


6 Optimally solving the Shortest Path Problem

- In what follows, we apply specific variants of the Primal-Dual Algorithm in order to derive new algorithms for the Shortest Path (Section 6) and for the Max-Flow Problem (Section 7)
- We commence our study with the Shortest Path Problem
- In the literature, two main types of shortest path problems are distinguished
  - The **single source shortest path problem**  
Find the shortest path from one distinguished node to all other nodes in the network
  - The **all pairs shortest path problem**  
Find the shortest path between all pairs of nodes in the network

Schumpeter School  
of Business and Economics

Business Computing and Operations Research

**WINEFOR** 519

---

---

---

---

---


---

---

---

Overview of the Section

- The **single source shortest path problem**
  - In Section 6.1, we will derive the famous Dijkstra algorithm as a special extended Primal Dual procedure
  - However, this procedure is not able to handle negative weights
  - Therefore, in Section 6.2, we consider the Bellman-Ford algorithm
- The **all pairs shortest path problem**
  - In Section 6.3, we finally introduce the Floyd Warshall procedure that is also able to deal with negative arc weights
  - It is also able to identify cycles of negative length

Schumpeter School  
of Business and Economics

Business Computing and Operations Research

**WINEFOR** 520

---

---

---

---

---

---


---

---

6.1 Deriving the Dijkstra algorithm

First of all, we have to introduce the problem of finding the shortest path from a distinguished node to all other nodes in a network

- In what follows, we consider directed weighted graphs
- In order to provide a complete LP-based problem definition of this Shortest Path Problem, we introduce several basic notations

Schumpeter School  
of Business and Economics

Business Computing and Operations Research

**WINEFOR** 521

---

---

---

---

---

---

---

---

# Graph, Network, ...

## 6.1.1 Definition

Assuming  $V$  is a finite set, in what follows, defined as

$$V = \{1, \dots, n\}, n \in \mathbb{N},$$

$$E = \{e_1, \dots, e_m\} \subseteq (V \times V) \setminus D, D = \{(v, v) \mid v \in V\}, \text{ and } c : E \rightarrow \mathbb{R}.$$

Then,  $N = (V, E, c)$  is denoted as a weighted directed graph (also denoted as a network).  $V$  is denoted as the vertices (nodes) and  $E$  the set of arcs.  $c(e)$  indicates the weight (length, costs) of the arc  $e \in E$ .

---

---

---

---

---

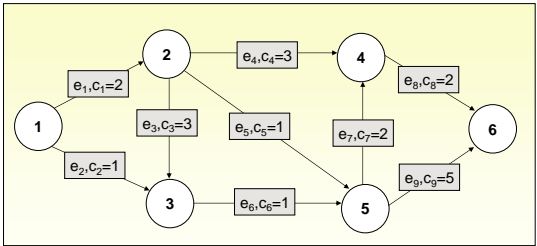
---

---

---

## A simple example

$$V = \{1, 2, 3, 4, 5, 6\}, E = \{(1, 2), (1, 3), (2, 3), (2, 4), (2, 5), (3, 5), (5, 4), (4, 6), (5, 6)\},$$
$$c = (2, 1, 3, 3, 1, 1, 2, 2, 5)^T \quad (c \in \mathbb{R}^9, c_j = c(e_j))$$



---

---

---

---

---

---

---

---

## Adjacency lists

- In general:  $v: w_1, c(v, w_1)$
- 1: 2,2; 3,1
- 2: 3,3; 4,3; 5,1
- 3: 5,1
- 4: 6,2
- 5: 4,2; 6,5
- 6: -

---

---

---

---

---

---

---

---

## Vertex-arc adjacency matrix

$$\tilde{A} = (\tilde{a}_{i,k})_{1 \leq i \leq n, 1 \leq k \leq m}, \text{ with } \tilde{a}_{i,k} = \begin{cases} +1 & \text{when } \exists j \in V : e_k = (i, j) \\ -1 & \text{when } \exists j \in V : e_k = (j, i) \\ 0 & \text{otherwise} \end{cases}$$

$\tilde{a}_{i,k} = 1 \Rightarrow i$  is source of arc  $e_k$ ;  $\tilde{a}_{i,k} = -1 \Rightarrow i$  is sink of arc  $e_k$

$e_k = (i, j) \Rightarrow \tilde{a}^k = e^i - e^j$ , with  $e^i$  as the  $i$ th unit vector

$$\Rightarrow \tilde{A} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

## Path

### 6.1.2 Definition

Assuming  $N = (V, E, c)$  is a weighted directed graph (also denoted as a network). Then, a path leading from  $i_0 \in V$  to  $i_k \in V$  is a sequence of nodes  $\langle i_0, i_1, i_2, \dots, i_k \rangle$ , with  $e_t = (i_t, i_{t+1})$ ,  $0 \leq t \leq k-1$ .

The length (weight, costs) of the path is calculated by

$$c(\langle i_0, i_1, i_2, \dots, i_k \rangle) = \sum_{t=0}^{k-1} c(e_t) = \sum_{t=0}^{k-1} c(i_t, i_{t+1}).$$

If  $i_k = i_0$ , the path  $\langle i_0, i_1, i_2, \dots, i_k \rangle$  is denoted as a cycle

## Definition of variable $x$

Assuming  $p = \langle i_0, i_1, i_2, \dots, i_k \rangle$  is a path in a network  $N$ . Then, we define  $x \in \mathbb{R}^m$  as follows

$$x_i = \begin{cases} 1 & \text{if } e_i = (i_l, i_{l+1}), l \in \{0, 1, 2, \dots, k-1\} \\ 0 & \text{otherwise} \end{cases}$$

Then, we obtain

$$\tilde{A} \cdot x = \sum_{t=0}^{k-1} \alpha^{i_t} = \sum_{t=0}^{k-1} (e^{i_t} - e^{i_{t+1}}) = e^{i_0} - e^{i_k}$$

If  $p$  is cyclic, we have  $\tilde{A} \cdot x = e^{i_0} - e^{i_k} = e^{i_0} - e^{i_0} = 0$

## Consequences

The other way round

$\tilde{A} \cdot x = 0 \Rightarrow x$  defines a sequence of cycles in  $N$

$\tilde{A} \cdot x = e^i - e^j$  defines a path from  $i$  to  $j$  (may be combined with a sequence of cycles)

In what follows, we assume that  $c_i > 0, \forall i \in \{1, \dots, m\}$

## The Shortest Path Problem

- Generate a path from  $i$  to  $j$

Minimize  $c^T \cdot x$

s.t.

$$\tilde{A} \cdot x = e^i - e^j \wedge x \in \mathbb{N}_0^m \stackrel{!}{\Rightarrow} x \in \{0, 1\}^m$$

Since we minimize the total flow, this problem is equivalent to restricting the variable vector  $x$  to  $\{0, 1\}^m$ .

## Observation

- By adding all rows of the matrix, we obtain the null vector
- This results from the fact that each column represents an arc with a definitely defined **source** and **sink** (represented by the entries 1 and -1)
- Consequently,  $m-1$  is an upper bound of the rank of the matrix
- We denote  $A$  as the resulting matrix that arises by erasing the last row in  $\tilde{A}$
- Hence, in what follows, we consider the following general Shortest Path Problem

## The Shortest Path Problem

- Generate a path from 1 to destination n

Minimize  $c^T \cdot x$

s.t.

$$A \cdot x = e^1 \wedge x \in \{0,1\}^m$$

Then, we get the corresponding dual problem

$$\text{Maximize } (e^1)^T \cdot \pi = \pi_1,$$

$$\text{s.t., } A^T \cdot \pi \leq c \Leftrightarrow \pi_i - \pi_j \leq c(i,j), \forall e_k = (i,j) \in E$$

$\pi$  free

- Note that in the section named "Integer Programming" we will see that this problem is equivalent to its LP-relaxation (switching back to continuous variables)

## The RP and its dual counterpart

Based on a dual solution  $\pi$  and the resulting sets  $J$  and  $J^c$ , we define the reduced problem  $\text{RP}(\pi)$  as follows:

$$\text{Minimize } \sum_{j=1}^n x_j^a,$$

$$\text{s.t., } \left( E_n, (a^i)_{i \in J} \right) \cdot \begin{pmatrix} x_j^a \\ x_j^b \end{pmatrix}_{1 \leq j \leq n} = e^1 \wedge x \in \text{IN}_0^m = \{0,1\}^m$$

Hence, we get the corresponding dual of the reduced problem  $\text{DRP}(\pi)$

$$\text{Maximize } (e^1)^T \cdot \pi = \pi_1,$$

$$\text{s.t., } \pi \leq 1 \wedge (a^i)_{i \in J}^T \cdot \pi \leq 0 \Leftrightarrow \pi_i - \pi_j \leq 0, \forall e_k = (i,j) \in E \wedge k \in J$$

$\pi$  free

## Solving $\text{DRP}(\pi)$

Hence, we get the corresponding dual of the reduced primal problem  $\text{DRP}(\pi)$

$$\text{Maximize } (e^1)^T \cdot \pi = \pi_1,$$

$$\text{s.t., } \pi \leq 1^n \wedge \pi_i - \pi_j \leq 0, \forall e_k = (i,j) \in E \wedge k \in J$$

Let us consider the problem  $\text{DRP}(\pi)$ . In what follows, we denote a solution to  $\text{DRP}(\pi)$  as  $\bar{\pi}$ . Obviously, each feasible solution with  $\bar{\pi}_1 = 1$  is optimal. Thus, we have to follow all paths generated by the edges of set  $J$ .

## Solving DRP( $\pi$ )

Hence, if node  $i$  is reachable from node 1, we define  $\bar{\pi}_i = 1$ . But, if we commence our examination at the destination  $n$ , we know that it holds  $\bar{\pi}_i \leq 0, \forall i \in V$  with  $(i, n) \in J$ .

Note that this results from the fact that  $\bar{\pi}_i - \bar{\pi}_n \leq 0$  has to be fulfilled and  $\bar{\pi}_n$  was erased by replacing  $\tilde{A}$  with  $A$ . Thus, we obtain  $\bar{\pi}_i \leq 0$ .

## Solving DRP( $\pi$ )

Obviously, in these constellations, we can set  $\bar{\pi}_i = 0$ .

This value is also propagated along each path generated by arcs of set  $J$ . Consequently, we may conclude

$$\bar{\pi}_i = \begin{cases} 1 & \text{when there exists a path in } J \text{ from 1 to } i \\ 0 & \text{when there exists a path in } J \text{ from } i \text{ to } n \\ a \leq 1 & \text{otherwise} \end{cases}$$

In what follows, we define  $a = 1$  in order to distinguish two sets of nodes

$$W = \{i \mid i \in V \wedge \bar{\pi}_i = 0\} \wedge W^c = \{i \mid i \in V \wedge \bar{\pi}_i \neq 0\}.$$

## Solving DRP( $\pi$ )

$$W = \{i \mid i \in V \wedge \bar{\pi}_i = 0\} \wedge W^c = \{i \mid i \in V \wedge \bar{\pi}_i \neq 0\}.$$

In order to generate a shortest path from 1 to  $n$ , in case  $\bar{\pi}_1 = 1$ , we have to add additional arcs  $j \notin J$ .

We know  $\forall (i, j) \in E$ , with  $(i, j) \notin J : c_{i,j} - \pi_i + \pi_j > 0$

We consider those edges that have negative relative costs, i.e., it holds:  $0 - \bar{\pi}_i + \bar{\pi}_j < 0 \Leftrightarrow \bar{\pi}_i - \bar{\pi}_j > 0 \Rightarrow \bar{\pi}_i = 1 \wedge \bar{\pi}_j = 0$

The Primal-Dual Simplex generates

$$\begin{aligned} \lambda_0 &= \min \left\{ \frac{c_{i,j} - \pi_i + \pi_j}{\bar{\pi}_i - \bar{\pi}_j} \mid \forall (i, j) \in E \text{ with } (i, j) \notin J \right\} \\ &= \min \{ c_{i,j} - \pi_i + \pi_j \mid \forall (i, j) \in E \text{ with } (i, j) \notin J \} \end{aligned}$$

## Observations

- $\text{DRP}(\pi)$  determines a cut between the sets  
 $W = \{i \mid i \in V \wedge \bar{\pi}_i = 0\} \wedge W^c = \{i \mid i \in V \wedge \bar{\pi}_i \neq 0\}$
- The considered edges with  $\bar{\pi}_i = 1 \wedge \bar{\pi}_j = 0$  are just the edges that bridge the gap, i.e., they connect the incompleted path found to node  $n$  with the beginning of the graph
- $\pi_i$  indicates the length of the shortest path from  $i$  to  $n$ , for  $i \in W$ . This is the invariante of the procedure
- $\min \{c_{i,j} - \pi_i + \pi_j \mid \forall (i,j) \in E, \text{ with } (i,j) \notin J\}$  gives the length of the shortest edge bridging the gap between  $W$  and  $W^c$
- Specifically, for this edge it holds:  $c_{i,j} - \pi_i + \pi_j = 0 \Leftrightarrow \pi_i = c_{i,j} + \pi_j$

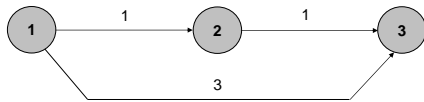
## Further observations

- If  $(i,j) \in E$  has become admissible, it stays admissible for the remaining calculations, i.e., it holds  $\pi_i - \pi_j = c_{i,j}$ . This results from the fact that  $\bar{\pi}_i = \bar{\pi}_j = 0$
- Consequently, we can conclude that if a node  $i$  has entered  $W$ , it stays there for the rest of the calculation process

## Applying the Primal-Dual Simplex

- Consider the dual of the Shortest Path Problem
- Obviously, since  $c \geq 0$ , we know that  $\pi=0$  is a first feasible solution to  $(D)$
- By making use of  $\pi=0$ , we have an initial dual solution in order to commence the calculation of the Primal-Dual Simplex Algorithm

### A simple example (warm up)



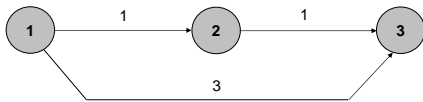
$\Rightarrow (P)$

Minimize  $c^T \cdot x = (1 \ 3 \ 1) \cdot x$ ,

s.t.,

$$\left( \tilde{A} = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix} \right) \Rightarrow A \cdot x = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \cdot x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

### A simple example (warm up)



$$(D) \text{ Maximize } \pi_1, \text{ s.t., } A^T \cdot \pi = \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \pi \leq c = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix}$$

We additionally set  $\pi_u = 0$

### Applying the Primal-Dual Simplex

$$\begin{aligned} \text{We have } \pi &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} - A^T \cdot \pi = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} \Rightarrow J = \emptyset \wedge J^c = \{1, 2, 3\} \end{aligned}$$



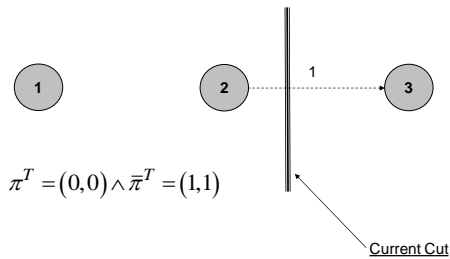
## RP( $\pi$ )

$$\begin{array}{c|cccccc|cccc} 0 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & -1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 1 \end{array} \Rightarrow 1$$

$$\Rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \bar{\pi} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Leftrightarrow \bar{\pi} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \lambda_0 = \min \left\{ \frac{c_2 - (0,0) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}}{\bar{\pi}^T \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}}, \frac{c_3 - (0,0) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}}{\bar{\pi}^T \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}} \right\}$$

$$= \min \left\{ \frac{3}{1}, \frac{1}{1} \right\} = 1$$

## Illustration of RP( $\pi$ )



## Updating $\pi$ and $J$

$$\lambda_0 = 1 \Rightarrow \pi = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\text{We have } \pi = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} - A^T \cdot \pi = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1-1+1 \\ 3-1 \\ 1-1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \Rightarrow J = \{3\} \wedge J^c = \{1,2\}$$

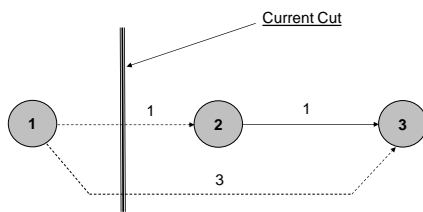
## RP( $\pi$ )

$$\begin{array}{c|cccccc} -1 & 0 & 0 & 0 & -1 & [-1] \\ \hline 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & (1) \end{array} \Rightarrow \begin{array}{c|cccccc} -1 & 0 & 1 & -1 & -1 & 0 \\ \hline 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 \end{array}$$

$$\Rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \bar{\pi} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Leftrightarrow \bar{\pi} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \lambda_0 = \min \left\{ \frac{c_1 - (1,1) \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}}{\bar{\pi}^T \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}}, \frac{c_2 - (1,1) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}}{\bar{\pi}^T \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}} \right\}$$

$$= \min \left\{ \frac{1-0}{1}, \frac{3-1}{1} \right\} = \min \left\{ \frac{1}{1}, \frac{2}{1} \right\} = \min \{1, 2\} = 1$$

## Illustration of RP( $\pi$ )



$$\pi^T = (1, 1) \wedge \bar{\pi}^T = (1, 0)$$

## Updating $\pi$ and $J$

$$\lambda_0 = 1 \Rightarrow \pi = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\text{We have } \pi = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} - A^T \cdot \pi = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1-2+1 \\ 3-2 \\ 1-1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \Rightarrow J = \{1, 3\} \wedge J^c = \{2\}$$

## RP( $\pi$ )

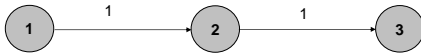
$$\begin{array}{c|cccc|cccc} -1 & 0 & 1 & [-1] & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & (1) & 1 & 0 & \Rightarrow 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{array}$$

$\Rightarrow \xi_0 = 0$  optimal solutions are found, i.e.,

$$x = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \wedge \pi = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \text{ are proven to be optimal for } (P) \text{ and } (D),$$

respectively

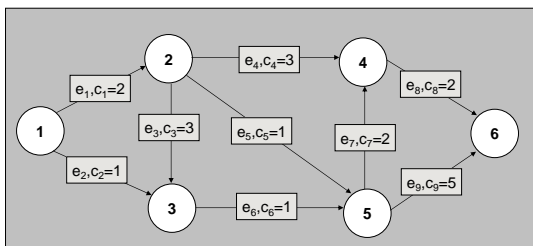
## Illustration of RP( $\pi$ )



$$\pi^T = (2, 1) \wedge \bar{\pi}^T = (0, 0)$$

The shortest path  $\langle 1, 2, 3 \rangle$  has an objective function value of 2.

## A somewhat more complicated example



### Iteration 1 – step 1

We commence our calculations with  $\pi^T = (0,0,0,0,0)$

$$\Rightarrow J = \emptyset \wedge J^c = \{1,2,3,4,5,6,7,8,9\}$$

Consequently, we obtain the following tableau

0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	-1	0	0	-1	1	0
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

### Iteration 1 – step 2

-1	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	-1	0	0	-1	1	0
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

$\Rightarrow$

$$(0,0,0,0,0) = (1,1,1,1,1) - \bar{\pi}^T \Leftrightarrow \bar{\pi}^T = (1,1,1,1,1) \Rightarrow \lambda_0 = \min\{2,5\} = 2$$

$$\Rightarrow \pi^T = (2,2,2,2,2) \Rightarrow J = \{8\} \wedge J^c = \{1,2,3,4,5,6,7,9\}$$

### Iteration 2 – step 1

-1	0	0	0	0	0	0	0	0	0	0	0	0	[-1]	-1
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	-1	0	0	-1	(1)	0
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

### Iteration 2 – step 2

-1	0	0	0	1	0	0	0	0	-1	0	0	-1	0	-1
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	-1	0	0	-1	1	0
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

⇒

$$\bar{\pi}^T = (1, 1, 1, 0, 1) \Rightarrow \lambda_0 = \min \{3, 2, 3\} = 2 \Rightarrow \pi^T = (2, 2, 2, 2, 2) + 2 \cdot (1, 1, 1, 0, 1)$$

$$= (4, 4, 4, 2, 4) \Rightarrow J = \{7, 8\} \wedge J^c = \{1, 2, 3, 4, 5, 6, 9\}$$

### Iteration 3 – step 1

-1	0	0	0	1	0	0	0	0	-1	0	0	[-1]	0	-1
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	-1	0	0	-1	1	0
0	0	0	0	0	1	0	0	0	0	-1	-1	(1)	0	1

### Iteration 3 – step 2

-1	0	0	0	1	1	0	0	0	-1	-1	-1	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	0	-1	-1	-1	0	1	1
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

$$\Rightarrow \bar{\pi}^T = (1, 1, 1, 0, 0) \Rightarrow \lambda_0 = \min \{3 - 2, 1, 1\} = \min \{1, 1, 1\} = 1$$

$$\Rightarrow \pi^T = (4, 4, 4, 2, 4) + 1 \cdot (1, 1, 1, 0, 0) = (5, 5, 5, 2, 4)$$

$$\Rightarrow J = \{4, 5, 6, 7, 8\} \wedge J^c = \{1, 2, 3, 9\}$$

Iteration 4 – step 1

-1	0	0	0	1	1	0	0	0	[-1]	-1	-1	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	(1)	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	0	0	1	1	0	0	0	-1	-1	-1	0	1	1
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

Iteration 4 – step 2

-1	0	1	0	1	1	-1	0	1	0	0	-1	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	1	0	0	0
0	0	1	0	1	1	-1	0	1	0	0	-1	0	1	1
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

Iteration 4 – step 3

-1	0	1	0	1	1	-1	0	1	0	0	[-1]	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	(1)	0	0	0
0	0	1	0	1	1	-1	0	1	0	0	-1	0	1	1
0	0	0	0	0	1	0	0	0	0	-1	-1	1	0	1

### Iteration 4 – step 4

-1	0	1	1	1	1	-1	-1	0	0	0	0	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	(1)	0	0	0
0	0	1	1	1	1	-1	-1	0	0	0	0	0	1	1
0	0	0	1	0	1	0	-1	-1	0	-1	0	1	0	1

$$\Rightarrow \bar{\pi}^T = (1, 0, 0, 0, 0) \Rightarrow \lambda_0 = \min \{2, 1\} = 1$$

$$\Rightarrow \pi^T = (5, 5, 5, 2, 4) + 1 \cdot (1, 0, 0, 0, 0) = (6, 5, 5, 2, 4)$$

$$\Rightarrow J = \{2, 4, 5, 6, 7, 8\} \wedge J^c = \{1, 3, 9\}$$

### Iteration 5 – step 1

-1	0	1	1	1	1	-1	[-1]	0	0	0	0	0	0	0
1	1	0	0	0	0	1	(1)	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	-1	-1	0	0	(1)	0	0	0
0	0	1	1	1	1	-1	-1	0	0	0	0	0	1	1
0	0	0	1	0	1	0	-1	-1	0	-1	0	1	0	1

### Iteration 5 – step 2

0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0	1	1	1	0	0	0	0
1	1	0	1	0	0	1	0	-1	0	0	1	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	1	1
1	1	0	1	0	1	1	0	-1	0	-1	0	1	0	1

$$\Rightarrow \xi_0 = 0 \Rightarrow x^T = (0, 1, 0, 0, 0, 1, 1, 1, 0) \wedge \pi^T = (6, 5, 5, 2, 4) \text{ are}$$

optimal solutions to  $(P)$  and  $(D)$ , respectively.

The shortest path  $\langle 1, 3, 5, 4, 6 \rangle$  has an objective function value of 6.

Dijkstra's Algorithm

```
BEGIN
   $c_{i,j} := \infty, \forall (i,j) \notin E$       The following must hold :  $c_{i,j} \geq 0, \forall (i,j) \in E$ 
   $W := \{s\}; \pi(s) := 0;$       Denote  $s$  as the source of the graph
  FOR all  $y \in V \setminus \{s\}$  DO  $\pi(y) := c_{s,y}$ 
  WHILE  $(W \neq V)$  DO
     $\pi(x) := \min \{ \pi(y) \mid y \notin W \}$ 
     $W := W \cup \{x\}$ 
    FOR all  $y \in V \setminus W$  DO  $\pi(y) := \min \{ \pi(y), \pi(x) + c_{x,y} \}$ 
  END DO
END
```

Laufzeit  $O(n \cdot \log n + m)$

---

---

---

---

---

---

---

---

Full version with storing an optimal path

```
BEGIN
   $c_{ij} := \infty \forall (i,j) \notin E$       The following must hold for this algorithm :  $c_{ij} \geq 0 \forall (i,j) \in E$ 
   $W := \{s\}$       Denote  $s$  as the source of the graph
   $\pi_i := \begin{cases} 0 & \text{if } i = s \\ c_{si} & \text{otherwise} \end{cases} \forall i \in V$       Let  $\pi_i$  be the length of the shortest path  $\langle s, \dots, i \rangle$ 
   $Pre_i := s \forall (s,i) \in E$       Let  $Pre_i$  be the preceding vertex of  $i$  in the shortest path  $\langle s, \dots, Pre_i, i \rangle$ 
  WHILE  $W \neq V$  DO
     $\pi_x := \min \{ \pi_y \mid y \notin W \}$ 
     $W := W \cup \{x\}$ 
    FOR all  $y \in V \setminus W$  DO
      IF  $\pi_x + c_{xy} < \pi_y$  THEN DO
         $\pi_y := \pi_x + c_{xy}$ 
         $Pre_y := x$ 
      END DO
    END DO
  END DO
END
```

---

---

---

---

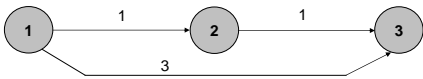
---

---

---

---

Dijkstra's Algorithm and the simple example



$$(c_{i,j}) = \begin{pmatrix} \infty & 1 & 3 \\ \infty & \infty & 1 \\ \infty & \infty & \infty \end{pmatrix}$$

$x$	1	2	3	START
$\pi_x$	0	1	3	$W = \{1\}$
$Pre_x$		1	1	ITERATION 1
$\pi_x$	0	1	2	$W = \{1,2\}$
$Pre_x$		1	2	ITERATION 2
$\pi_x$	0	1	2	$W = \{1,2,3\}$
$Pre_x$		1	2	

$V \setminus W = \emptyset \Rightarrow \text{STOP}$

The shortest path  $\langle 1, 2, 3 \rangle$  has an objective function value of 2.

---

---

---

---

---

---

---

---



## Dijkstra algorithm – running time

- In each step of the procedure a node is determined (labeled) to which a shortest path is found
- Hence, there are  $n-1$  steps for  $n=|V|$  nodes
- Moreover, each arc of set  $E$  in the network has to be considered once
- If all nodes are stored in a min-heap (sorting criterion is the distance to the labeled nodes) we obtain the total asymptotic running time

$$O(|E| + |V| \cdot \log(|V|))$$

## Negative arc weights

- The basic idea of the Dijkstra procedure is based on the fact that if we have identified a node with a minimum distance to the labeled nodes the shortest path to this node is found
- However, this is not necessarily correct if negative arc weights occur
- In this case, a path to another node with even longer length may become shorter over an arc with negative weight
- Note that the Dijkstra algorithm can be extended to the case of negative arc weights. However, this results in an increased time complexity of  $O(n^3)$  (cf., Nemhauser (1972), Bazaraa and Langley (1974))

## Cycles of negative weights

- The shortest path problem in a network may be not well-defined anymore if there exists cycles of negative length
  - In this case, some paths can be arbitrarily shortened by integrating this cycle infinitely often
  - Hence, if there is a connection to this cycle, the problem has no solution and, therefore, is not well-defined

6.2 Bellman-Ford algorithm

- The Bellman-Ford algorithm is based on separate algorithms by Bellman and by Ford (cf. Bellman (1958), Ford and Fulkerson (1962))
  - Like the Dijkstra algorithm, it solves the single source shortest path problem starting from a source node  $s$
  - But, in contrast to the Dijkstra algorithm, it is able to deal with edges that possess a negative weight
  - Moreover, the algorithm of Bellman-Ford also identifies whether a cycle of negative length exists in the graph that is reachable from  $s$
- The algorithm possesses a very simple structure that enables us to easily derive its asymptotic running time
- However, the proving of the correctness of the algorithm becomes quite technical

---

---

---

---

---

---

---

---

Attributes of each vertex  $v$

- $s$  single source from that the shortest paths have to be found
- $d(v)$  shortest path estimate of vertex  $v$
- $\pi(v)$  predecessor node in graph  $G_\pi$  (node that lastly brought an reduction of the estimate of vertex  $v$ )
- $w(u, v)$  weight of arc  $(u, v)$  in network  $G = (V, E)$
- $\delta(v)/\delta(s, v)$  actual length of the shortest path from  $s$  to  $v$
- $\delta(u, v)$  actual length of the shortest path from  $u$  to  $v$

Initialization of the attributes

**procedure** initialization( $G = (V, E), s$ )

$d(s) = 0$

**for each** vertex  $v \in V$

**do**  $d(v) = \infty, \pi(v) = -1$  **od**

---

---

---

---

---

---

---

---

Technique of *relaxation*

- The algorithm of Bellman and Ford iteratively applies the technique of relaxation
- This operation tries to reduce the estimate  $d(v)$  of a node  $v$  by considering a reduction over an arc  $(u, v)$  that connects the estimate  $d(u)$  of node  $u$  to node  $v$

**procedure** relax( $u, v, w$ )

**if**  $d(v) > d(u) + w(u, v)$

**then**  $d(v) = d(u) + w(u, v), \pi(v) = u$

---

---

---

---

---

---

---

---

## Bellman-Ford – pseudo code

**procedure** Bellman-Ford( $G = (V, E), w, s$ )

1. *initialization*( $G = (V, E), s$ )
2. **for**  $i = 1$  to  $|V| - 1$
3.     **for each edge**  $(u, v) \in E$
4.         *relax*( $u, v, w$ )
5. **for each edge**  $(u, v) \in E$
6.     **if**  $d(v) > d(u) + w(u, v)$
7.         **then return** FALSE, stop
8. **return** TRUE

## Predecessor subgraph $G_\pi$

- We often wish to compute not only shortest-path weights, but also the nodes visited on these shortest paths
- For this purpose, for a given graph  $G = (V, E)$ , we introduce a predecessor subgraph  $G_\pi$  as follows
  - For each vertex  $v \in V$ , a predecessor  $\pi(v)$  that is either another vertex or “-1”
  - The Bellman-Ford algorithm introduced in the following will generate a predecessor subgraph  $G_\pi$  such that the chain of predecessors originating at a vertex  $v$  runs backwards along a shortest path from  $s$  to  $v$ .
  - We define the predecessor subgraph  $G_\pi = (V_\pi, E_\pi)$  with
 
$$V_\pi = \{v \in V \mid \pi(v) \neq -1\} \cup \{s\}$$
 and  $E_\pi = \{(\pi(v), v) \in E \mid v \in V_\pi - \{s\}\}$

## Shortest-paths tree

- Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and source node  $s$ .
- A shortest path tree rooted at node  $s$  of  $G$  is a directed subgraph  $G' = (V', E')$  with
  1.  $V' \subseteq V$  and  $E' \subseteq E$
  2.  $V'$  is a set of nodes that are reachable from node  $s$
  3.  $G' = (V', E')$  forms a rooted tree (a tree is a connected graph such that each node possesses an unambiguously defined predecessor) with root node  $s$
  4. For all  $v \in V'$ , the unique simple path from  $s$  to  $v$  in  $G' = (V', E')$  is a shortest path from  $s$  to  $v$  in  $G$

Triangle inequality

6.2.1 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and source node  $s$ . Then, for all edges  $(u, v) \in E$ , we have

$$\delta(s, v) \leq \delta(s, u) + w(u, v)$$

---

---

---

---

---

---

---

---

Proof of Lemma 6.2.1

- Suppose that  $p$  is a shortest path from source  $s$  to vertex  $v$
- Then  $p$  has no more weight than any other path from  $s$  to  $v$
- Specifically, path  $p$  has no more weight than the particular path that takes a shortest path from source  $s$  to vertex  $u$  and then takes edge  $(u, v)$

---

---

---

---

---

---

---

---

Upper bound property

6.2.2 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and source node  $s$ . Moreover, the attributes are initialized by executing the procedure  $\text{initialization}(G = (V, E), w, s)$ . Then,  $d(v) \geq \delta(s, v), \forall v \in V$  and this invariant is maintained over any sequence of relaxation steps on the edges of  $G$ . Furthermore, once  $d(v)$  coincides with  $\delta(s, v)$ , it never changes.

---

---

---

---

---

---

---

---

### Proof of Lemma 6.2.2

- This proof is given by induction over the number  $k$  of performed relaxation steps
- Start of induction with  $k=0$ , i.e., no relaxation step is executed
  - Here, the proposition obviously holds for all  $v \in V - \{s\}$  since we initialized the shortest path estimate by  $d(v) = \infty \geq \delta(s, v) = \delta(v)$
  - Moreover,  $d(s) = 0 \geq \delta(s, s) = \delta(s)$  holds since  $\delta(s, s) = \delta(s) = -\infty$  if  $s$  is on a cycle of negative length and  $\delta(s, s) = \delta(s) = 0$  otherwise
  - Therefore, the proposition holds

### Proof of Lemma 6.2.2

- Induction step  $k \rightarrow k+1$ 
  - We consider the relaxation of an edge  $(u, v)$ . By the inductive proposition we know that, prior to the  $k+1$ th relaxation, it holds that  $d(x) \geq \delta(s, x), \forall x \in V$
  - In this particular relaxation of edge  $(u, v)$  only the estimate  $d(v)$  may be updated
  - If it is not updated we know, by the inductive proposition  $d(v) \geq \delta(s, v)$
  - Otherwise, we have  $d(v) = d(u) + w(u, v)$ 
    - Due to the inductive proposition, we know that  $d(u) \geq \delta(s, u)$  and thus  $d(v) = d(u) + w(u, v) \geq \delta(s, u) + w(u, v)$
    - And due to the triangle property (Lemma 6.2.1), we have  $d(v) = d(u) + w(u, v) \geq \delta(s, u) + w(u, v) \geq \delta(s, v)$

### Proof of Lemma 6.2.2

- In order to see that the value of  $d(v)$  never changed once it coincides with  $\delta(s, v)$ , note that we have just proven that  $d(v) \geq \delta(s, v), \forall v$ , and it cannot increase since the application of the relaxation operation may only reduce the estimate  $d(v)$  but never increase it
- This completes the proof

## No-path property

### 6.2.3 Corollary

Suppose that in a weighted directed graph  $G = (V, E)$  with weight function  $w: E \rightarrow \mathbb{R}$  no path connects a source node  $s$  to a given node  $v$ . Then, after the graph is initialized by calling the procedure  $\text{initialization}(G = (V, E), w, s)$ , we have  $d(v) = \infty$  and this invariant is maintained over any sequence of relaxation steps on the edges of  $G$ .

## Proof of Corollary 6.2.3

- Due to the upper bound property (Lemma 6.2.2), we conclude that
$$\infty = \delta(s, v) \leq d(v) \Rightarrow d(v) = \infty$$

## Simple consequence

### 6.2.4 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and  $(u, v) \in E$ . Then, immediately after relaxing edge  $(u, v) \in E$  by executing the procedure  $\text{relax}(u, v, w)$ , we have  $d(v) \leq d(u) + w(u, v)$ .

### Proof of Lemma 6.2.4

- If, just before relaxing the edge  $(u, v) \in E$ , we have  $d(v) > d(u) + w(u, v)$ , then we have  $d(v) = d(u) + w(u, v)$  afterward
- If, instead, we have  $d(v) \leq d(u) + w(u, v)$  just before relaxing the edge  $(u, v) \in E$ , then no update is conducted and we also obtain  $d(v) \leq d(u) + w(u, v)$  afterward
- This completes the proof

---

---

---

---

---

---

---

---

### Convergence property

#### 6.2.5 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$ , source node  $s \in V$  and two nodes  $u, v \in V$ . Moreover, let  $p$  a shortest path from  $s$  to  $v$ , while the last used arc of  $p$  is  $(u, v) \in E$ . After executing the procedure initialization( $G = (V, E), w, s$ ) and performing a sequence of relaxation steps that includes the call  $\text{relax}(u, v, w)$  is executed on the edges of  $G = (V, E)$ . If  $d(u) = \delta(s, u)$  at any time prior to the call, then  $d(v) = \delta(s, v)$  at all times after the call.

---

---

---

---

---

---

---

---

### Proof of Lemma 6.2.5

- Due to the upper bound property (Lemma 6.2.2), if we obtain  $d(u) = \delta(s, u)$  at some point before calling  $\text{relax}(u, v, w)$ , then this equality holds thereafter. Moreover, after calling  $\text{relax}(u, v, w)$ , due to Lemma 6.2.4, we obtain  $d(v) \leq d(u) + w(u, v) = \delta(s, u) + w(u, v)$
- And due to the definition of  $p$  and the fact that subpaths of a shortest path are also shortest paths (otherwise, the shortest path can be shortened), we conclude  $d(v) \leq d(u) + w(u, v) = \delta(s, u) + w(u, v) = \delta(s, v)$

---

---

---

---

---

---

---

---

### Proof of Lemma 6.2.5

- Again, due to the upper bound property (Lemma 6.2.2), after obtaining  $d(v) = \delta(s, v)$ , this equality is maintained thereafter
- This completes the proof

---

---

---

---

---

---

---

---

### Path-relaxation property

#### 6.2.6 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and a source node  $s \in V$ . Moreover, let  $p = \langle v_0, \dots, v_k \rangle$  any shortest path from  $s = v_0$  to  $v_k$ . After executing the procedure  $\text{initialization}(G = (V, E), w, s)$  and performing a sequence of relaxation steps that includes, in order, the calls  $\text{relax}(v_0, v_1, w)$ ,  $\text{relax}(v_1, v_2, w), \dots, \text{relax}(v_{k-1}, v_k, w)$ , then  $d(v_k) = \delta(s, v_k) = \delta(v_0, v_k)$  after these relaxations and at all times afterward. This property holds no matter what other edge relaxations occur, including relaxations that are intermixed with relaxations of the edges of  $p$ .

---

---

---

---

---

---

---

---

### Proof of Lemma 6.2.6

- This proof is given by induction, i.e., specifically, we show that, after the  $i$ th edge of path  $p$  (i.e., edge  $(v_{i-1}, v_i)$ ) is relaxed, we have  $d(v_i) = \delta(s, v_i) = \delta(v_0, v_i)$
- The basis of the induction is  $i = 0$ 
  - No relaxation of edges of path  $p$  is performed
  - Hence, due to the initialization, we have  $d(v_0) = d(s) = 0 = \delta(s, s) = \delta(s, v_0)$
  - Due to the upper bound property (Lemma 6.2.2), the value of  $d(v_0)$  never changes after the initialization

---

---

---

---

---

---

---

---



Proof of Lemma 6.2.6

- For the inductive step, we assume, by induction, that it holds  $d(v_{i-1}) = \delta(s, v_{i-1}) = \delta(v_0, v_{i-1})$  and we call  $relax(v_{i-1}, v_i, w)$
- Hence, due to the convergence property (Lemma 6.2.5), we conclude  $d(v_i) = \delta(s, v_i) = \delta(v_0, v_i)$  and, again, due to the upper bound property (Lemma 6.2.2), the value of  $d(v_i)$  never changes after this relaxation
- This completes the proof

---

---

---

---

---

---

---

---

Relaxation and shortest-paths trees

- We now show that once a sequence of relaxations has caused the shortest-path estimates to coincide with the shortest-path weights, the predecessor subgraph  $G_\pi$  induced by the resulting values is a shortest-paths tree for  $G$
- We start with the following lemma, which shows that the predecessor subgraph always forms a rooted tree whose root is the source

---

---

---

---

---

---

---

---

Rooted tree with root  $s$

6.2.7 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and a source node  $s \in V$ , while there exists no cycle of negative length that is reachable from node  $s$ . Then, after executing the procedure  $initialization(G = (V, E), w, s)$ , the predecessor subgraph  $G_\pi$  forms a rooted tree with root  $s$ , and any sequence of relaxation steps on edges of  $G$  maintains this property as an invariant.

---

---

---

---

---

---

---

---

## Proof of Lemma 6.2.7

- Initially,  $s$  is the only node in the predecessor subgraph  $G_\pi$  and the proposition holds
- Therefore, we consider the situation after performing a sequence of relaxation steps
- First, we show that  $G_\pi$  is acyclic
  - Suppose by performing the relaxation steps there occurs a first cycle  $c = \langle v_0, \dots, v_k \rangle$  in  $G_\pi$  with  $v_0 = v_k$ . This implies  $\forall i \in \{1, \dots, k\}: \pi(v_i) = v_{i-1}$
  - By renumbering the nodes on the cycle, we can assume, without loss of generality, that this cycle occurs after calling the operation  $relax(v_{k-1}, v_k, w)$
  - Clearly, all nodes  $v_i$  on the cycle are reachable from  $s$  since  $\pi(v_i) \neq -1$  and, therefore, the upper bound property (Lemma 6.2.2) tells us that  $d(v_i)$  is finite and through  $d(v_i) \geq \delta(s, v_i)$ , we have  $\delta(s, v_i) \neq \infty$  and, hence, there is a connection from  $s$  to  $v_i$

## Proof of Lemma 6.2.7

- First, we show that  $G_\pi$  is acyclic (continuation)
  - We consider the situation just before calling the operation  $relax(v_{k-1}, v_k, w)$
  - There, since it holds  $\forall i \in \{1, \dots, k-1\}: \pi(v_i) = v_{i-1}$ , the last update of  $d(v_i)$  was  $d(v_i) = d(v_{i-1}) + w(v_{i-1}, v_i)$  and since then,  $d(v_{i-1})$  was only further decreased, i.e., we have  $d(v_i) \geq \delta(s, v_{i-1}) + w(v_{i-1}, v_i), \forall i \in \{1, \dots, k-1\}$
  - Due to  $\pi(v_k) = v_{k-1}$ , just prior to the update, we have  $d(v_k) > d(v_{k-1}) + w(v_{k-1}, v_k)$  (otherwise, no update would be performed by calling  $relax(v_{k-1}, v_k, w)$ )
  - We calculate the estimates of nodes on cycle  $c$ 

$$\sum_{i=1}^k d(v_i) > \sum_{i=1}^k (d(v_{i-1}) + w(v_{i-1}, v_i)) = \sum_{i=1}^k d(v_{i-1}) + \sum_{i=1}^k w(v_{i-1}, v_i)$$

Since  $v_0 = v_k$ , we have  $\sum_{i=1}^k d(v_i) = \sum_{i=1}^k d(v_{i-1})$  and this implies  $0 > \sum_{i=1}^k w(v_{i-1}, v_i)$
  - Hence, we have a cycle of negative length which provides the desired contradiction
  - Thus, no cycle is possible

## Proof of Lemma 6.2.7

- In order to show that  $G_\pi$  is a rooted tree with root  $s$ , it is sufficient to prove that for all  $v \in V_\pi$  there is a unique single path from  $s$  to  $v$  in  $G_\pi$
- First, we show that there is a path from  $s$  to  $v$  in  $G_\pi$ 
  - Nodes  $v$  in  $G_\pi$  are those with  $\pi(v) \neq -1$  plus the source node  $s$
  - By induction over the number of the relaxation steps  $k$ , we show that a path exists from  $s$  to  $v \in V_\pi$  in  $G_\pi$ 
    - $k = 0$ : Trivial case since the path starts at  $s \in V_\pi$
    - $k > 0$ : We consider the  $k$ th relaxation that relaxes an edge  $(u, v) \in E$  and consider node  $v \in V_\pi$ . If the estimate  $d(v)$  was not reduced the connection results by the proposition of the induction
    - Otherwise, if the estimate  $d(v)$  was reduced, we have a connection over  $\pi(v) = u$  that is connected by the proposition of the induction

## Proof of Lemma 6.2.7

- Finally, we have to show for all  $v \in V_\pi$  that there is a single path from  $s$  to  $v$  in  $G_\pi$
- Let us assume  $G_\pi$  contains two paths from  $s$  to  $v$ 
  - Path 1:  $s \rightsquigarrow u \rightsquigarrow x \rightarrow z \rightsquigarrow v$
  - Path 2:  $s \rightsquigarrow u \rightsquigarrow y \rightarrow z \rightsquigarrow v$
  - With  $x \neq y$  (Note that  $u$  may be  $s$  and/or  $z$  may be  $v$ )
  - But, then  $\pi(z) = x$  and  $\pi(z) = y$  which implies the contradiction that  $x = y$
- All in all, we conclude that for all  $v \in V_\pi$  there is a unique single path from  $s$  to  $v$  in  $G_\pi$  and, therefore, predecessor subgraph  $G_\pi$  forms a rooted tree with root  $s$

## Predecessor-subgraph property

### 6.2.8 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and a source node  $s \in V$ , while there exists no cycle of negative length that is reachable from node  $s$ . Then, after calling the procedure  $\text{initialization}(G = (V, E), w, s)$  any sequence of relaxation steps on edges of  $G = (V, E)$  is executed that produces for all  $v \in V$   $d(v) = \delta(s, v)$ . Then, the predecessor subgraph  $G_\pi$  is a shortest path tree rooted at  $s$ .

## Proof of Lemma 6.2.8

- In what follows, it is shown that the four attributes of shortest path trees are fulfilled by the predecessor subgraph  $G_\pi$
- These are the following

A shortest path tree rooted at node  $s$  of  $G$  is a directed subgraph  $G' = (V', E')$  if

- $V' \subseteq V$  and  $E' \subseteq E$
- $V'$  is a set of nodes that are reachable from node  $s$
- $G' = (V', E')$  forms a rooted tree (a tree is a connected graph such that each node possesses an unambiguously defined predecessor) with root node  $s$
- For all  $v \in V'$ , the unique simple path from  $s$  to  $v$  in  $G' = (V', E')$  is a shortest path from  $s$  to  $v$  in  $G$

- Is trivial
- If a node  $v$  is reachable from  $s$  we have  $\delta(s, v) \neq \infty$ . Therefore, if  $v \in V_\pi$  we have  $\pi(v) \neq -1$  and  $d(v) \neq \infty$ . Due to  $d(v) \geq \delta(s, v)$ , we know that  $\delta(s, v) \neq \infty$  and node  $v$  is reachable from  $s$
- Follows directly from Lemma 6.2.7

## Proof of Lemma 6.2.8

4. Let  $p = \langle v_0, \dots, v_k \rangle$  the unique path in  $G_\pi$  with  $v_0 = s$  and  $v_k = v$ . This implies  $\forall i \in \{1, \dots, k\}: \pi(v_i) = v_{i-1}$ ,  $d(v_i) \geq d(v_{i-1}) + w(v_{i-1}, v_i)$  and (by proposition)  $d(v_i) = \delta(s, v_i)$ . Hence, we obtain  $\delta(s, v_i) \geq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \Rightarrow \delta(s, v_i) - \delta(s, v_{i-1}) \geq w(v_{i-1}, v_i)$ . By summing the weights along the path  $p$  we get

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i) \leq \sum_{i=1}^k (\delta(s, v_i) - \delta(s, v_{i-1})) = \delta(s, v_k) - \underbrace{\delta(s, v_0)}_{=\delta(s, s)=0} = \delta(s, v_k)$$

Thus, we have  $w(p) \leq \delta(s, v_k) = \delta(s, v)$  and since  $\delta(s, v)$  is the length of the shortest path, we conclude  $w(p) = \delta(s, v)$ , and thus  $p$  is a shortest path from  $s$  to  $v$  in  $G$

This completes the proof

## Correctness of the found estimates

### 6.2.9 Lemma

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and a source node  $s \in V$ , while there exists no cycle of negative length that is reachable from node  $s$ . Then, after calling the procedure  $\text{initialization}(G = (V, E), w, s)$  and  $|V| - 1$  iterations of the for loop of lines 2-4 of the Bellman-Ford algorithm, we have  $d(v) = \delta(s, v) \forall v \in V$  with  $v$  is reachable from  $s$ .

## Proof of Lemma 6.2.9

- We apply the path-relaxation property (Lemma 6.2.6). For this purpose, consider any node  $v$  that is reachable from  $s$  and  $p = \langle v_0, \dots, v_k \rangle$  any shortest path from  $s = v_0$  to  $v_k = v$ .
- Clearly,  $p$  has at most  $|V| - 1$  edges, and so we have  $k \leq |V| - 1$ . Each of the  $|V| - 1$  iterations of the for loop of lines 2-4 relaxes all  $|E|$  edges. Among the edges relaxed in the  $i$ th iteration, for  $i = 1, 2, \dots, k$  is  $(v_{i-1}, v_i)$ .
- By applying the path-relaxation property (Lemma 6.2.6), we conclude  $d(v) = d(v_k) = \delta(v_0, v_k) = \delta(s, v_k) = \delta(s, v)$
- This completes the proof

Identifying cycles of negative length

6.2.10 Corollary

Let  $G = (V, E)$  be a weighted directed graph with weight function  $w: E \rightarrow \mathbb{R}$  and a source node  $s \in V$ , while there exists no cycle of negative length that is reachable from node  $s$ . Then,  $\forall v \in V$  there is a path from  $s$  to  $v$  if and only if the Bellman-Ford algorithm terminates with  $d(v) < \infty$  when it is run on  $G$ .

---

---

---

---

---

---

---

---

Proof of Corollary 6.2.10

- First, we assume that there is a path from  $s$  to  $v$
- Then, there exists a shortest path  $p = \langle v_0, \dots, v_k \rangle$  from  $s = v_0$  to  $v_k = v$
- Hence, by Lemma 6.2.9, we have  $d(v) = \delta(s, v) < \infty$
  
- Second, we assume that there is no path from  $s$  to  $v$
- Therefore, by Lemma 6.2.3, we have  $d(v) = \infty = \delta(s, v)$
  
- This completes the proof

---

---

---

---

---

---

---

---

Correctness of the Bellman-Ford algorithm

6.2.11 Theorem

Let the Bellman-Ford algorithm run on a weighted, directed graph  $G = (V, E)$  with weight function  $w: E \rightarrow \mathbb{R}$  and a source node  $s \in V$ . If  $G = (V, E)$  contains no cycle of negative length that is reachable from node  $s$ , then the algorithm returns TRUE, we have  $d(v) = \delta(s, v) \forall v \in V$ , and the predecessor subgraph  $G_\pi$  is a shortest path tree rooted at  $s$ . If  $G$  does contain a negative-weight cycle reachable from  $s$ , then the algorithm returns FALSE.

---

---

---

---

---

---

---

---

## Proof of Theorem 6.2.11

- First, we assume that  $G$  does not contain a cycle of negative length that is reachable from  $s$ 
  - If node  $v$  is reachable from  $s$ , then the proposition  $d(v) = \delta(s, v) \forall v \in V$  results from Lemma 6.2.9
  - If node  $v$  is not reachable from  $s$ , then the proposition  $d(v) = \delta(s, v) = \infty$  results from applying Corollary 6.2.3
  - Moreover, the predecessor-subgraph property (Lemma 6.2.8) proves that the predecessor subgraph  $G_\pi$  is a shortest path tree rooted at  $s$ .
  - It remains to show that the TRUE/FALSE output is correct
  - At termination, we have for all edges  $(u, v) \in E$

$$d(v) = \delta(s, v) \leq \underbrace{\delta(s, u) + w(u, v)}_{\text{by the triangle inequality}} = d(u) + w(u, v)$$

## Proof of Theorem 6.2.11

We consider the lines 5-8 of the Bellman-Ford algorithms

5. **for each edge**  $(u, v) \in E$
6.     **if**  $d(v) > d(u) + w(u, v)$
7.         **then return FALSE, stop**
8.     **return TRUE**

- Hence, none of the tests in line 6 causes the algorithm to return FALSE. Therefore, it returns TRUE
- Second, if there is a cycle of negative length in graph  $G$  that is reachable from the source  $s$
- Let the cycle be  $c = (v_0, \dots, v_k)$  with  $v_0 = v_k$ . Then, it holds

$$\sum_{i=1}^k w(v_{i-1}, v_i) < 0$$

## Proof of Theorem 6.2.11

- We assume that the Bellman-Ford algorithm returns TRUE
- Thus, since we have not return FALSE, it holds that

$$d(v_i) \leq d(v_{i-1}) + w(v_{i-1}, v_i), \forall i = 1, 2, \dots, k$$

- Summing the inequalities around cycle  $c$  results in

$$\begin{aligned} \sum_{i=1}^k d(v_i) &\leq \sum_{i=1}^k (d(v_{i-1}) + w(v_{i-1}, v_i)) = \sum_{i=1}^k d(v_{i-1}) + \sum_{i=1}^k w(v_{i-1}, v_i) \\ &= \underbrace{\sum_{i=1}^k d(v_i)}_{\text{Since } v_k = v_0} + \sum_{i=1}^k w(v_{i-1}, v_i) \Leftrightarrow 0 \leq \sum_{i=1}^k w(v_{i-1}, v_i) \end{aligned}$$

- This is a contradiction to the assumption of the negative length of cycle  $c$
- Therefore, the algorithm provides the correct output FALSE if there is a cycle of negative length in graph  $G$  that is reachable from the source  $s$

Complexity

- The initialization step (line 1) possesses an asymptotic running time of  $O(|V|)$
- Each of the  $|V| - 1$  passes over the edges (lines 2-4) requires an asymptotic running time of  $O(|E|)$
- The final for loop of lines 5-7 takes asymptotic running time of  $O(|E|)$
- Hence, all in all, we have a total asymptotic running time of  $O(|V| \cdot |E|)$

---

---

---

---

---

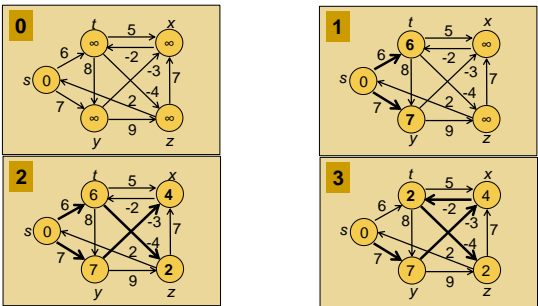
---

---

---

Example

If edge  $(u,v) \in E$  is printed in bold it holds that  $\pi(v)=u$  and  $\pi(v)=-1$ , otherwise  
The sequence of edges is given by  $((t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y))$



---

---

---

---

---

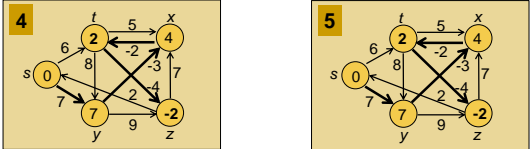
---

---

---

Example

If edge  $(u,v) \in E$  is printed in bold it holds that  $\pi(v)=u$  and  $\pi(v)=-1$ , otherwise  
The sequence of edges is given by  $((t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y))$



Node	$\pi$	d	Path
s	-1	0	s
t	x	2	s-y-x-t
y	s	7	s-y
x	y	4	s-y-x
z	t	-2	s-y-x-t-z

---

---

---

---

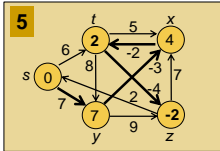
---

---

---

---

Example



Node	$\pi$	d	Path
s	-1	0	s
t	x	2	s-y-x-t
y	s	7	s-y
x	y	4	s-y-x
z	t	-2	s-y-x-t-z

- 5. for each edge  $(u, v) \in E$
- 6.     if  $d(v) > d(u) + w(u, v)$
- 7.         then return FALSE, stop
- 8.     return TRUE

Since  $d(v) > d(u) + w(u, v)$  does not apply for any edge  $(u, v) \in E$ , the Bellman-Ford algorithm returns TRUE in this example

---

---

---

---

---

---

---

---

6.3 Floyd-Warshall algorithm

- In what follows, we introduce a second shortest path algorithm that computes the shortest path between all pairs of nodes in a network
- Therefore, this algorithm is frequently denoted as the “all pairs shortest path” procedure
- In contrast to the Dijkstra algorithm, it works with negative arc weights
- Moreover, the algorithm can be extended in order to deal with cycles of negative length
- The running time of this procedure is  $O(n^3)$

---

---

---

---

---

---

---

---

Triangle operation

6.3.1 Definition

We consider a quadratic distance matrix  $d_{i,j}$ . A triangle operation for a fixed node  $k$  is

$$d_{i,j} = \min \{ d_{i,j}, d_{i,k} + d_{k,j} \} \quad \forall i, k = 1, \dots, n \text{ but } i, k \neq j.$$

This includes  $i = j$ .

- This operation provides the basic idea of the algorithm  
For each relation it is iteratively tested whether a length reduction over an intermediate node  $k$  is possible or not

---

---

---

---

---

---

---

---



Iterative application of the triangle operation

6.3.2 Theorem

We initialize  $d_{i,j}$  with  $c_{i,j}$  and set  $d_{i,i} = 0$ .

By iteratively performing the triangle operation defined in Definition 6.2.1 for successive values  $k=1,2,\dots,n$ ,  $d_{i,j}$  becomes equal to the length of the shortest path from  $i$  to  $j$  according to the arc weights  $[c_{i,j}]$ .

The arc weights may be negative, but we assume that the input graph contains no negative-weight cycles.

---

---

---

---

---

---

---

---

Proof of Theorem 6.3.2

- This proof is given by induction over the index of the executed iteration  $k_0=0,1,\dots,n$
- Specifically, we claim that after the execution of the triangle operation for  $k_0$  the entry  $d_{i,j}$  gives the length of the shortest path from  $i$  to  $j$  with intermediate nodes  $v \leq k_0$
- Initial step of the induction
  - We commence the induction for  $k_0=0$
  - Therefore, the initialization of  $d_{i,j}$  fulfills this invariant for  $k_0=0$  since it coincides with the respective weight of a potentially existing direct connection

---

---

---

---

---

---

---

---

Proof of Theorem 6.3.2

Induction step  $k_0-1 \rightarrow k_0$

- We assume that the proposition holds for  $k_0-1 \geq 0$  and consider  $d_{i,j}$
- There are two possibilities

Case 1: The shortest path from  $i$  to  $j$  includes a visit of node  $k_0$

  - Therefore, the length of the shortest path from  $i$  to  $j$  that includes only intermediate locations with an index  $v \leq k_0$  coincides with the length of the shortest path from  $i$  to  $k_0$  (integrating only locations with an index  $v < k_0$ ) plus the length of the shortest path from  $k_0$  to  $j$  (integrating only locations with an index  $v < k_0$ )
  - This is just the current sum  $d_{i,k_0} + d_{k_0,j}$

---

---

---

---

---

---

---

---

## Proof of Theorem 6.3.2

Case 2: The shortest path from  $i$  to  $j$  does not include a visit of node  $k_0$

- In that case the length of the shortest path from  $i$  to  $j$  that includes only intermediate locations with an index  $v \leq k_0$  coincides with the length of the shortest path from  $i$  to  $j$  (integrating only locations with an index  $v < k_0$ )
- This is just the current value  $d_{i,j}$

Hence, in both cases, the triangle operation executed with node  $k_0$  updates  $d_{i,j}$  such that it defines the length of the shortest path from  $i$  to  $j$  (integrating only locations with an index  $v \leq k_0$ ). This completes the proof. Note that this includes negative arc weights if there is no cycle of negative length.

## Floyd-Warshall algorithm

**Input:** An  $n \times n$ -matrix  $[c_{i,j}]$  with nonnegative entries

**Output:** An  $n \times n$ -matrix  $[d_{i,j}]$  with  $d_{i,j}$  as the shortest distance from  $i$  to  $j$  according to the  $n \times n$ -matrix  $[c_{i,j}]$ ,  $e_{i,j}$  gives the vertex that is intermediately visited (reduction was possible)

```

for all  $i \neq j$  do  $d_{i,j} = c_{i,j}$ ,  $e_{i,j} = 0$ 
for  $i = 1, \dots, n$  do  $d_{i,i} = 0$ ,  $e_{i,i} = 0$ 
for  $k = 1, \dots, n$  do
    for  $i = 1, \dots, n$ ,  $i \neq k$  do
        for  $j = 1, \dots, n$ ,  $k \neq j$  do
            if  $d_{i,j} > d_{i,k} + d_{k,j}$ 
            then begin
                 $d_{i,j} = d_{i,k} + d_{k,j}$ 
                 $e_{i,j} = k$ 
            end
        end
    end
end
    
```

## Path reconstruction

- Based on the found reductions over a vertex  $k$  that is stored in  $e_{i,j}$ , we can backtrack the shortest path
- Specifically, if it holds that  $e_{i,j} = k$ , we know that the path arises by concatenating the paths from node  $i$  to node  $k$  and from node  $k$  to node  $j$
- However, if it holds that  $e_{i,j} = 0$ , the path from node  $i$  to node  $j$  is a direct path and does not include any intermediate vertices

Dealing with cycles of negative length

- As mentioned above, the results of Theorem 6.3.2 also apply if we allow some arc weights in the  $n \times n$ -matrix  $[c_{i,j}]$  to become negative as long as there is no cycle of negative length
- However, if there exists such a negative-length cycle, during the calculation of the Floyd-Warshall algorithm, it will cause some  $d_{h,h}$  to become negative
  - We consider  $h$  as the highest-numbered node on the existing cycle, while  $k$  is the second highest-numbered node on this cycle
  - Therefore, in the iteration that considers an improvement over the intermediate node  $k$ , the length of this cycle can be computed by  $d_{h,h} = d_{h,k} + d_{k,h} < 0$
  - Hence, after this iteration, the entry  $d_{h,h}$  is negative and the algorithm terminates since the shortest path is not defined

---

---

---

---

---

---

---

---

Complexity

- By analyzing the pseudo code of the complete Floyd-Warshall algorithm, all the loops are of fixed length, and the algorithm requires a total of  $n \cdot (n - 1)^2$  comparisons
- Hence, we obtain a total complexity of  $O(n^3)$

---

---

---

---

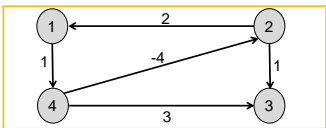
---

---

---

---

Example – Initialization



Initialization of the matrices

	$d_{ij}$					$e_{ij}$			
	1	2	3	4		1	2	3	4
1	$\infty$	$\infty$	$\infty$	1	1	0	0	0	0
2	2	$\infty$	1	$\infty$	2	0	0	0	0
3	$\infty$	$\infty$	$\infty$	$\infty$	3	0	0	0	0
4	$\infty$	-4	3	$\infty$	4	0	0	0	0

---

---

---

---

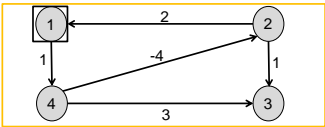
---

---

---

---

Example – first iteration



Iteration  $k=1$

	$d_{ij}$			
	1	2	3	4
1	$\infty$	$\infty$	$\infty$	1
2	2	$\infty$	1	③
3	$\infty$	$\infty$	$\infty$	$\infty$
4	$\infty$	-4	3	$\infty$

	$e_{ij}$			
	1	2	3	4
1	0	0	0	0
2	0	0	0	①
3	0	0	0	0
4	0	0	0	0

---

---

---

---

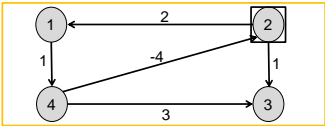
---

---

---

---

Example – second iteration



Iteration  $k=2$

	$d_{ij}$			
	1	2	3	4
1	$\infty$	$\infty$	$\infty$	1
2	2	$\infty$	1	3
3	$\infty$	$\infty$	$\infty$	$\infty$
4	②	-4	③	①

	$e_{ij}$			
	1	2	3	4
1	0	0	0	0
2	0	0	0	1
3	0	0	0	0
4	②	0	②	②

---

---

---

---

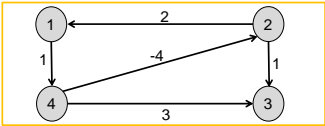
---

---

---

---

Example – Final results



	$d_{ij}$			
	1	2	3	4
1	$\infty$	$\infty$	$\infty$	1
2	2	$\infty$	1	3
3	$\infty$	$\infty$	$\infty$	$\infty$
4	-2	-4	-3	①

	$e_{ij}$			
	1	2	3	4
1	0	0	0	0
2	0	0	0	1
3	0	0	0	0
4	2	0	2	2

Cycle of negative length (4-2-1-4) is found and the algorithm terminates

---

---

---

---

---

---

---

---

Additional literature to Section 6

- Bazaraa, M.S.; Langley, R.W. (1974): A Dual Shortest Path Algorithm. SIAM Journal of Applied Mathematics Vol. 26(3) pp. 496-501.
- Bellman, R. (1958): On a routing problem. Quarterly of Applied Mathematics, 16(1) pp.87-90.
- Ford, L.R. Jr.; Fulkerson, D.R. (1962): Flows in Networks. Princeton University Press.
- Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. (2009): Introduction to Algorithms. 3rd edition. The MIT Press.
- Dijkstra, E.W. (1959): A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1, pp. 269-271.

---

---

---

---

---

---

---

---

Additional literature to Section 6

- Floyd, R.W. (1962): Algorithm 97: Shortest Path. Communications of the ACM Vol.5(6), p.345.
- Nemhauser, G. L. (1972): A Generalized Permanent Label Setting algorithm for the Shortest Path between Specified Nodes. Journal of Mathematical Analysis and Applications, Vol. 38, pp. 328-334.
- Warshall, S. (1962): A Theorem on Boolean Matrices. Journal of the ACM, Vol.9(1), pp.11-12.

---

---

---

---

---

---

---

---