Network Flows

UPCOPENCOURSEWARE number 34414

Topic 3:
Shortest Path Problems

F.-Javier Heredia

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament d'Estadística
i Investigació Operativa

3.- Shortest Paths Problems (SPP)
(Chap. 4&5 Ahuja,Magnanti,Orlin)

- Notation and assumptions.
- Classification of Shortest Paths Problems.
- *Label-setting algorithm*: Dijkstra.
  - Description.
  - Example.
  - Convergence.
  - Complexity.
- *Label-correcting algorithm.*
  - Description.
  - Example.
  - Properties, convergence and negative cycles.
- Exercises.

Notation and assumptions.

- **SPP**: given a directed network $G=(N,A)$ with cost c, the SPP is to determine how to send one unit flow as cheaply as possible from node s to each one of the nodes (N-$\{s\}$).

$$\begin{cases} \min & z = \sum_{(i,j)\in A} c_{ij} x_{ij} & (4.1a) \\ \text{s.to:} & & \\ & \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} n-1 & i = s \\ -1 & \forall i \in N - \{s\} \end{cases} & (4.1b) \\ & x_{ij} \geq 0 \quad \forall (i,j) \in A & (4.1c) \end{cases}$$

- **Assumptions :**
  - *i. All costs **c** are integers.*
  - *ii. The network contains a directed path from node s to every other node in the network.*
  - *iii. The network does not contain any negative cycle (i.e. a directed cycle of negative length).*
  - *iv. The network is directed.*

Classification of the SPP and algorithms

- **Types of SPP:**
  - Finding shortest paths from one node to all other nodes when arc lengths are nonnegative.
  - Finding shortest paths from one node to all other nodes for networks with arbitrary arc lengths.
  - Finding shortest paths from every node to every other node.
- **SPP algorithms:**
  - ***Label-setting***: valid for (a) SPP defined on acyclic networks (no directed cycles) and (b) for SPP with nonnegative arc lengths. The fastest SPP alg.
  - ***Label-correcting***: valid for all classes of SPP, including those with negative arc lengths.

Dijkstra's Algorithm: description.

Dijkstra's algorithm: finds the shortest path from the source node s to all other nodes over the graph $G=(N,A)$ with non-negative arc costs $c_{ij} \geq 0$.

begin

$S:=\emptyset$; $\underline{S}:=N$;

$d(i):=\infty$ for each node $i \in N$; $d(s):=0$; $pred(s):=0$;

while $|\ S\ | < n$ **do**

let $i \in \underline{S}$ be a node such that $d(i) = \min\{\ d(j) : j \in \underline{S}\ \}$;

$S:= S \cup \{i\}$; $\underline{S}:= \underline{S} - \{i\}$;

for each $(i, j) \in A(i)$ **do**

if $d(j) > d(i) + c_{ij}$ **then** $d(j):= d(i) + c_{ij}$ and $pred(j):= i$;

end while

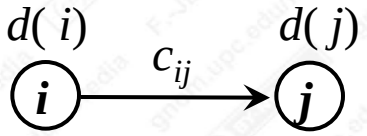
end

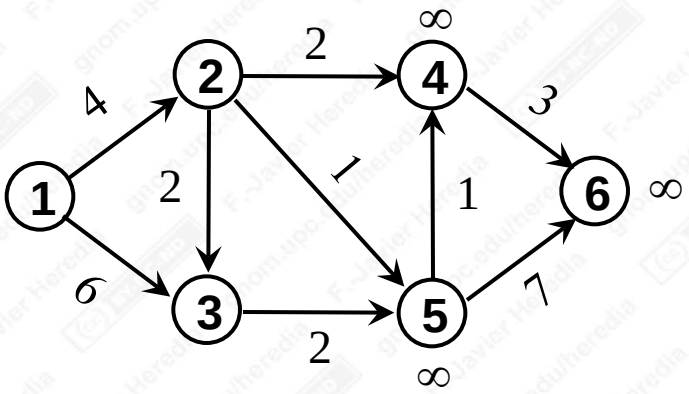
- ***d(i)*** : label that represents the distance from node s to node i.
- ***pred(j)*** : $pred(j) = i \Rightarrow$ the arc (i,j) belong to the shortest path between s and j .
- ***A(i)*** : set of arcs which origin is the node i.

Dijkstra's Algorithm: example.

- We can see the main idea in the following Applet:
  - http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/DijkstraApp.shtml?demo1

- Solve the following SPP with Dijkstra:





(Solution)

Dijkstra's Algorithm: convergence (I).

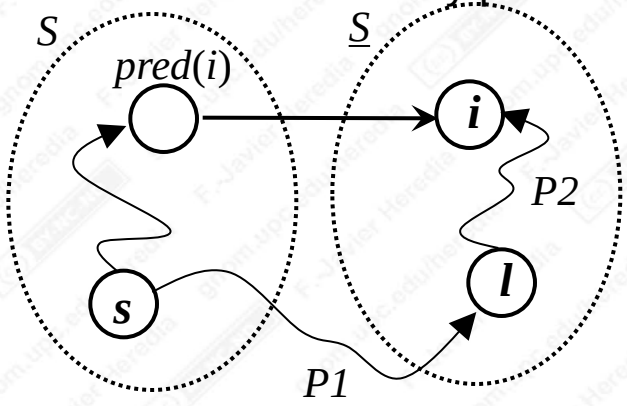
- **Convergence Theorem**:
  "*At termination the Dijkstra's algorithm, the labels* $d(i)$ *of nodes* $i \in N$ *represent the distances of the shortest path* $s \to i$."
- **Proof:** by induction. Assuming that at the iteration k of it is true that:
  a) The distance $d(i)$ of the nodes S is optimal.
  b) The distance $d(j)$ of the nodes $\underline{S}$ is the shortest path length from the source provided that each internal node in the path lies in S.

  It will be shown that at iteration $k+1$ both hypothesis are still true. At the end of the algorithm, $S = N$ and the hypothesis a) is verified in all nodes

Dijkstra's Algorithm: convergence (II).

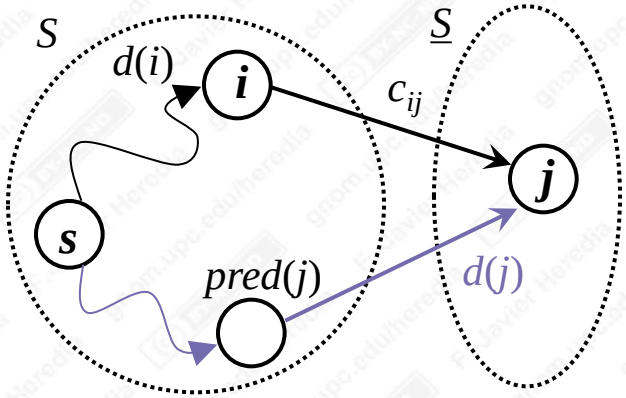
Proof a)

- Let $i \in \underline{S}$ be the node to be transferred to S.
- By the hypothesis b), $d(i)$ is the optimal distance $s \to i$ over S.
- We will show that for any path P $s \to i$ passing through $\underline{S}$ the distance $d(P)$ is $\geq d(i)$:



  - Let $P=P1+P2$ be any path $s \to i$ that contains at least one node $l \in \underline{S}$, $l \neq i$.
  - b) $\Rightarrow$ the length $P1$ is $d(P1) \geq d(l)$.
  - $d(i) \leq d(l) \leq d(P1)$
  - $c_{ij} \geq 0 \Rightarrow d(P2) \geq 0 \Rightarrow d(P)=d(P1+P2) \geq d(i)$ there is not any path P over $\underline{S}$ with a distance lower than $d(i) \Rightarrow d(i)$ is the optimal distance $s \to i$ ■

Proof b)



- Let $i \in \underline{S}$ be the node to be transferred to S.
- Let's assume that b) is true, i.e. the length $d(j)$ of the nodes of $\underline{S}$ are minimal over S.
- Once transferred i to S , the labels and predecessors of $\underline{S}$ are updated: if $d(j) > d(i) + c_{ij}$ then the shortest path $s \to j$ over S includes $i \Rightarrow$ $d(j) := d(i) + c_{ij}$ is the length of the new shortest path $s \to j$ over S ■

Dijkstra‘s Algorithm: complexity.

- **Running time (algorithmic complexity)**: the compational burden of the Dijkstra’s algorithm is due to two operations:
  - **Node selection:** in every iteration all the nodes of $\underline{S}$ are explored in order to select the node to be transferred to S : $n+(n\text{-}1)+(n\text{-}2)+...+2+1 = (n^2+n)/2 = O(n^2)$
  - **Distance updates:** the algorithm perform this operation $|A(i)|$ times for node $i \in N$. Each distance update operation requires $O(1)$ time: hence total time is $O(m)$. $\sum_{i \in N} |A(i)| = m$
- Dijkstra solve the SPP in $O(n^2)$ time (the best possible time for completely dense networks).
  - The simplex algorithm for network flow SPP requires $O(n^2m)$, $O(n^3)$ or $O\ (nmlog\ C)$ depending on the implementation.

Label-correcting algorithm

Label-Correcting algorithm: finds the shortest path from the source node s to the rest of nodes over a graph $G=(N,A)$ without negative cycles.

begin

$d(s):= 0$; $pred(s):= 0$;

$d(j):= \infty$ for every node $j \in N-\{s\}$;

$c^d_{ij}:= c_{ij} + d(i) - d(j)\ \forall (i,j) \in A$;

while $\exists (i,j) \in A : c^d_{ij} < 0$ **do**

$d(j):= d(i) + c_{ij}$; $pred(j):= i$;

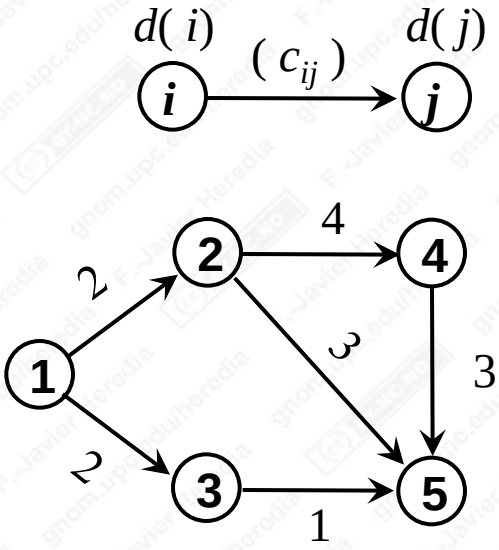
$c^d_{kl}:= c_{kl} + d(k) - d(l)\ \forall (k,l) \in A \mid k=j \text{ ó } l=j$;

end while

end

- $\boldsymbol{c^d_{ij} = c_{ij} + d(i) - d(j)}$: reduced arc length.
- $\boldsymbol{d(i)}$: label that represents the distance from s to node i.
- $\boldsymbol{pred(j)}$: $pred(j) = i \Rightarrow$ arc (i,j) belongs to the shortest path between s and j .

Label-correcting algorithm: example.



(Solution)

Label-Correcting Algorithm: properties.

- The value of the reduced arc length c^d_{ij} at the solution represents the increase of the optimal cost of the SPP if arc (i,j) would belongs to the solution.
- Computational cost: $O(n^2C)$ with

$$C=\max\{c_{ij} : (i,j)\in A\}$$

  - Proof: page 140. AMO
  - Dijkstra $O(n^2)$

Label-correcting algorithm: convergence.

- **Convergence Theorem**:

  "*Let $d(j)$ be the label of node $j \in N$ that represents the distance of a path $s \to j$. Then $d(j)$ represents the distances of the SPP if and only if the following optimality conditions are satisfied:*

$$d(j) \leq d(i) + c_{ij} \quad \forall (i,j) \in A \text{"} \qquad (1)$$

- **Proof:**

  $\Rightarrow$ If $d(j)$ is the distance of the shortest path $s \to j$, then (1) applies: otherwise the shortest path $s \to i \to j$ would improve the distance $d(j)$ ■

  $\Leftarrow$ Let's assume that (1) applies. Let´s consider the directed path

$$s = i_1 \to i_2 \to i_3 \to \ldots \to i_k = j$$

  Applying (1) to nodes i_1 to i_k we get:

$$\begin{aligned} d(j) = d(i_k) &\leq d(i_{k-1}) + c_{i_{k-1}i_k} \\ d(i_{k-1}) &\leq d(i_{k-2}) + c_{i_{k-2}i_{k-1}} \\ \vdots &\quad \vdots \\ d(i_2) &\leq d(i_1) + c_{i_1 i_2} \end{aligned}$$

  Adding all the inequalities we have:

$$d(j) \leq c_{i_{k-1}i_k} + c_{i_{k-2}i_{k-1}} + \ldots + c_{i_1 i_2} = \sum_{(i,j) \in P} c_{ij} \quad \forall \text{camí } P$$

  Furthermore $d(j)$ is optimal.

Label-correcting algorithm: negative cycles.

- **Negative cycle:** directed cycle W such that $\sum_{(i,j)\in W} c_{ij} < 0$
- **Theorem:** " if the graph $G=(N,A)$ *contains any negative cycle, then it is not possible to find a set of optimal distances* $d(\cdot)$ *that satisfy* (1)"
  - **Proof:** if W is a negative cycle, we can verify that

$$\sum_{(i,j)\in W} c_{ij}^{d} = \sum_{(i,j)\in W} c_{ij} < 0$$

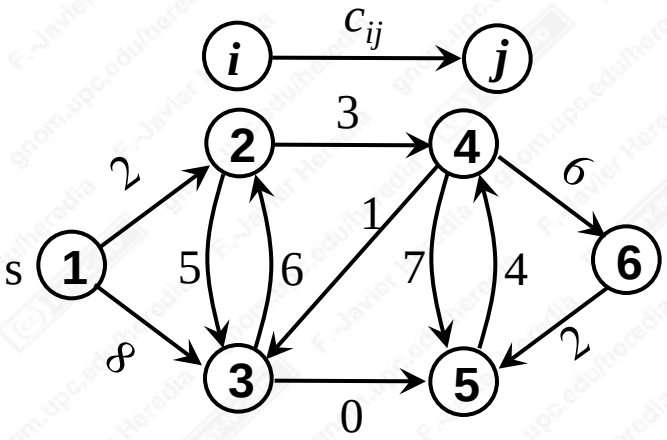
On the other hand, if $d(\cdot)$ are optimal, (1) imposes

$$\sum_{(i,j)\in W} c_{ij}^{d} \geq 0$$

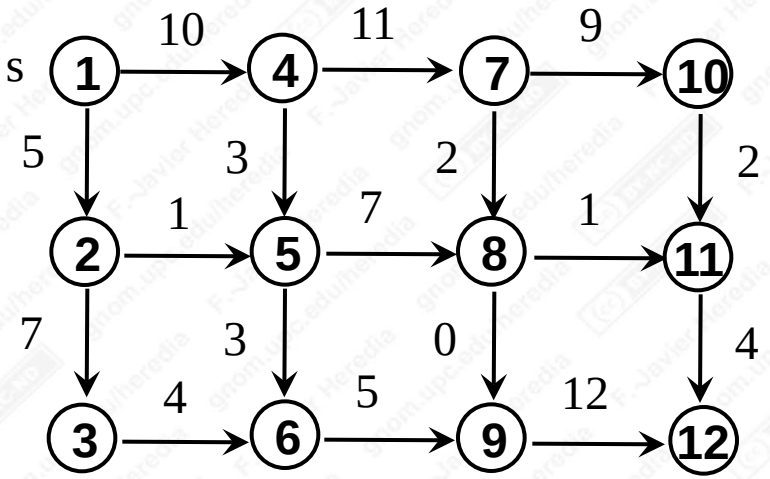
Which is a contradiction■

Dijkstra Algorithm: exercises (I).

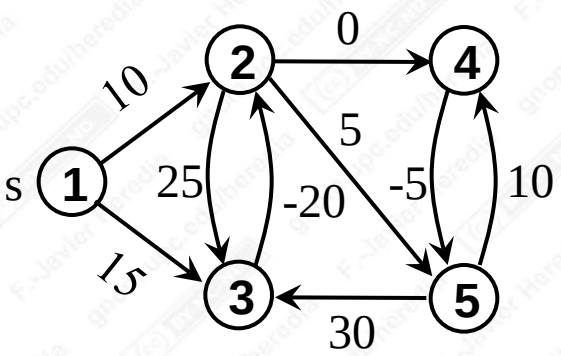
- Solve the SPP of the following networks with the appropriate algorithm.



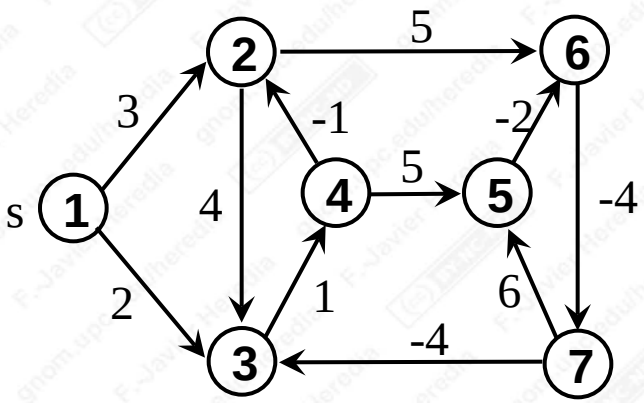
(a)



(b)



(c)



(d)

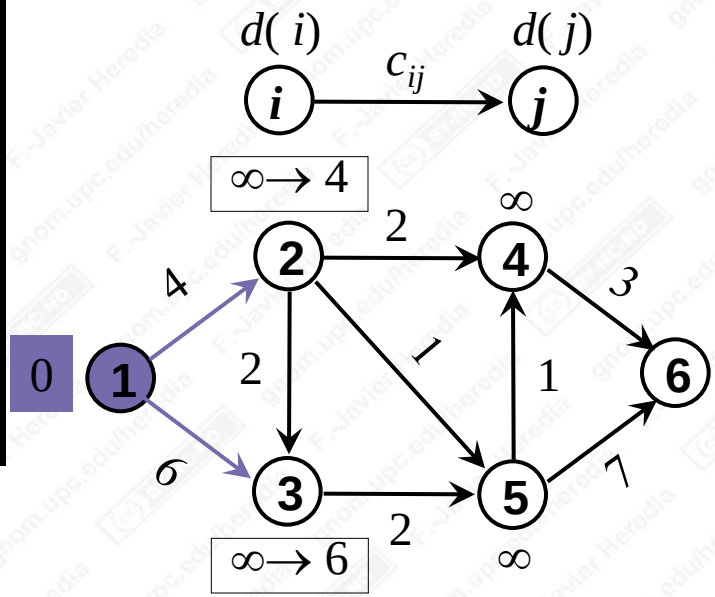
Dijkstra algorithm: exercises (II).

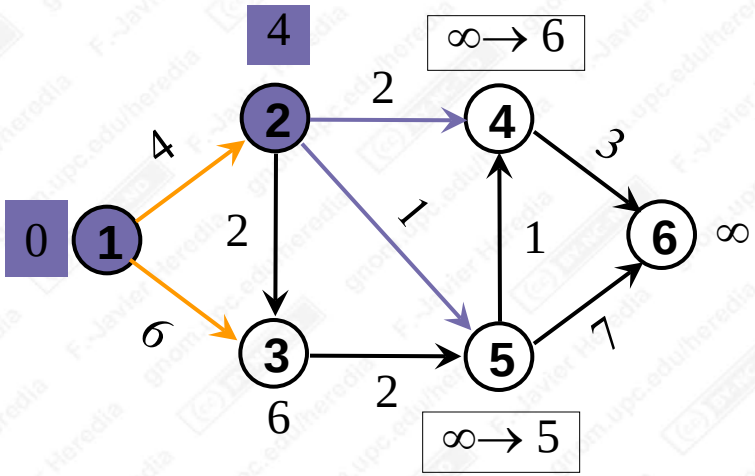
- Solve the following exercises from book AMO:

  4.1,4.3, 4.6, 4.7, 4.9, 4.13.

- Each one shows an application of the SPP. The tasks to complete are:
  - To formulate the SPP.
  - If the statement does not specify the MFP formulation, complete the network topology with appropriate costs.
  - Solve numerically with GIDEN the SPP formulated
- Send the answers (.pdf o .doc i .gdn) by e-mail before the date: ../../..

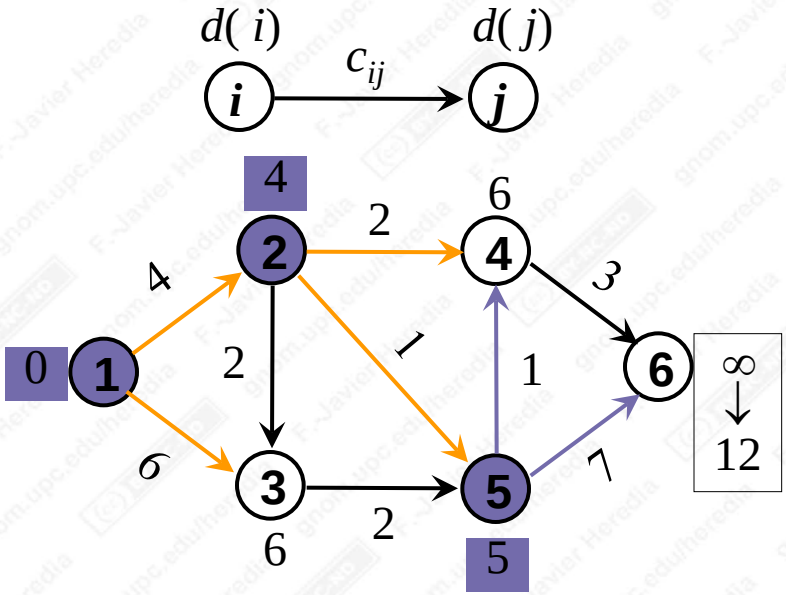
Dijkstra algorithm: example (I).



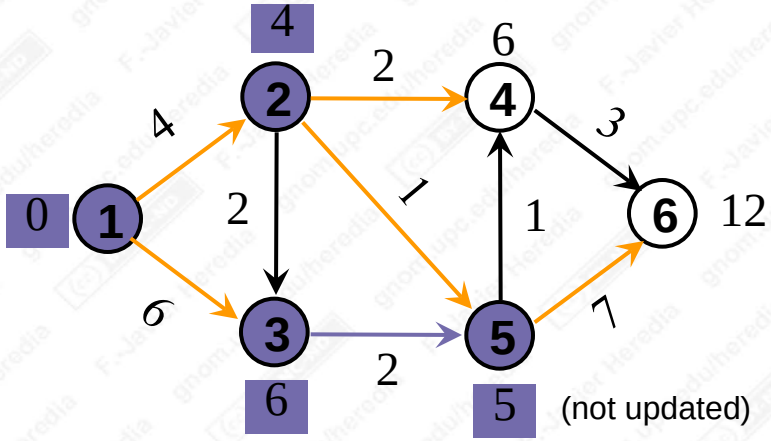


- **Initialization**:
  - $S := \emptyset$; $\underline{S} := \{1,2,3,4,5,6\}$
  - $d := [0, \infty, \infty, \infty, \infty, \infty]$
  - $pred := [0, -, -, -, -, -]$
- **1st iteration**: $|S| = 0 < n = 6$
  - $d(1) = \min\{ d(j) : j \in \underline{S} \} = 0$
  - $S := \{1\}$; $\underline{S} := \{2,3,4,5,6\}$
  - $d := [0, 4, 6, \infty, \infty, \infty]$
  - $pred := [0, 1, 1, -, -, -]$
- **2nd iteration**: $|S| = 1 < n = 6$
  - $d(2) = \min\{ d(j) : j \in \underline{S} \} = 4$
  - $S := \{1,2\}$; $\underline{S} := \{3,4,5,6\}$
  - $d := [0, 4, 6, 6, 5, \infty]$
  - $pred := [0, 1, 1, 2, 2, -]$

Dijkstra Algorithm: example (II).

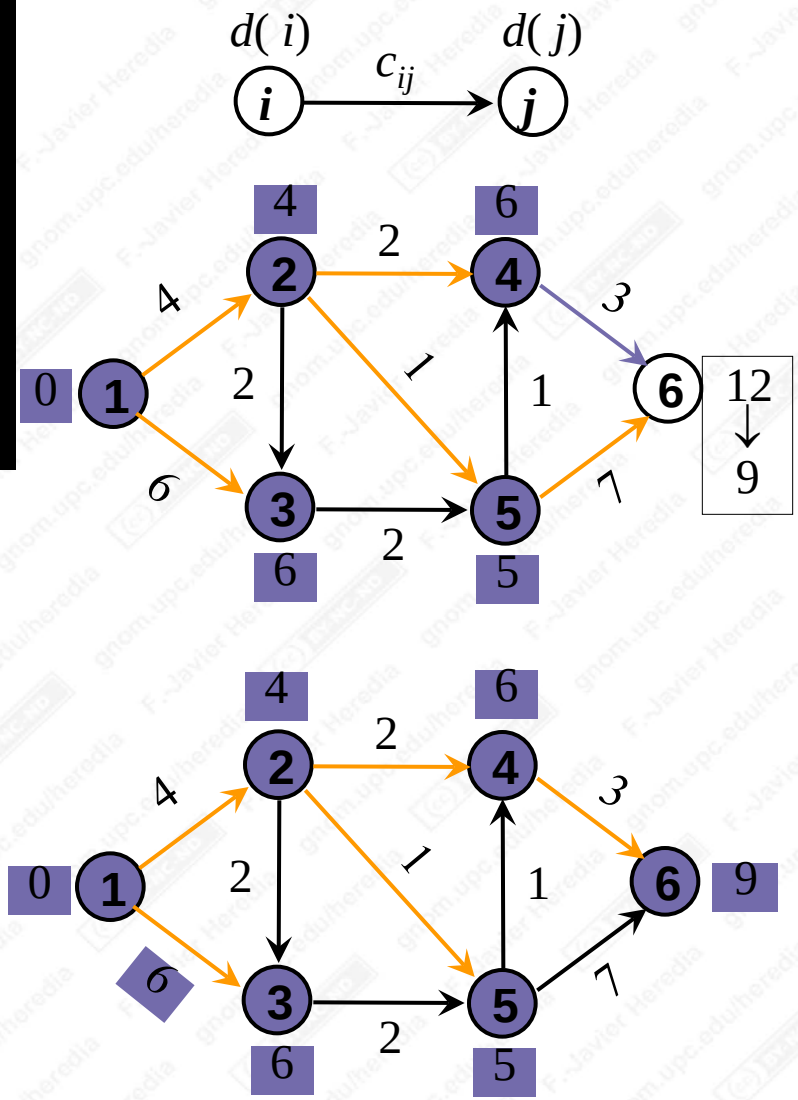


- **3rd iteration**: $|S| = 2 < n = 6$
  - $d(5) = \min\{ d(j) : j \in \underline{S} \} = 5$
  - $S := \{1,2,5\}$; $\underline{S} := \{3,4,6\}$
  - $d := [0, 4, 6, 6, 5, 12]$
  - $pred := [0, 1, 1, 2, 2, 5]$



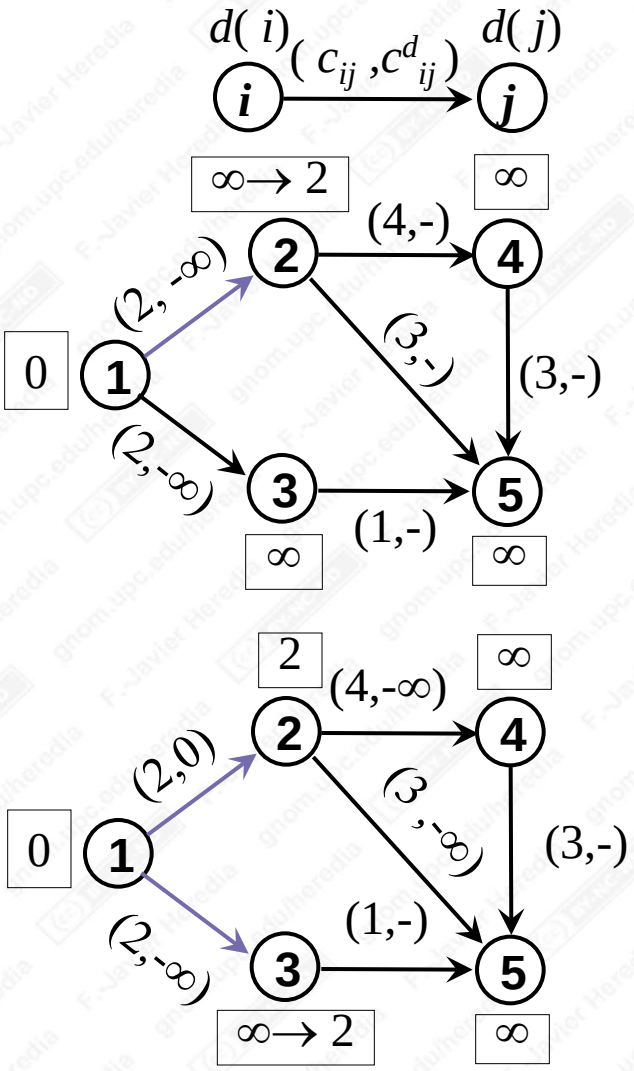
- **4th iteration**: $|S| = 3 < n = 6$
  - $d(3) = \min\{ d(j) : j \in \underline{S} \} = 6$
  - $S := \{1,2,3,5\}$; $\underline{S} := \{4, 6\}$
  - $d := [0, 4, 6, 6, 5, 12]$
  - $pred := [0, 1, 1, 2, 2, 5]$

Dijkstra Algorithm: example (III).



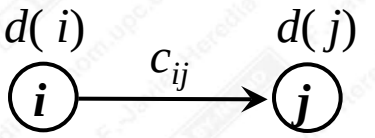
- **5th iteration**: $|S| = 4 < n = 6$
  - $d(4) = \min\{ d(j) : j \in \underline{S} \} = 6$
  - $S := \{1,2,3,4,5\}$; $\underline{S} := \{6\}$
  - $d := [0, 4, 6, 6, 5, 9]$
  - $pred := [0, 1, 1, 2, 2, 4]$
- **6th iteration**: $|S| = 5 < n = 6$
  - $d(6) = \min\{ d(j) : j \in \underline{S} \} = 9$
  - $S := \{1,2,3,4,5,6\}$; $\underline{S} := \varnothing$
  - $d := [0, 4, 6, 6, 5, 9]$
  - $pred := [0, 1, 1, 2, 2, 4]$
- **7th iteration**: $|S| = n = 6$: **STOP**
- **Shortest path tree:**
  $T = \{(1,2),(1,3),(2,4),(2,5),(4,6)\}$

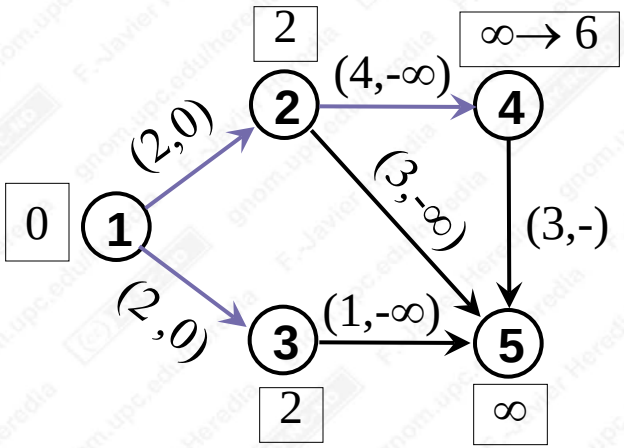
Label-correcting algorithm: example (I).



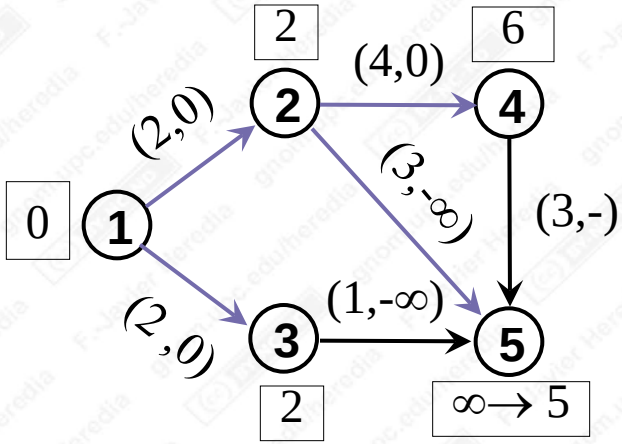
- **Initialization**:
  - d:= [0, ∞, ∞, ∞, ∞];
  - *pred* := [0, -, -, -, -, -];
  - c^d := [-∞, -∞, -,-,-,-];
- **1st iteration**: $c^d_{12} < 0$
  - d:= [0, 2, ∞, ∞, ∞];
  - *pred* := [0, 1, -, -, -];
  - c^d := [0, -∞, -∞, -∞,-,-];
- **2nd iteration**: $c^d_{13} < 0$
  - d:= [0, 2, 2, ∞, ∞];
  - *pred* := [0, 1, 1, -, -];
  - c^d := [0, 0, -∞, -∞, -∞,-];

Label-correcting algorithm: example (II).



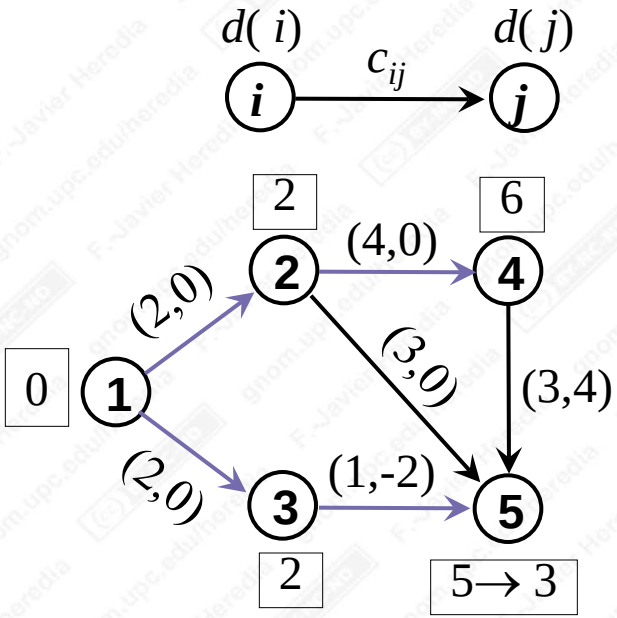


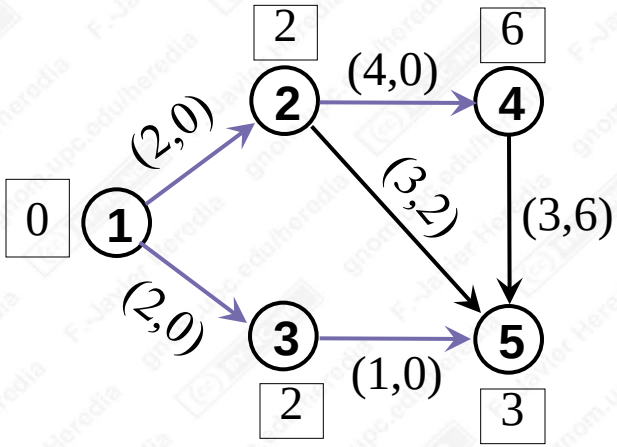
- **3rd iteration**: $c^d_{24} < 0$
  - d:= [0, 2, 2, 6, ∞];
  - *pred* := [0, 1, 1, 2, -];
  - c^d := [0, 0, 0, -∞, -∞,-];



- **4th iteration**: $c^d_{25} < 0$
  - d:= [0, 2, 2, 6, 5];
  - *pred* := [0, 1, 1, 2, 2];
  - c^d := [0, 0, 0, 0, -2, 4];

Label-correcting algorithm: example (III).





- **5th iteration:** $c^d_{35} < 0$
  - d:= [0, 2, 2, 6, 3];
  - *pred* := [0, 1, 1, 2, 3];
  - c^d := [0, 0, 0, 2, 0, 6];
- **6th iteration:** $c^d \geq 0$ **: STOP**