ is the cost of a shortest path from s to i (initialization $L(s) = 0$); n-dimensional vector p where $p(i)$ is the predecessor of node i in the shortest path from s to i (initialization $p(s) = s$).
- *Description*: at each iteration, we identify in the outgoing cut induced by S, $\delta^+(S)$, an arc (v, h) such that $L(v) + c_{vh}$ is minimum, where c_{vh} is the cost of the arc (v, h). Then we set $S := S \cup \{h\}$, $L(h) := L(v) + c_{vh}$ and $p(h) := v$. The algorithm halts when $S = V$.

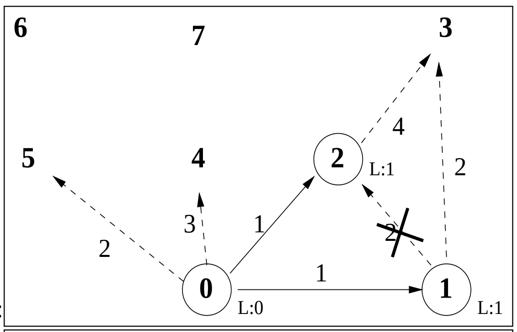
Iterations:

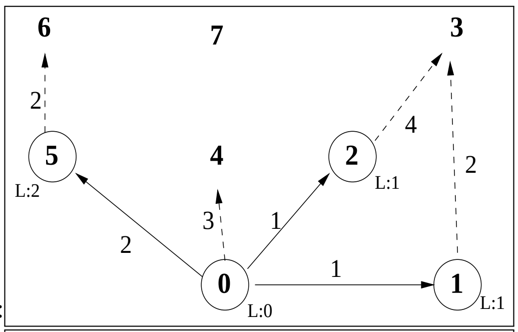
- Initialization: $S = \{0\}, L(0) = 0, p(0) = 0$;
- $(v, h) = (0, 1)$, $L(1) = 1$, $p(1) = 0$, $S = \{0, 1\}$;
- $(v, h) = (0, 2)$, $L(2) = 1$, $p(2) = 0$, $S = \{0, 1, 2\}$;
- $(v, h) = (0, 5)$, $L(5) = 2$, $p(5) = 0$, $S = \{0, 1, 2, 5\}$;
- $(v, h) = (0, 4)$, $L(4) = 3$, $p(4) = 0$, $S = \{0, 1, 2, 4, 5\}$;
- $(v, h) = (1, 3)$, $L(3) = 3$, $p(3) = 1$, $S = \{0, 1, 2, 3, 4, 5\}$;
- $(v, h) = (3, 7)$, $L(7) = 3$, $p(7) = 3$, $S = \{0, 1, 2, 3, 4, 5, 7\}$;
- $(v, h) = (5, 6)$, $L(6) = 4$, $p(6) = 5$, $S = G$: STOP;

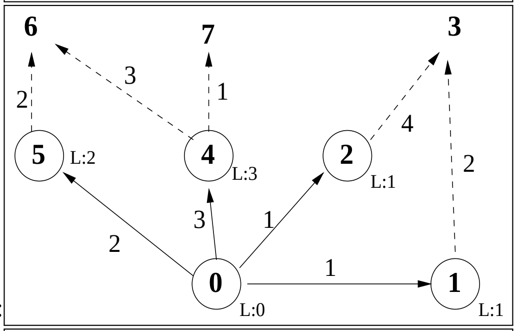
The behaviour of the algorithm can be summarized as follows. In the pictures, a node is highlighted if it belongs to S, while an arc is represented with a dashed line if it can be selected in the current iteration (if it belongs to $\delta^+(S)$), and in a solid line when it is chosen. The labels on the nodes in S indicate the cost of the shortest path from node 0. When a dashed arc is incident to two nodes in S, it is removed. This way, the dashed arcs always belongs to the cut induced by S. The arc (i, j) that is selected is always that with minimum cost $L(i) + c_{ij}$.

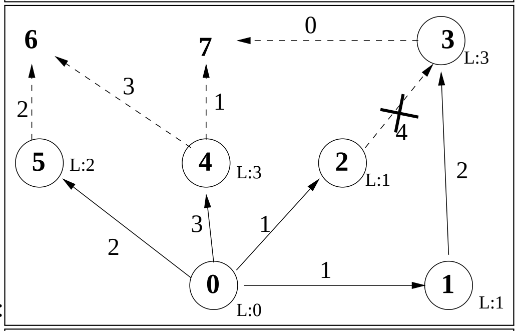
1:


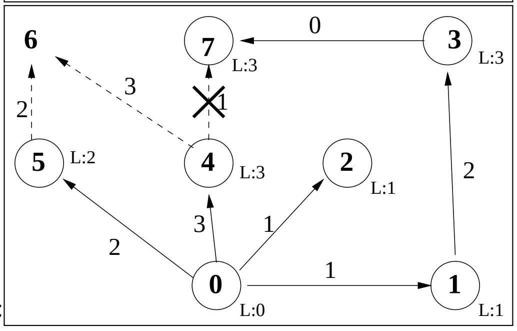
2:


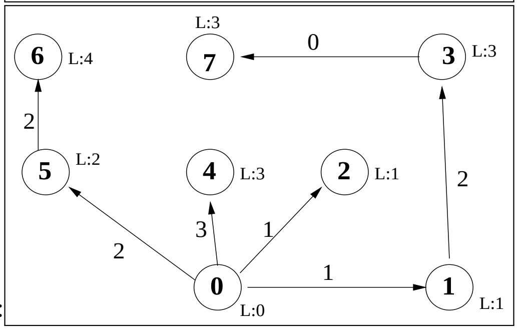
3: 

4: 

5: 

6: 

7: 

8: 

A topological order can be obtained as follows. At iteration i, for $i = \{1, \ldots, n\}$, pick a node with no incoming arcs, label it as 'node i', remove it from the graph, iterate until all nodes are removed, or there is no node with no incident arcs –in this case the graph is not acyclic.

The graph we are dealing with is acyclic, and its node indices already correspond to a topological order. The Dynamic Programming technique can therefore be applied. It has complexity $O(m)$, smaller than that of the Dijkstra algorithm of $O(n^2)$.

- $L(0) = 0$, $p(0) = 0$;
- $L(1) = L(0) + c_{01} = 1$, $p(1) = 0$;
- $L(2) = \min\{L(0) + c_{02}, L(1) + c_{12}\} = \min\{0 + 1, 1 + 2\} = 1$, $p(2) = 0$;
- $L(3) = \min\{L(1) + c_{13}, L(2) + c_{23}\} = \min\{1 + 2, 1 + 4\} = 3$, $p(3) = 1$;
- $L(4) = \min\{L(0) + c_{04}, L(3) + c_{34}\} = \min\{0 + 3, 3 + 1\} = 3$, $p(4) = 0$;
- $L(5) = \min\{L(0) + c_{05}, L(4) + c_{45}\} = \min\{0 + 2, 3 + 3\} = 3$, $p(5) = 0$;
- $L(6) = \min\{L(4) + c_{46}, L(5) + c_{56}\} = \min\{3 + 3, 2 + 2\} = 3$, $p(6) = 5$;
- $L(7) = \min\{L(3)+c_{37}, L(4)+c_{47}, L(6)+c_{67}\} = \min\{3+0, 3+1, 4+4\} = 3$, $p(7) = 3$;

2.6 Shortest paths with negative costs and ill-posedness

Since the graph contains negative arc costs, we use Floyd-Warshall's algorithm that finds a shortest path between each pair of nodes, or establishes that the problem is ill-posed by exhibiting a circuit of negative cost. The graph can contain cycles, but such cycles have to be of nonnegative cost. It has complexity $O(n^3)$, where n is the number of nodes in the graph.

- *Data structures*: let D be an $n \times n$ matrix whose element d_{ij} is, at the beginning of the algorithm, the cost c_{ij} of arc (i, j) and, at the end, the cost of a shortest path from i to j. Let P be an $n \times n$ matrix whose element p_{ij} is the precedecessor of node j in the shortest path from i to j.
- *Description*: We consider in turn each node of the graph according to the order of the indices. At iteration h, we consider the node with index h and, for every pair of nodes (i, j) with $i \neq h$ and $j \neq h$ (but including the case $i = j$), we compare the length of the path via h, that is, $d_{ih} + d_{hj}$, with the current d_{ij}. If $d_{ih} + d_{hj} < d_{ij}$, the matrices D and P are updated: we set $d_{ij} = d_{ih} + d_{hj}$ and p_{ij} is updated to p_{hj}. The algorithm halts when one of the following conditions holds
  (1) all nodes have been selected for the triangulation operation,
  (2) a circuit of negative cost has been found: the problem of finding a shortest path (not necessarily *simple*!) it not well defined. Indeed, it is possible to go through the negative cost circuit an arbitrary number of times, reducing the cost of the path containing such circuit at each iteration. The problem is, evidently, unbounded.

(a) Initial configuration

D	1	2	3	4
1	0	3	8	3
2	5	0	4	10
3	-2	∞	0	4
4	∞	-7	∞	0

P	1	2	3	4
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

(b) Iteration $h = 1$ (triangulation on node 1)

$$\begin{aligned}
d_{21} + d_{12} = 8 &\quad > \quad d_{22} = 0 \\
d_{21} + d_{13} = 13 &\quad > \quad d_{23} = 4 \\
d_{21} + d_{14} = 8 &\quad < \quad d_{24} = 10 \\
&\quad \Rightarrow \quad \textbf{update } d_{24}, p_{24} \\
d_{31} + d_{12} = 1 &\quad < \quad d_{32} = \infty \\
&\quad \Rightarrow \quad \textbf{update } d_{32}, p_{32} \\
d_{31} + d_{13} = 6 &\quad > \quad d_{33} = 0 \\
d_{31} + d_{14} = 1 &\quad < \quad d_{34} = 4 \\
&\quad \Rightarrow \quad \textbf{update } d_{34}, p_{34} \\
d_{41} + d_{ij} = \infty &\quad (\forall i, j)
\end{aligned}$$

D	1	2	3	4
1	0	3	8	3
2	5	0	4	**8**
3	-2	**1**	0	**1**
4	∞	-7	∞	0

P	1	2	3	4
1	1	1	1	1
2	2	2	2	**1**
3	3	**1**	3	**1**
4	4	4	4	4

(c) Iteration $h = 2$ (triangulation on node 2)

$$\begin{aligned}
d_{12} + d_{21} = 8 &> d_{11} = 0 \\
d_{12} + d_{23} = 7 &< d_{13} = 8 \\
&\Rightarrow \textbf{update } d_{13}, p_{13} \\
d_{12} + d_{24} = 11 &> d_{24} = 3 \\
d_{32} + d_{21} = 6 &> d_{31} = -2 \\
d_{32} + d_{23} = 5 &> d_{33} = 0 \\
d_{32} + d_{24} = 9 &> d_{34} = 1 \\
d_{42} + d_{21} = -2 &< d_{41} = \infty \\
&\Rightarrow \textbf{update } d_{41}, p_{41} \\
d_{42} + d_{23} = -3 &< d_{43} = \infty \\
&\Rightarrow \textbf{update } d_{43}, p_{43} \\
d_{42} + d_{24} = 1 &> d_{44} = 0
\end{aligned}$$

D	1	2	3	4
1	0	3	**7**	3
2	5	0	4	8
3	-2	1	0	1
4	**-2**	-7	**-3**	0

P	1	2	3	4
1	1	1	**2**	1
2	2	2	2	1
3	3	1	3	1
4	**2**	4	**2**	4

(d) Iteration $h = 3$ (triangulation on node 3)

$$\begin{aligned}
d_{13} + d_{31} &= 5 > d_{11} = 0 \\
d_{13} + d_{32} &= 8 > d_{12} = 3 \\
d_{13} + d_{34} &= 8 > d_{14} = 3 \\
d_{23} + d_{31} &= 2 < d_{21} = 5 \\
&\Rightarrow \textbf{update } d_{21}, p_{21} \\
d_{23} + d_{32} &= 5 > d_{22} = 0 \\
d_{23} + d_{34} &= 5 < d_{24} = 8 \\
&\Rightarrow \textbf{update } d_{24}, p_{24} \\
d_{43} + d_{31} &= -5 < d_{41} = -2 \\
&\Rightarrow \textbf{update } d_{41}, p_{41} \\
d_{43} + d_{32} &= -2 > d_{42} = -7 \\
d_{43} + d_{34} &= -2 < d_{44} = 0 \\
&\Rightarrow \textbf{update } d_{44}, p_{44}
\end{aligned}$$

D	1	2	3	4
1	0	3	7	3
2	**2**	0	4	**5**
3	-2	1	0	1
4	**-5**	-7	-3	**-2**

P	1	2	3	4
1	1	1	2	1
2	**3**	2	2	**1**
3	3	1	3	1
4	**3**	4	2	**1**

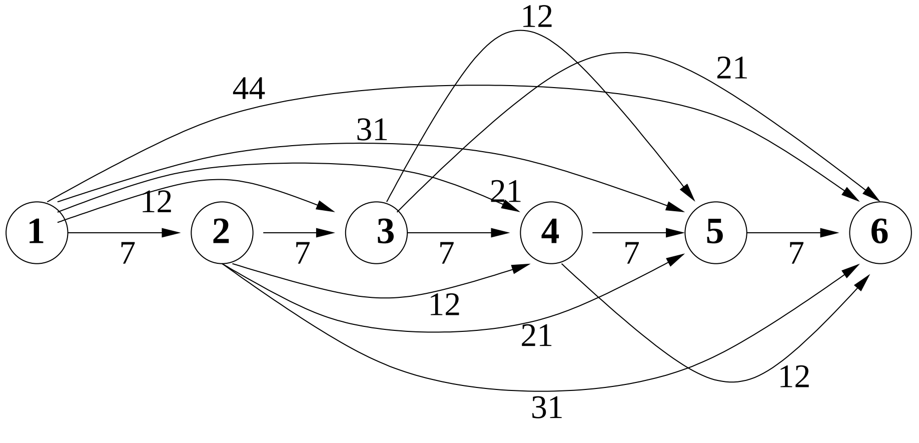
We obtain $d_{44} = -2 < 0$ and the algorithm halts: we found a circuit with a total negative cost of -2 ($4 \to 2 \to 3 \to 1 \to 4$).

2.7 An application of Dynamic Programming to machine renewal

Consider a directed graph with six nodes: nodes 1 to 5 are associated to the beginning of each year, while node 6 corresponds to the end of the 5 year time horizon. For each pair i, j, with $i, j = 1, \dots, 5$ and $i < j$, the arc (i, j) that represents the choice of bying a machine at the beginning of year i and selling it at the beginning of year j. The cost c_{ij} of arc (i, j) is defined as the net cost of the corresponding "partial renewal plan" from the beginning of year i to the beginning of year j, namely

$$c_{ij} = c_b + (\sum_{k=0}^{j-i-1} m_k) - r_{j-i},$$

where c_b is the buying cost of 12000 Euro, m_k is the annual maintenance cost for a machine that is k years old, and r_k is the selling price of a machine which is k years old. We obtain the following directed acyclic graph:



Any path from node 1 to node 6 corresponds to a (complete) renewal plan whose total net cost is equivalent to the total cost of the path. To look for a shortest path from node 1 to node 6, we apply the Dynamic Programming algorithm, obtaining

(a) $L(1) = 0$
(b) $L(2) = 7$, $pred(2) = 1$
(c) $L(3) = 12$, $pred(3) = 1$
(d) $L(4) = 19$, $pred(4) = 3$
(e) $L(5) = 24$, $pred(5) = 3$
(f) $L(6) = 31$, $pred(6) = 5$.

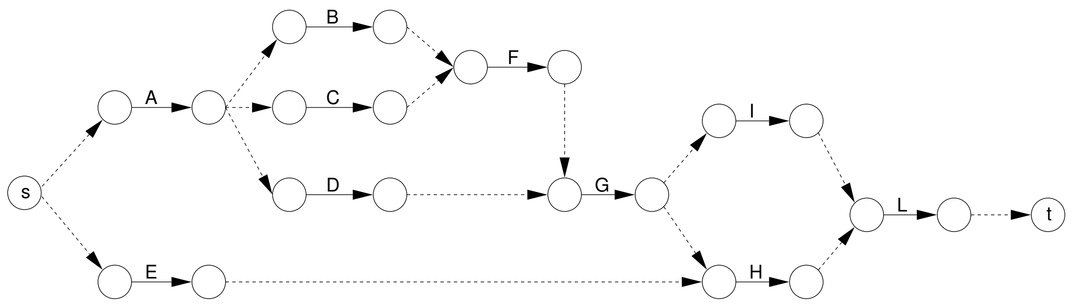
A shortest path (of cost 31) is $1 \to 3 \to 5 \to 6$. It amounts to buy a new machine every 2 years. Note that there are two other optimal solutions: path $1 \to 2 \to 4 \to 6$ and path $1 \to 3 \to 4 \to 6$.

2.8 **Project planning**

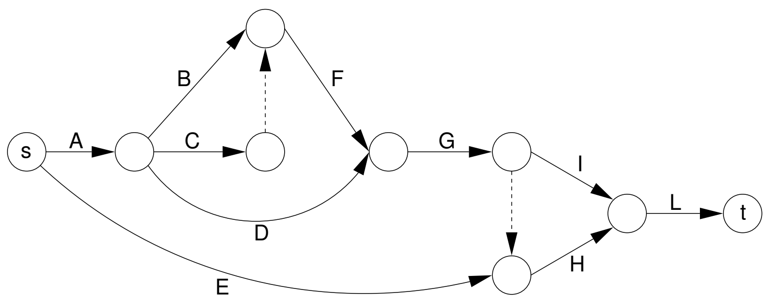
The following table indicates the duration of each activity and the related precedence relations:

Activities		Duration	Predecessors
A	Weight the ingredients	5	-
B	Meld the butter down	3	A
C	Mix flour, eggs, and sugar	5	A
D	Peel and cut the apples into slices	10	A
E	Heaten the oven	20	-
F	Add butter to the mixture	8	B,C
G	Add apples to the mixture	4	D,F
H	Cook the mixture in the oven	40	E,G
I	Whip the cream	10	G
L	Garnish	5	H,I

We derive the directed graph represeting the precedence relations as follows. For each activity, we introduce an arc whose cost is equivalent to the duration of the activity (its two nodes represent the beginning and the end of the activity). For each precedence relation $A_i < A_j$, a "fictitious" arc (i, j) of duration 0 is introduced (dashed line) between the ending node of the arc associated to A_i and the beginning node of the arc associated to A_j. We include a node s and, for each activity without predecessors associated to arc (v, w), we add the arc (s, v) of cost 0. Similarly, we include a node t and, for each activity without successors associated to arc $(v.w)$, we add the arc (w, t) of cost 0.



By contracting (deleting) some of the fictitious arcs while paying attention not to create any unwanted precedence relations, we obtain the following more compact directed acyclic graph representing the project:

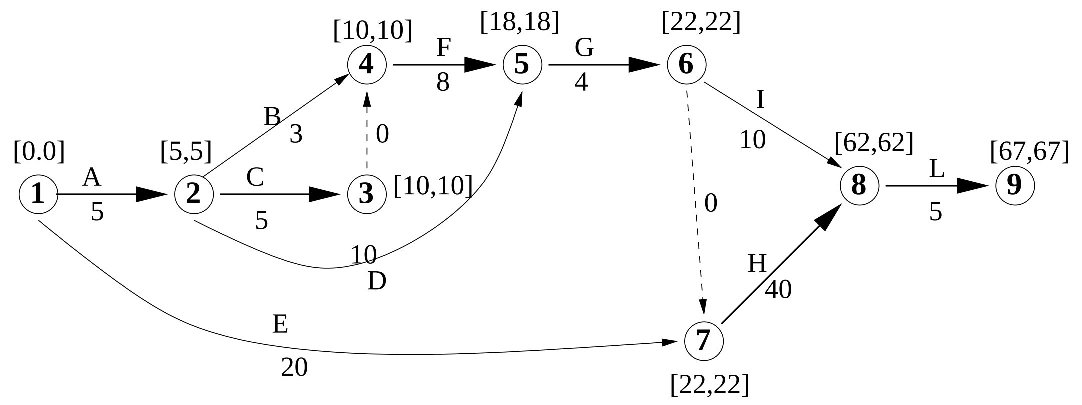


We use the Critical Path Method (CPM) to determine, for each node v of the graph, the "at earliest" and "at latest" times, denoted by $T_{min}(v)$ and respectively $T_{max}(v)$.

The algorithm exploits the topological order of the nodes and consists of two phases where Dynamic Programing is applied considering the n nodes in the increasing/decreasing order of the indices. Here is the pseudocode of the algorithm:

```
(a) Sort the nodes in topological order;
(b) $T_{min}(1) = 0$;
(c) FOR $h = 2, \dots, n$ DO $T_{min}(h) := \max\{T_{min}(i) + d_{ih} \mid (i,h) \in \delta^-(h)\}$;
(d) $T_{max}(n) = T_{min}(n)$;
(e) FOR $h = n-1, \dots, 1$ DO $T_{max}(h) := \min\{T_{max}(i) - d_{hi} \mid (h,i) \in \delta^+(h)\}$.
```

After ordering topologically the nodes, we obtain the following earliest times and latest times:



which are summarized in the following table:

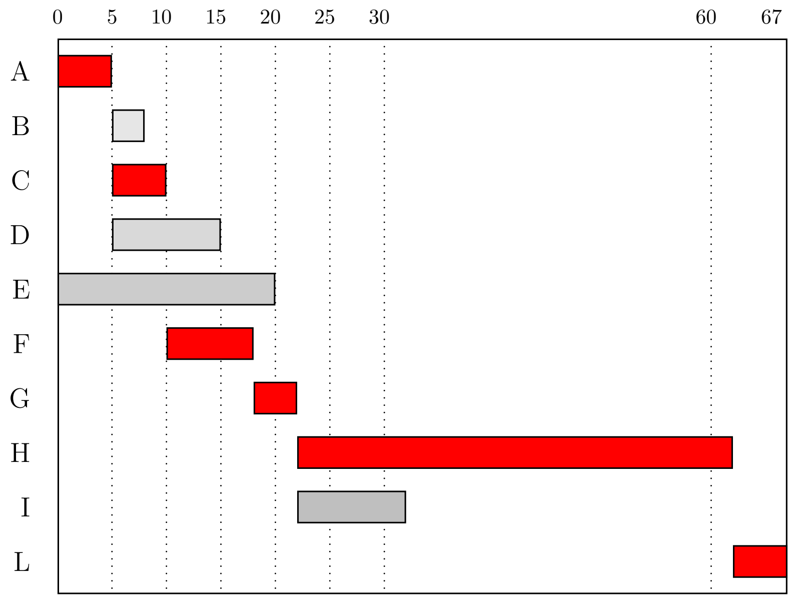
attivity	$T_{min}(i)$	$T_{max}(i)$	slack
A	0	0	0
B	5	7	2
C	5	5	0
D	5	8	3
E	0	2	2
F	10	10	0
G	18	18	0
H	22	22	0
I	22	52	30
L	62	62	0

The slacks for the activities are:

$$\begin{aligned}
\sigma(A) &= T_{max}(2) - T_{min}(1) - d_{12} = 5 - 0 - 5 = 0\\
\sigma(B) &= T_{max}(4) - T_{min}(2) - d_{24} = 10 - 5 - 3 = 2\\
\sigma(C) &= T_{max}(3) - T_{min}(2) - d_{23} = 10 - 5 - 5 = 0\\
\sigma(D) &= T_{max}(5) - T_{min}(2) - d_{25} = 18 - 5 - 10 = 3\\
\sigma(E) &= T_{max}(6) - T_{min}(1) - d_{16} = 22 - 0 - 20 = 2\\
\sigma(F) &= T_{max}(5) - T_{min}(4) - d_{45} = 18 - 10 - 8 = 0\\
\sigma(G) &= T_{max}(6) - T_{min}(5) - d_{56} = 22 - 18 - 4 = 0\\
\sigma(H) &= T_{max}(8) - T_{min}(7) - d_{78} = 62 - 22 - 40 = 0\\
\sigma(I) &= T_{max}(8) - T_{min}(6) - d_{68} = 62 - 22 - 10 = 30\\
\sigma(L) &= T_{max}(9) - T_{min}(8) - d_{89} = 67 - 62 - 7 = 0.
\end{aligned}$$

The critical activities are A, C, F, G, H, L.

The Gantt chart at earliest (where each activity (i, j) starts at the earliest time $T_{min}(i)$):



2.9 **Shortest paths with negative costs**.

We apply Floyd-Warshall's algorithm.

(a) Initialization

D	1	2	3
1	0	4	3
2	2	0	5
3	-2	-4	0

P	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

(b) Iteration $h = 1$ (triangulation on node 1)

$$d_{21} + d_{12} = 2 + 4 = 6 > d_{22} = 0$$
$$d_{21} + d_{13} = 2 + 3 = 5 = d_{23} = 5$$
$$d_{31} + d_{13} = -2 + 3 = 1 > d_{33} = 0$$
$$d_{31} + d_{12} = -2 + 4 = 2 > d_{32} = -4$$

D	1	2	3
1	0	4	3
2	2	0	5
3	-2	-4	0

P	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

(c) Iteration $h = 2$ (triangulation on node 2)

$$d_{12} + d_{21} = 4 + 2 = 6 > d_{11} = 0$$
$$d_{12} + d_{23} = 4 + 5 = 9 > d_{13} = 3$$
$$d_{32} + d_{23} = -4 + 5 = 1 > d_{33} = 0$$
$$d_{32} + d_{21} = -4 + 2 = -2 = d_{31} = -2$$

D	1	2	3
1	0	4	3
2	2	0	5
3	-2	-4	0

P	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

(d) Iteration $h = 3$ (triangulation on node 3)

$$d_{13} + d_{31} = 3 - 2 = 1 \; > \; d_{11} = 0$$
$$d_{13} + d_{32} = 3 - 4 = -1 \; < \; d_{12} = 4$$
$$\Rightarrow \textbf{update } d_{12}, p_{12}$$
$$d_{23} + d_{32} = 5 - 4 = 1 \; > \; d_{22} = 0$$
$$d_{23} + d_{31} = 5 - 2 = 3 \; > \; d_{21} = 2$$

D	1	2	3
1	0	**-1**	3
2	2	0	5
3	-2	-4	0

P	1	2	3
1	1	**3**	1
2	2	2	2
3	3	3	3