Shortest Path Algorithms

Single source shortest path algorithm

(Greedy Method)

All pair shortest path algorithm

(Dynamic Programming)

Dijkstra's algorithm

Bellman-Ford algorithm

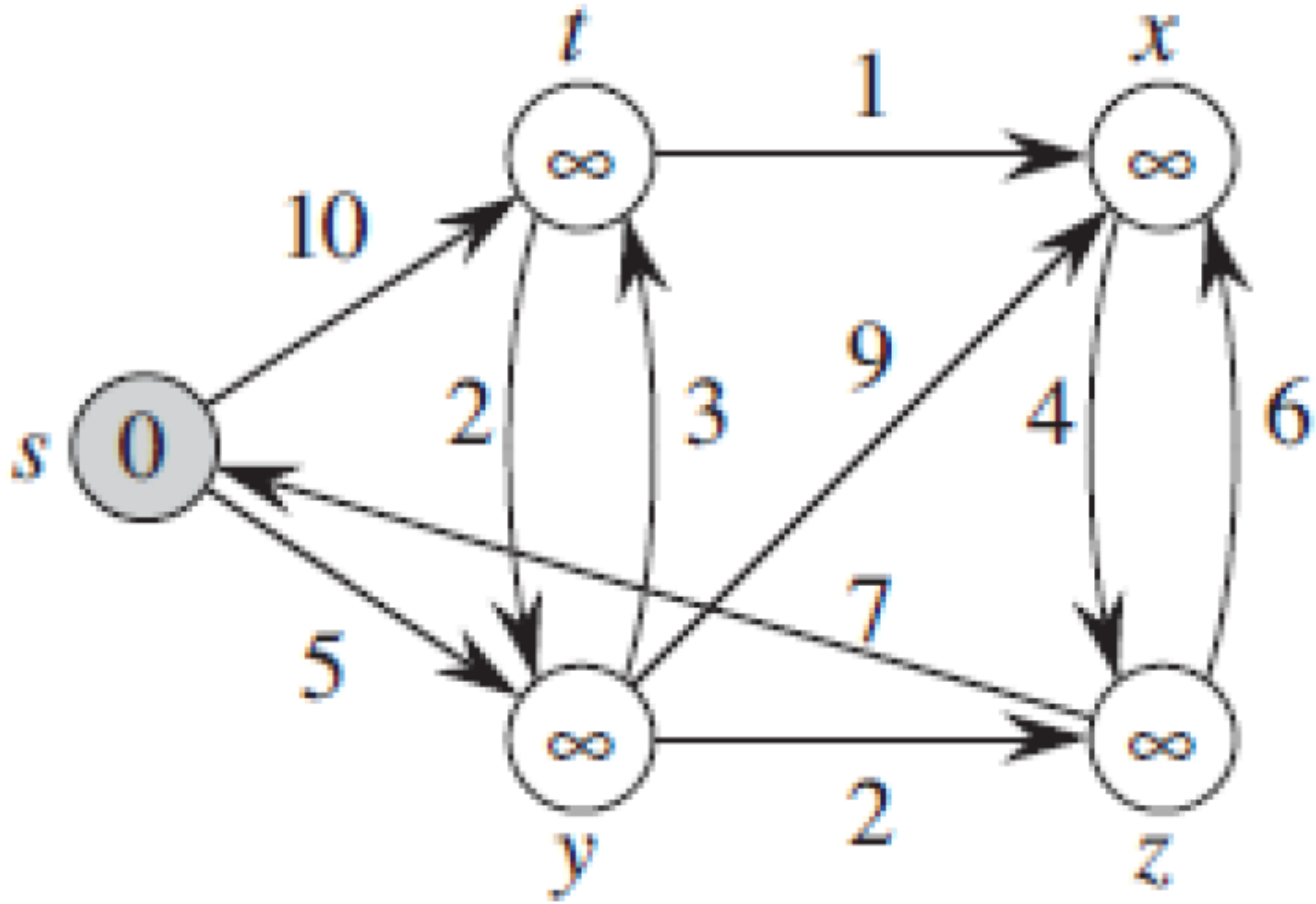
DAG Algorithm

Floyd's Warshall's algorithm

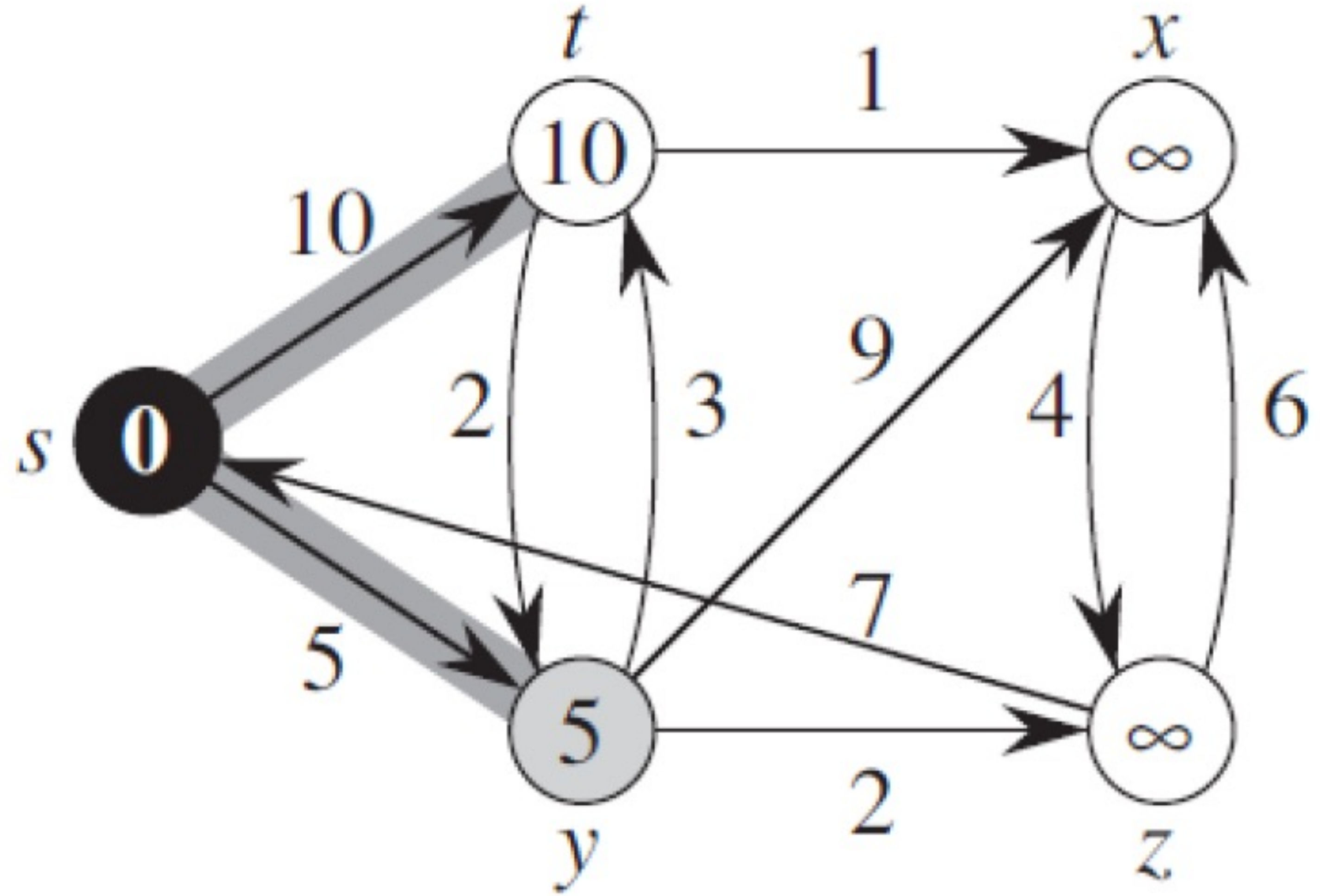
Dijkstra's Algorithm

- Can be used only when the graph do not have a negative weight cycle
- Minimization problem (optimization problem)
- Uses greedy approach
- Works on both directed and Undirected graphs

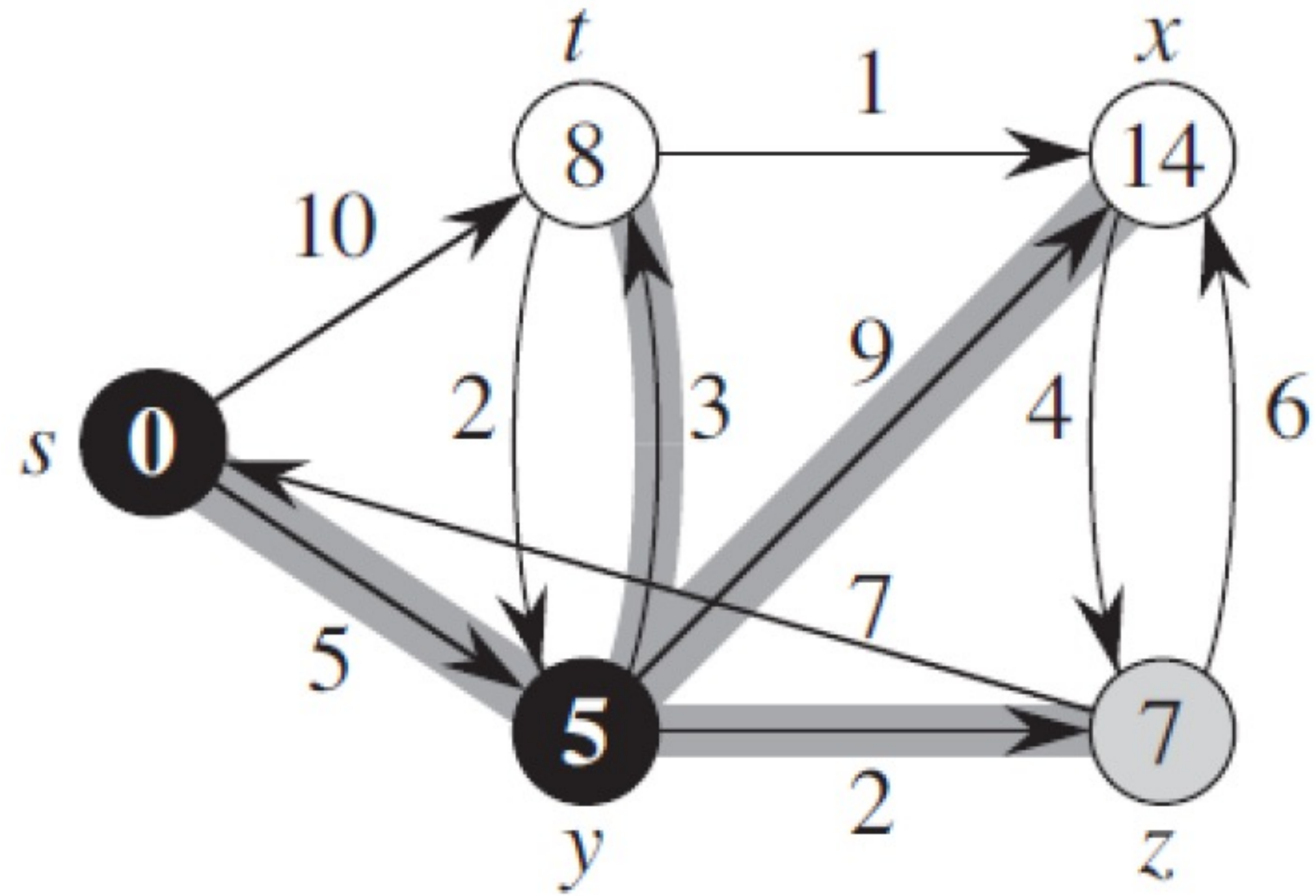
Dijkstra’s algorithm - Example



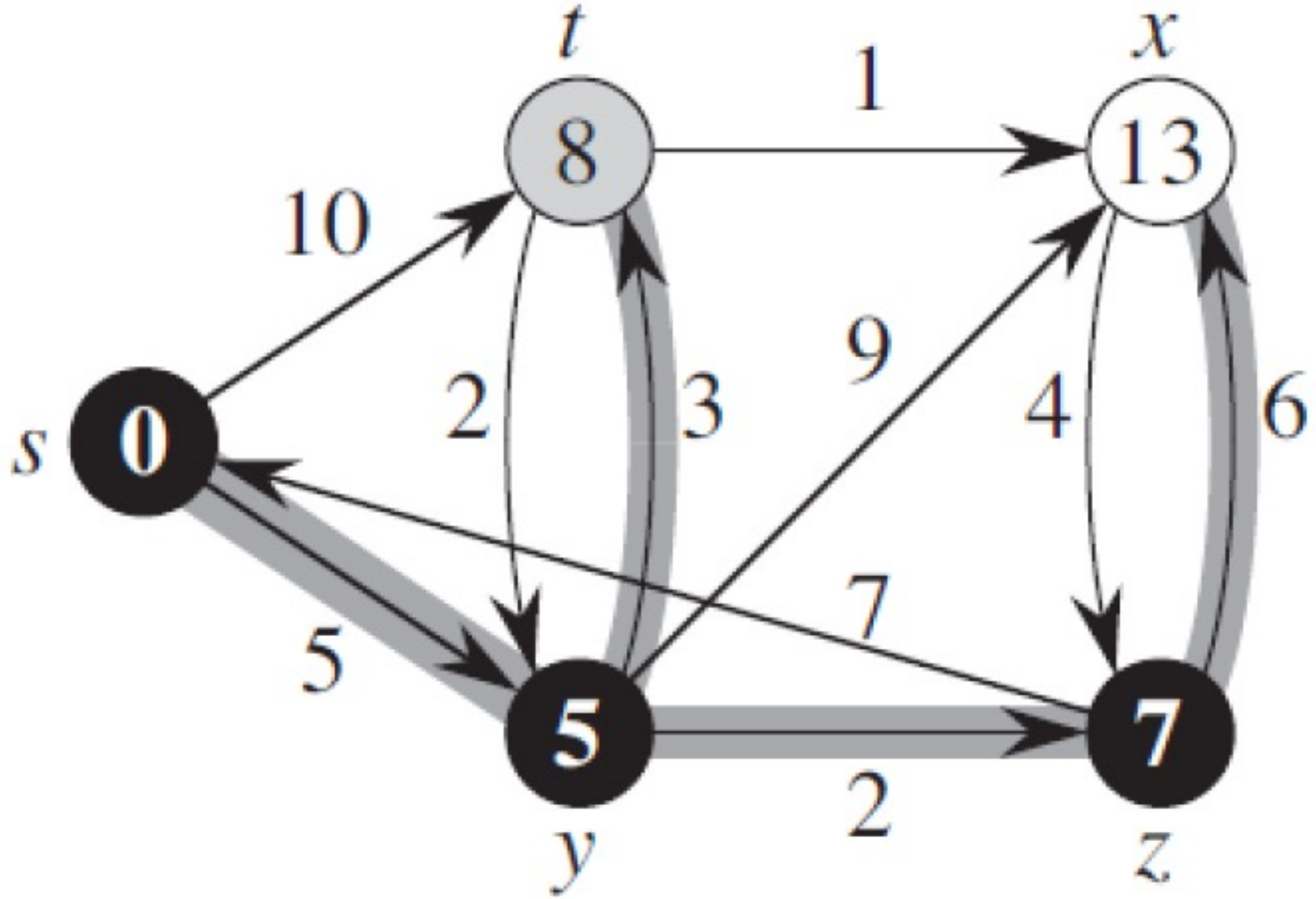
Dijkstra’s algorithm - Example



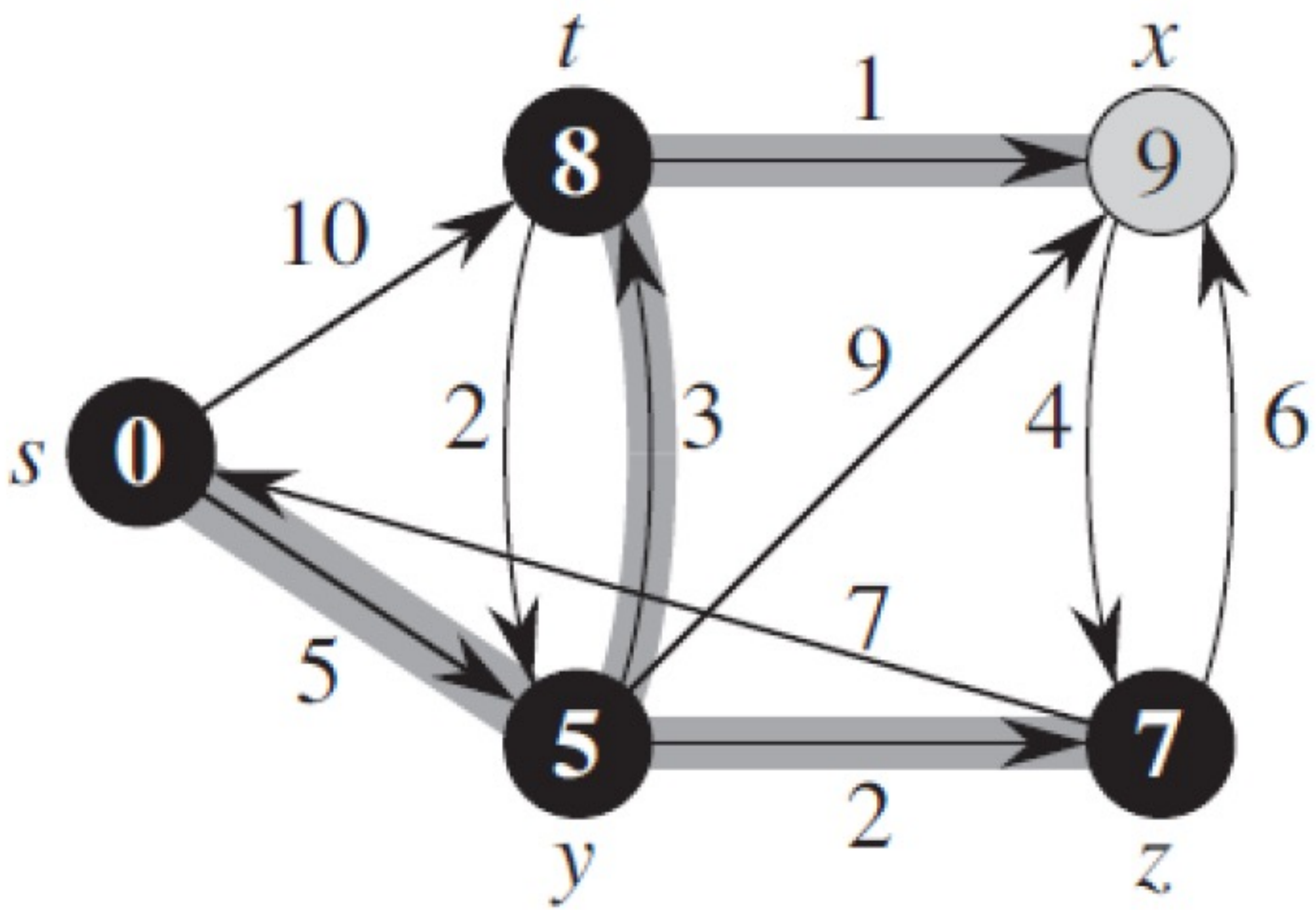
Dijkstra's algorithm - Example



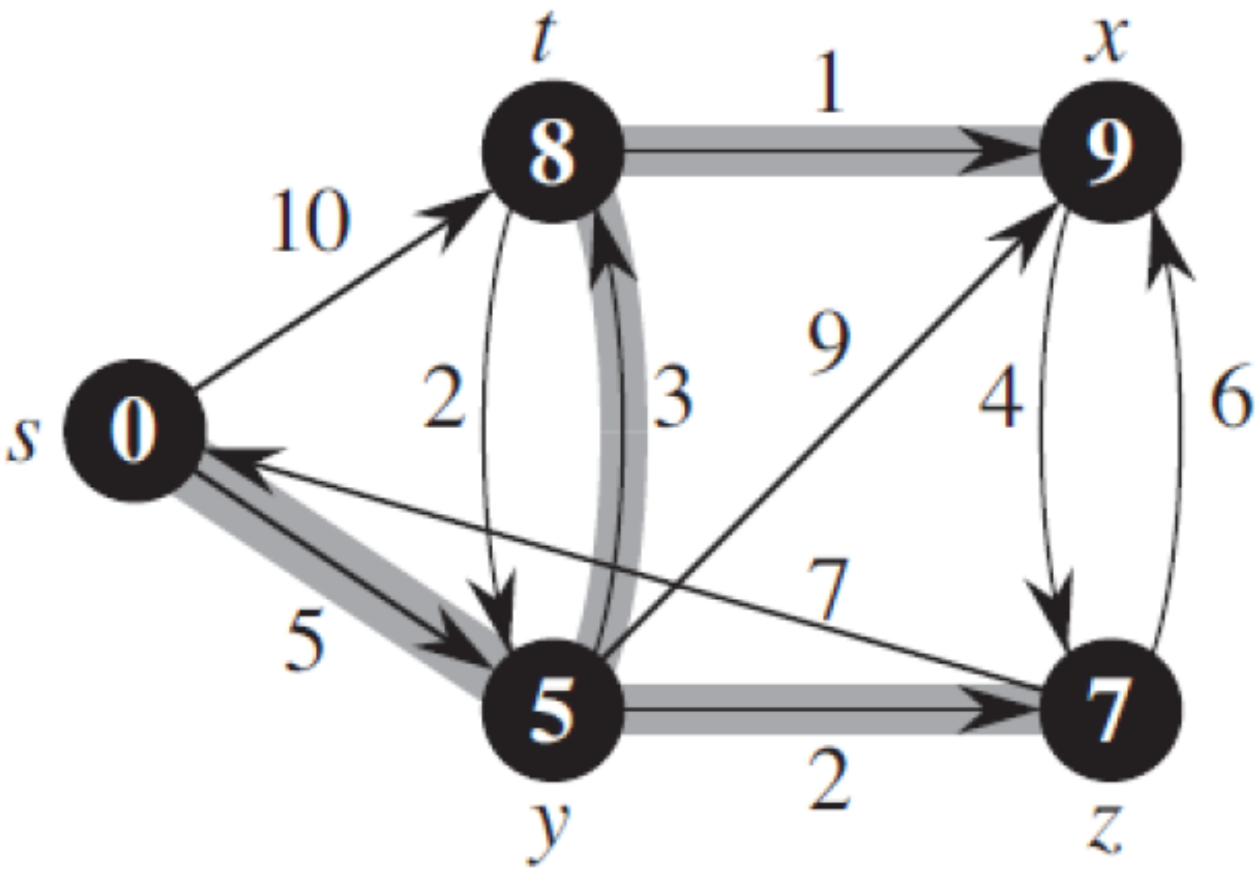
Dijkstra’s algorithm - Example



Dijkstra's algorithm - Example



Dijkstra’s algorithm - Example

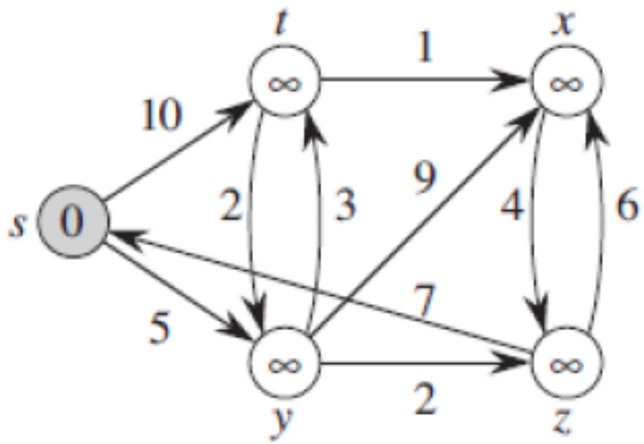


Dijkstra's algorithm

DIJKSTRA(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)
$S = \emptyset$
$Q = G.V$
while $Q \neq \emptyset$
    $u =$ EXTRACT-MIN(Q)
    $S = S \cup \{u\}$
    for each vertex $v \in G.Adj[u]$
        RELAX(u, v, w)

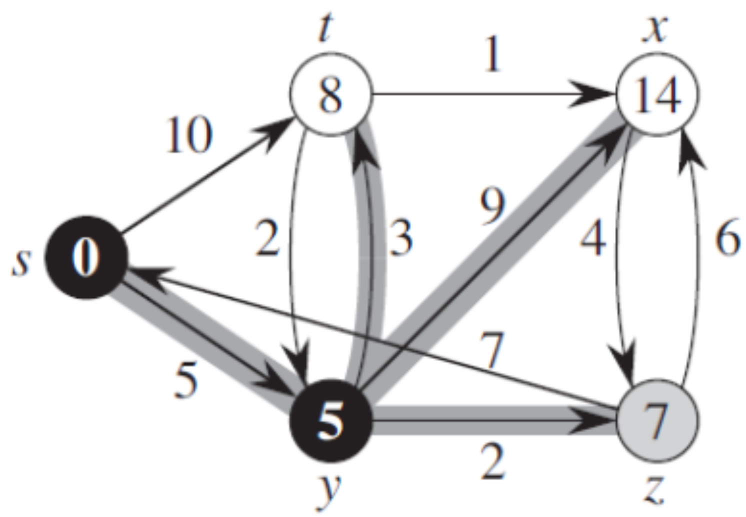
Time Complexity: O(|E| log |v|)



Starting vetrex ‘s’

Step 1:

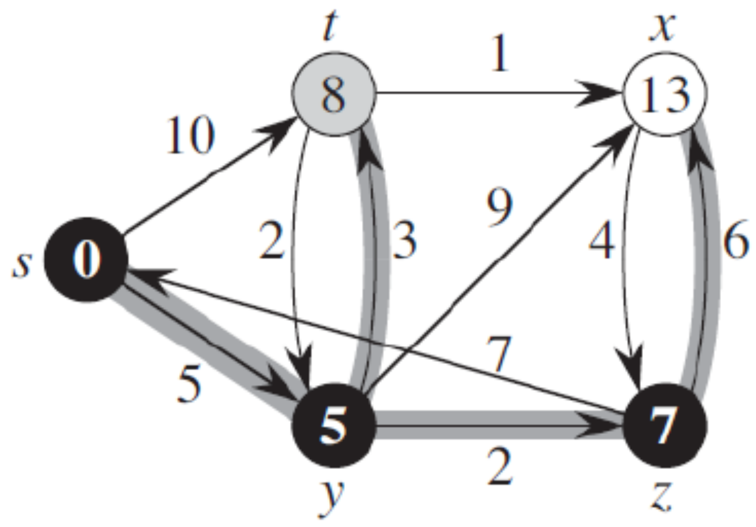
Selected vertex	t	y	x	z
s	10	5	∞	∞



(c)

Step 2: Select the vertex with minimum cost which is ‘y’

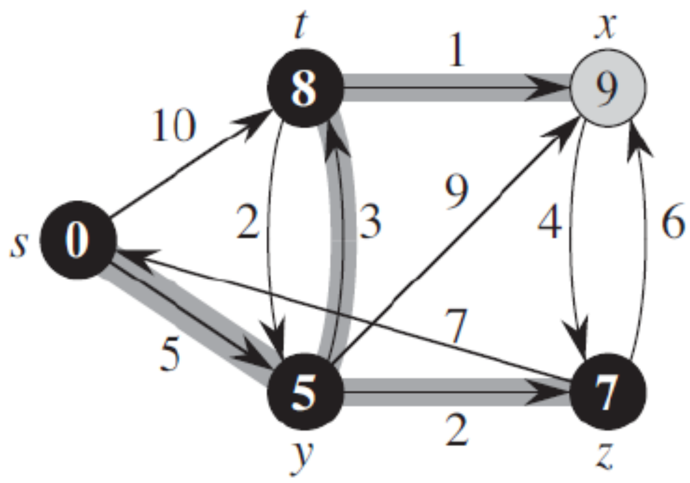
Selected vertex	t	y	x	z
s	10	5	∞	∞
y	8	**5**	14	7



(d)

Step 3: Select the vertex with minimum cost which is ‘z’

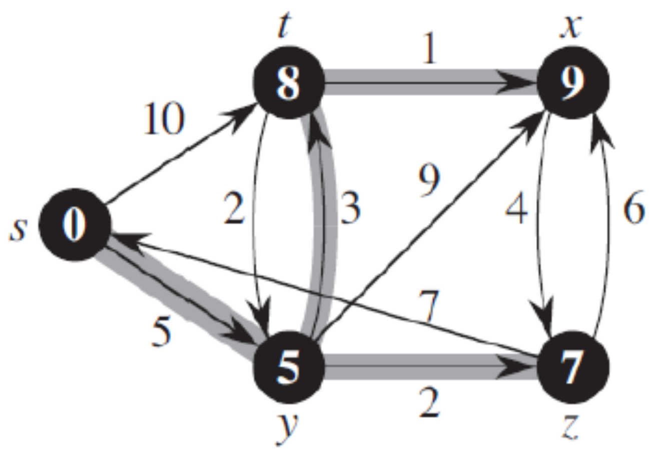
Selected vertex	t	y	x	z
s	10	5	∞	∞
y	8	**<u>5</u>**	14	7
z	8	**<u>5</u>**	13	**<u>7</u>**



(e)

Step 3: Select the vertex with minimum cost which is ‘t’

Selected vertex	t	y	x	z
s	10	5	∞	∞
y	8	**<u>5</u>**	14	7
z	8	**<u>5</u>**	13	**<u>7</u>**
t	**<u>8</u>**	**<u>5</u>**	9	**<u>7</u>**

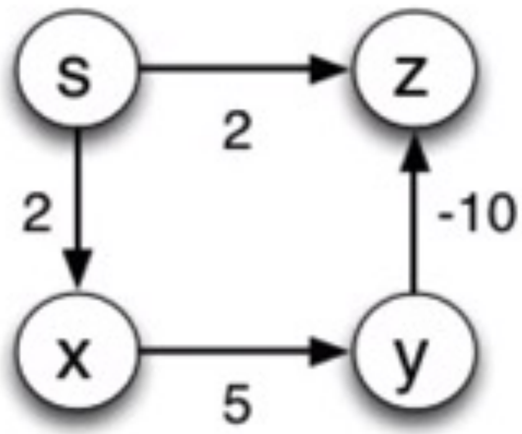


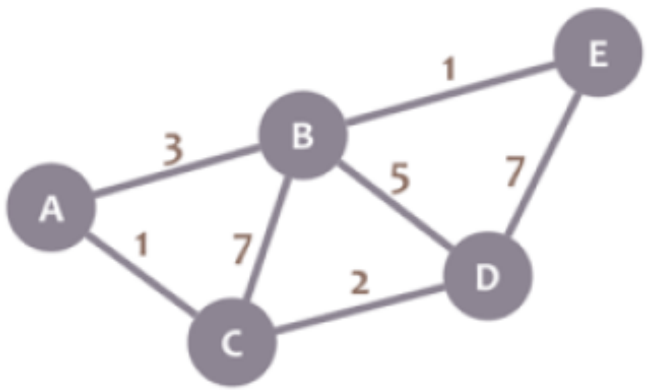
(f)

Step 3: Select the vertex with minimum cost which is ‘x’

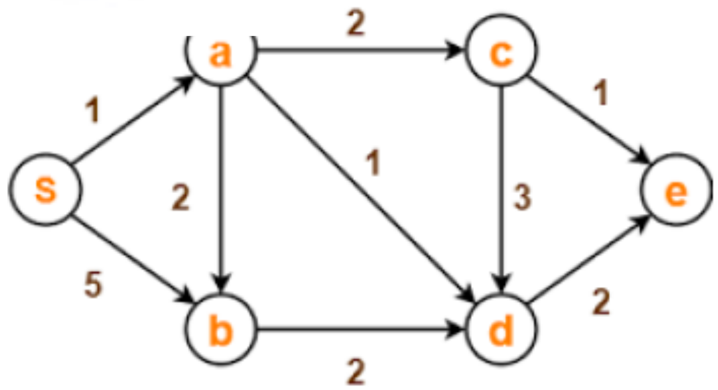
Selected vertex	t	y	x	z
s	10	5	∞	∞
y	8	**5**	14	7
z	8	**5**	13	**7**
t	**8**	**5**	9	**7**
x	**8**	**5**	**9**	**7**

Try yourself...!





Graph 1: source A



Graph 2: Source S

Bellman Ford Algorithm

•Single source shortest path algorithm.

•Bellman Ford algorithm works for negative weighted graphs.

•Principle of this algorithm is, go on relaxing all the edges (n-1) times where ‘n’ is number of edges.

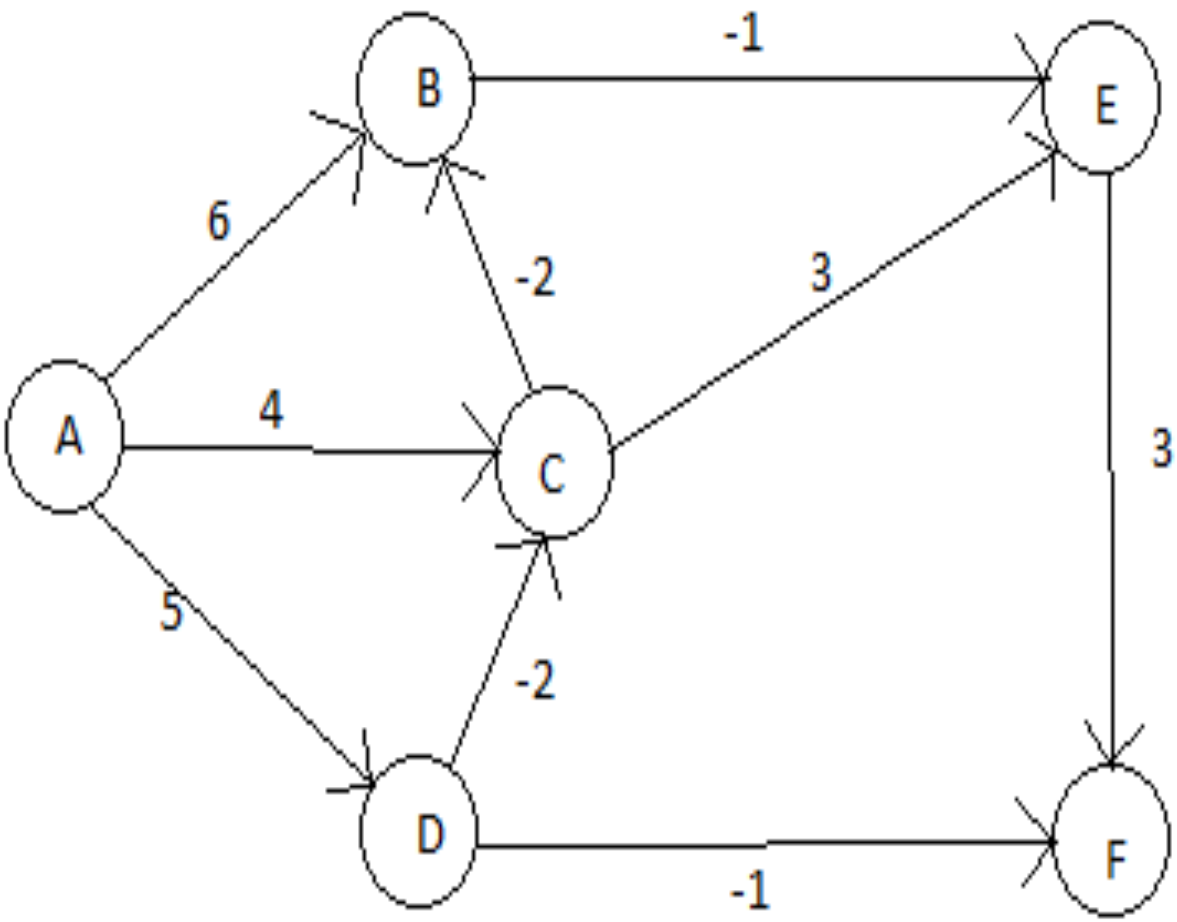
•**Relaxation** – updating process

If (d[u] + c(u,v) < d[v])
    d[v] = d[u] + c(u,v)

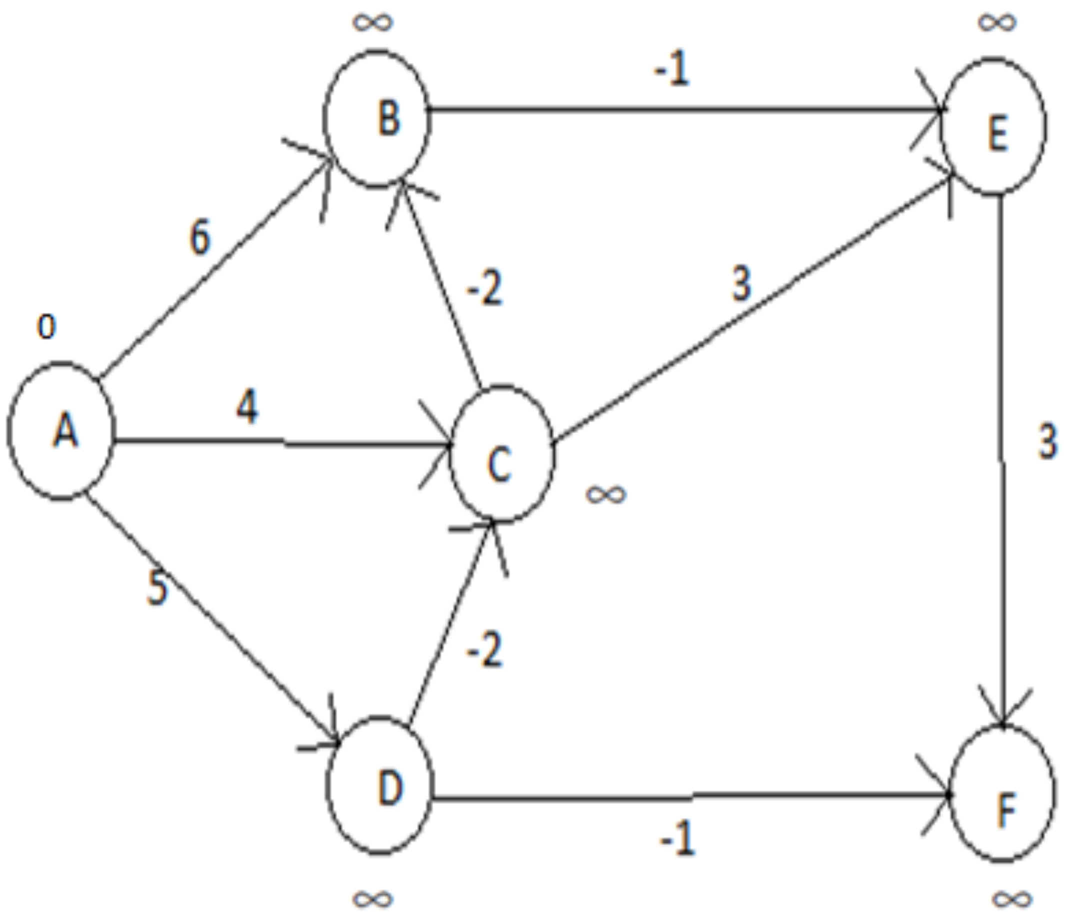
```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| − 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

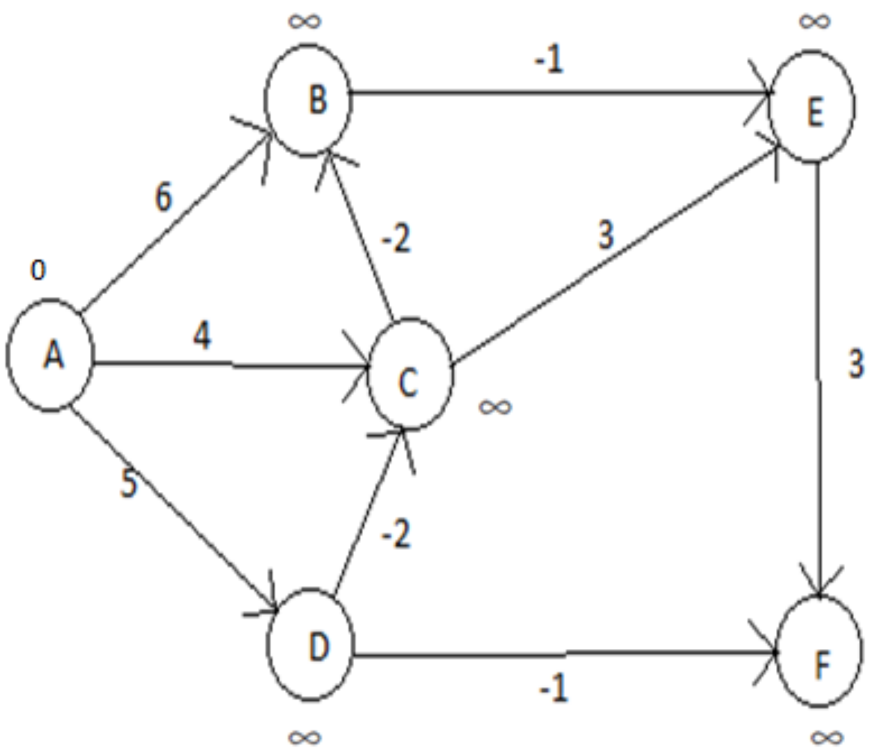
Example 1: Find the shortest path from A to all other vertices.



Step 1: Initialization



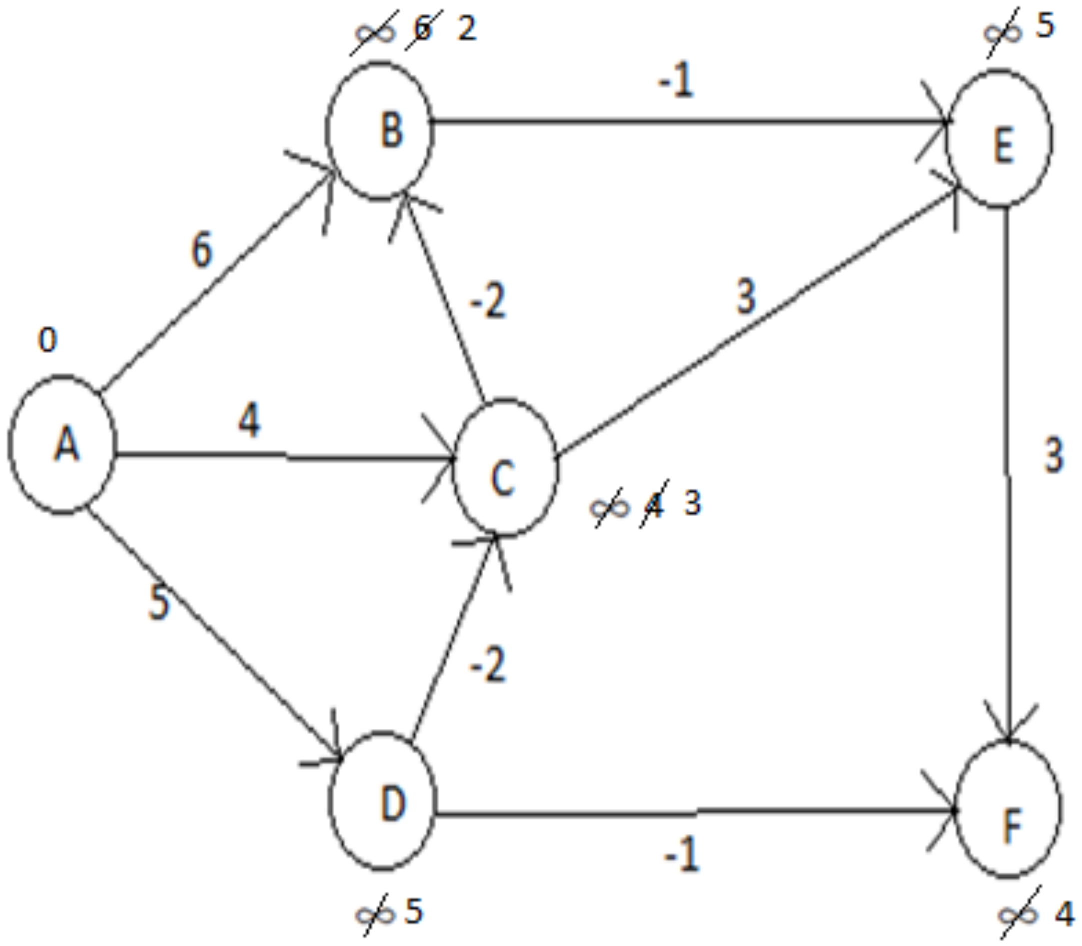
Step 2: List all the edges present in the graph.



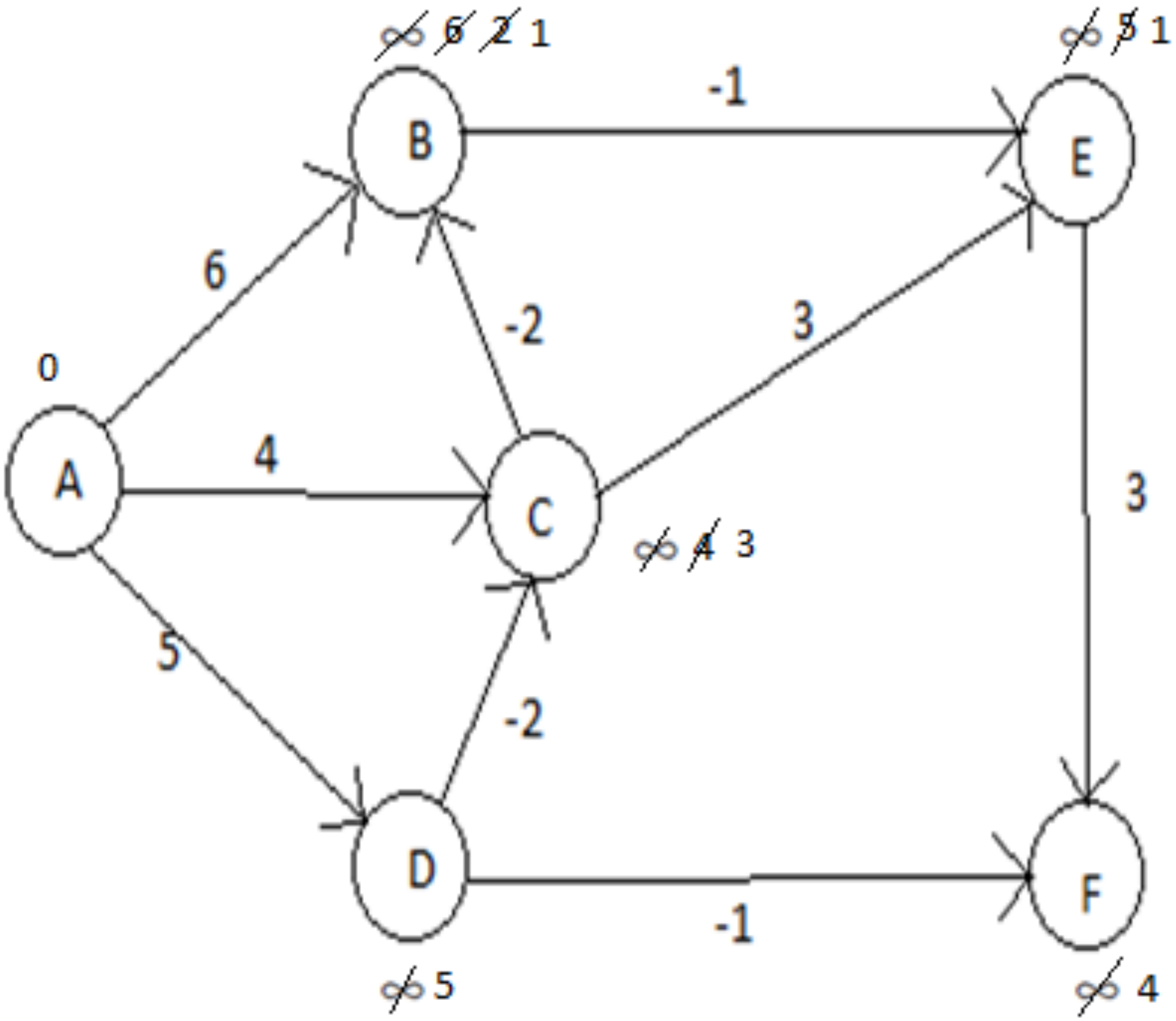
Edges: (AB), (AC), (AD), (BE), (CB), (CE), (DC), (DF), (EF)

Step 3: Relaxation

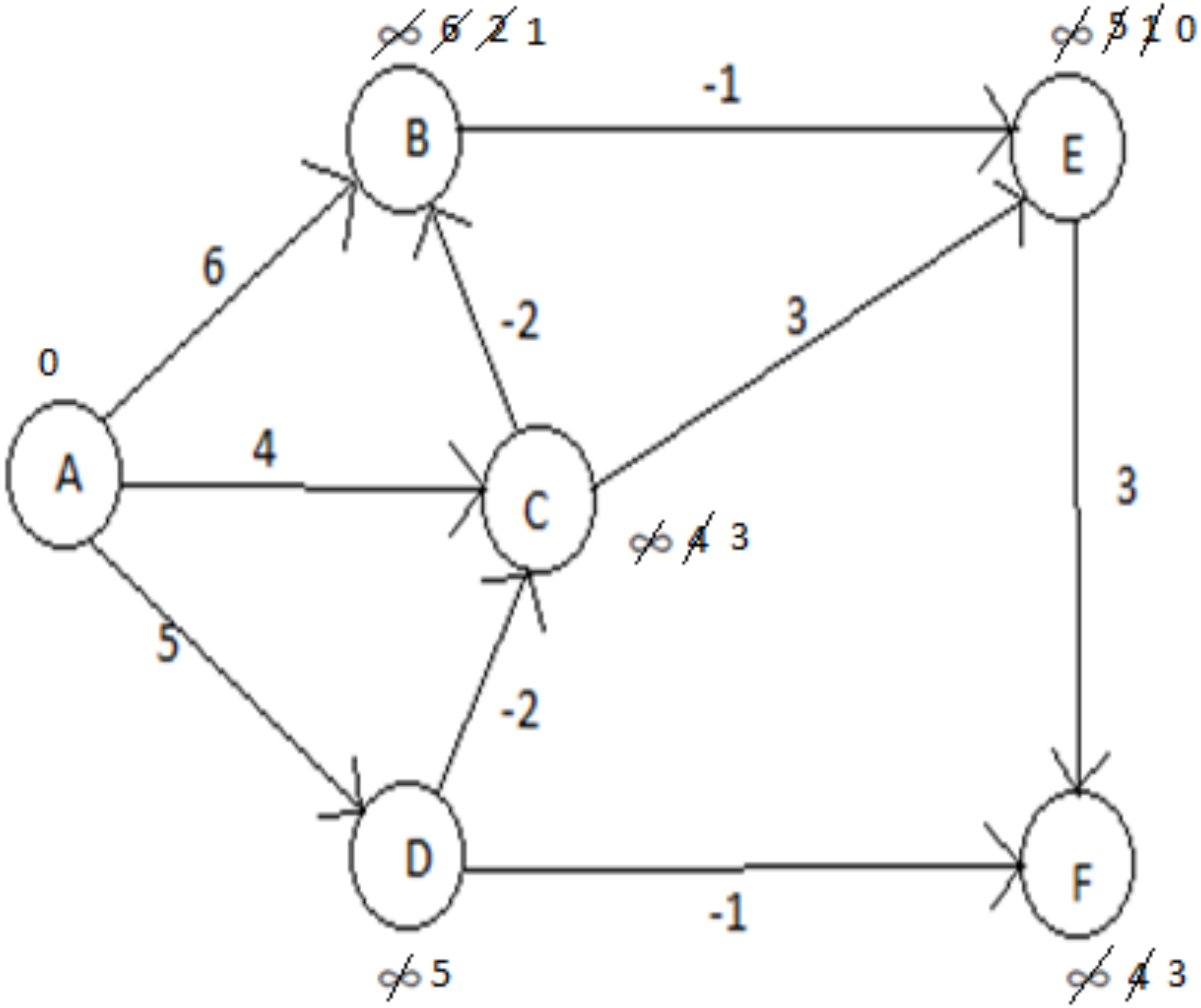
1st Iteration: (AB), (AC), (AD), (BE), (CB), (CE), (DC), (DF), (EF)



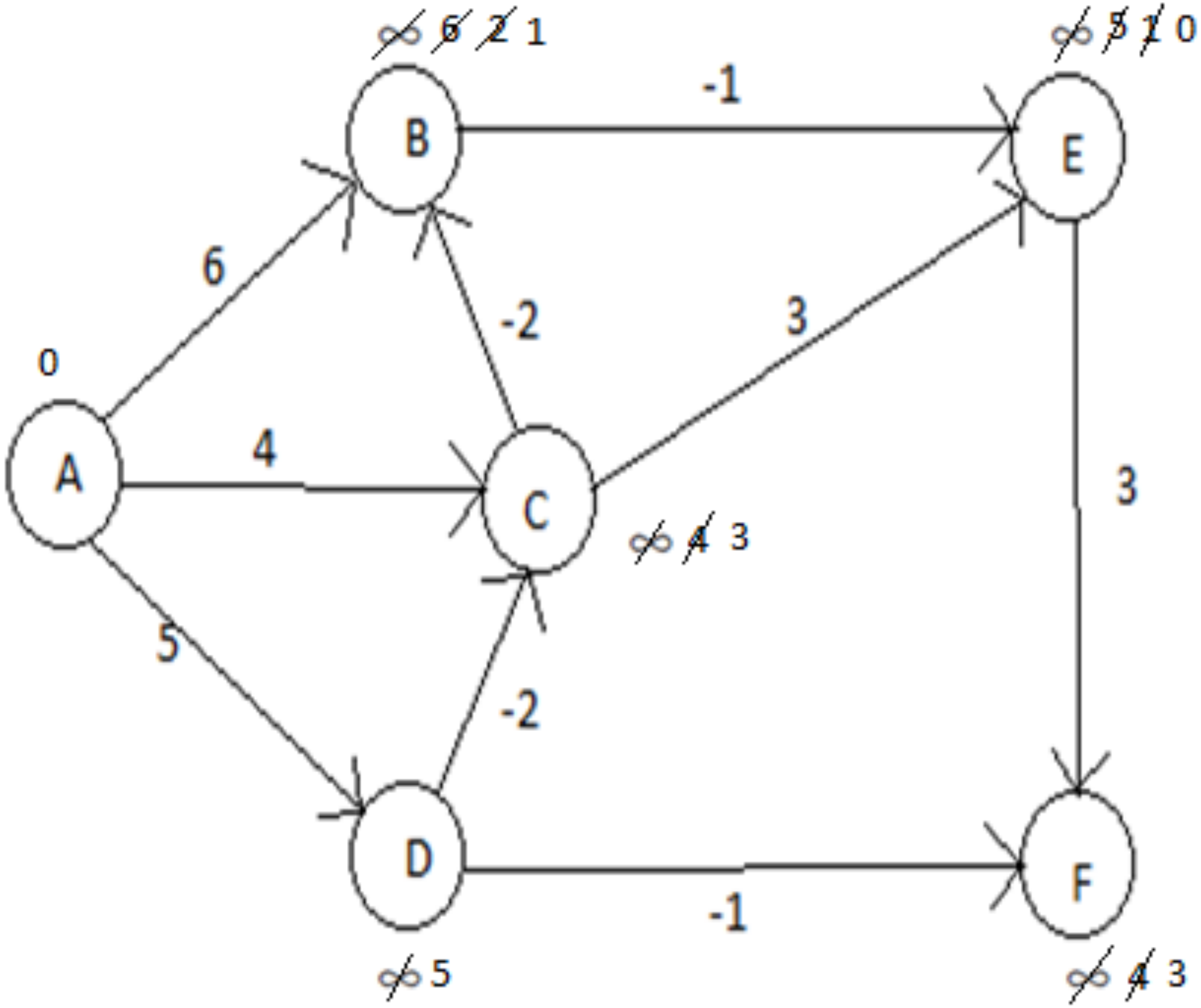
2nd Iteration : (AB), (AC), (AD), (BE), (CB), (CE), (DC), (DF), (EF)



3rd Iteration : (AB), (AC), (AD), (BE), (CB), (CE), (DC), (DF), (EF)



4th Iteration: (AB), (AC), (AD), (BE), (CB), (CE), (DC), (DF), (EF)



Observation:

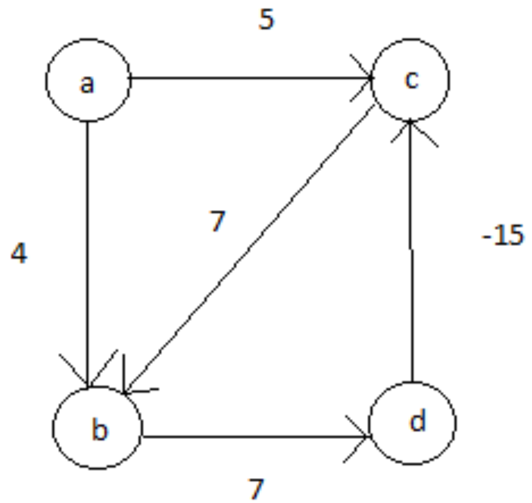
Though we are suppose to do five iterations here, we observed that there is no update in the 4th iteration. Hence we stop here, however the algorithm continues till (n-1) iterations.

Disadvantage:

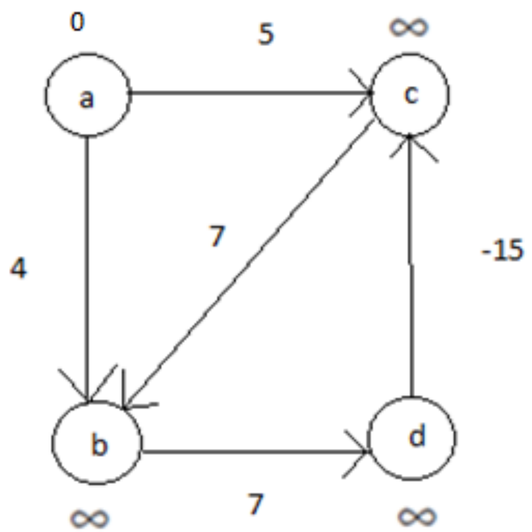
The disadvantage of Bellman Ford algorithm is, it will not work when there is a negative edge cycle present in the graph.

But a variation of the algorithm works for n-1 cycles and still if the cost changes, concludes that the graph leads to a negative weight cycle.

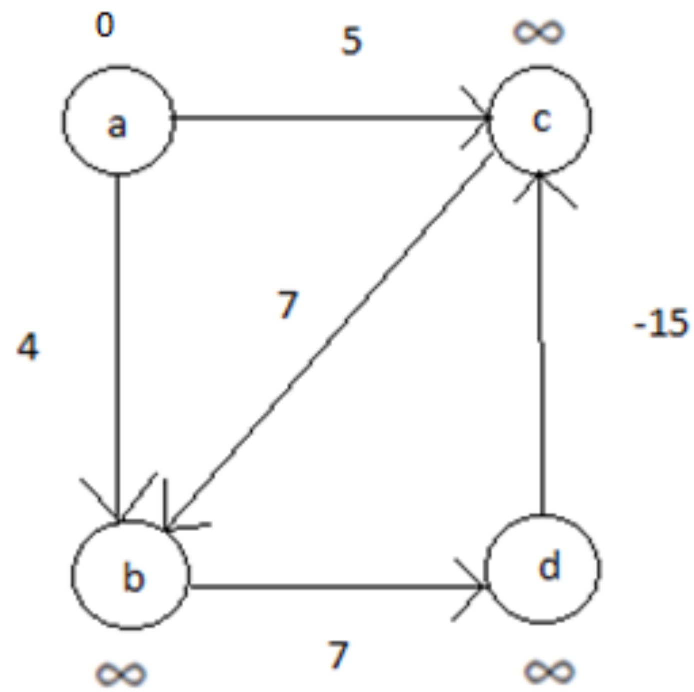
Example 2:



Step 1: Initialization



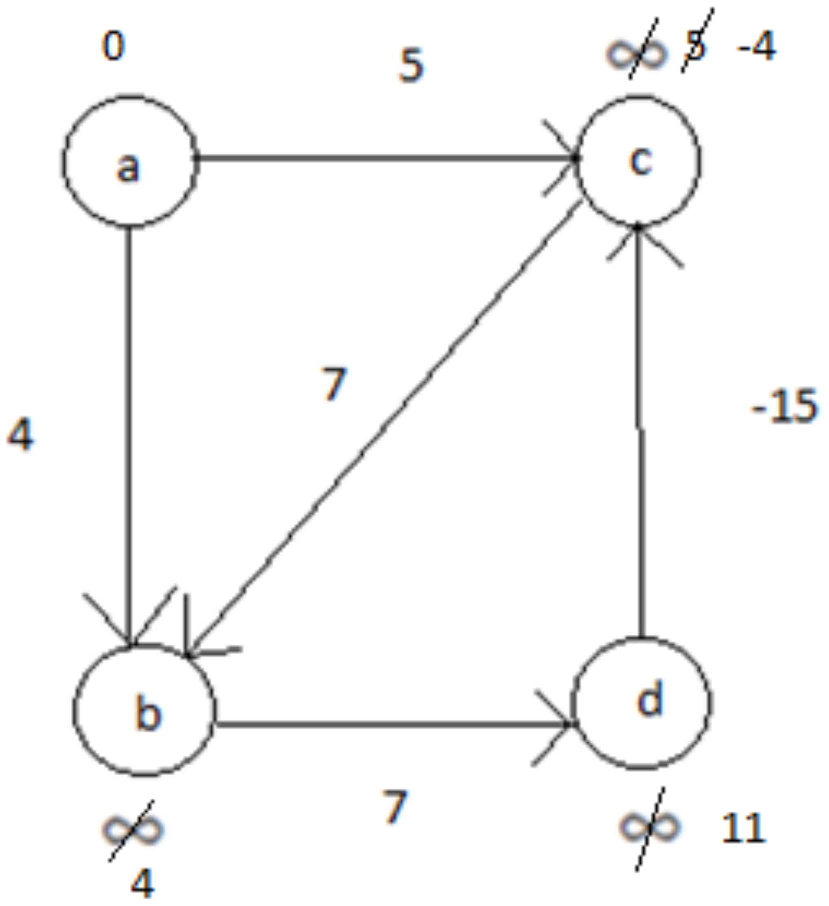
Step 2: List all the edges present in the graph.



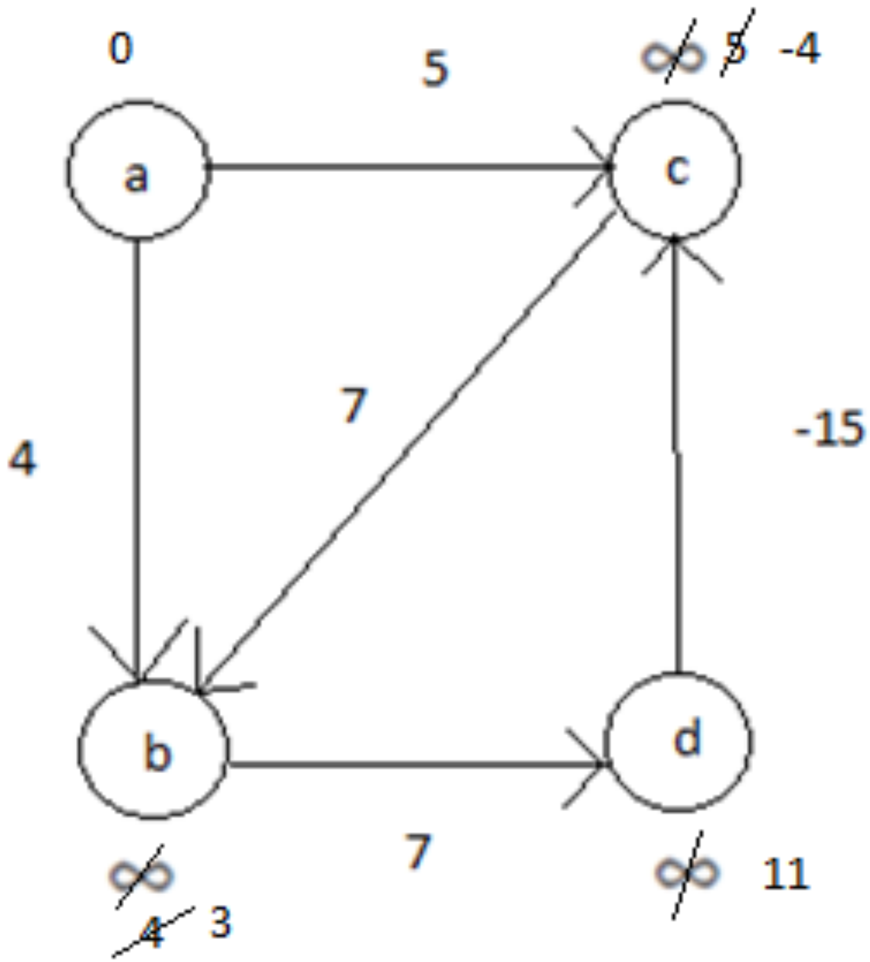
Edges: (ab), (ac), (bd), (cb), (dc)

Step 3: Relaxation.

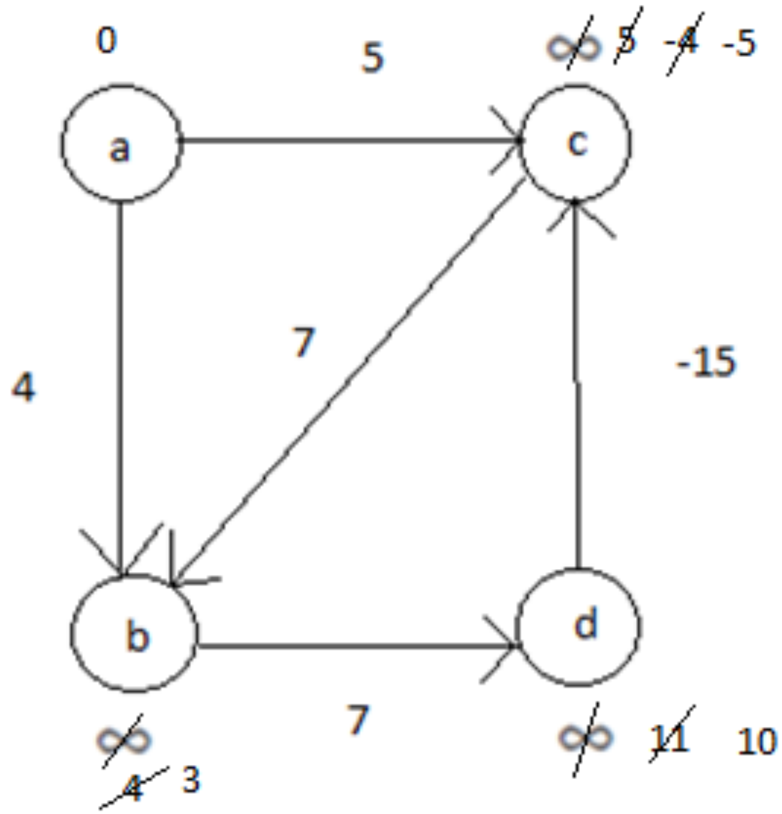
1st Iteration: (ab), (ac), (bd), (cb), (dc)



2nd Iteration: (ab), (ac), (bd), (cb), (dc)



3rd Iteration: (ab), (ac), (bd), (cb), (dc)



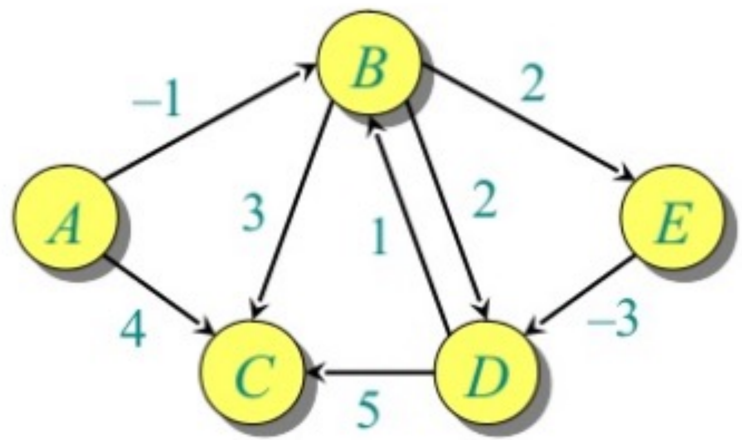
According to algorithm, from a we have found shortest path to all other nodes, but continue for one more iteration...

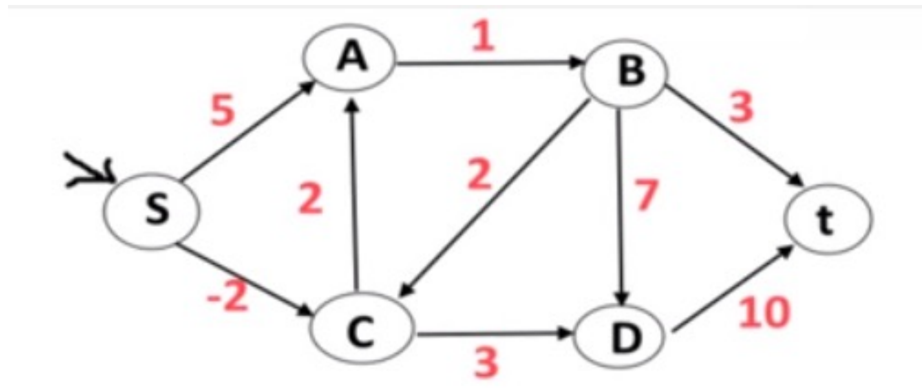
Bellman-Ford Analysis

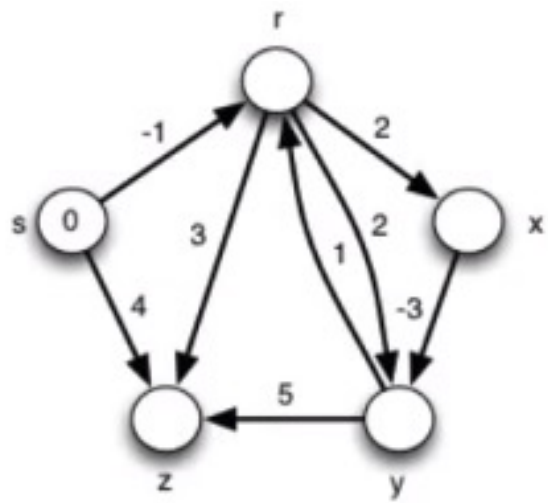
TOTAL: O(VE)

for v in V:
    $v.d = \infty$
    $v.\pi =$ None   O(V)
$s.d = 0$
for i from 1 to $|V| - 1$:
    for (u, v) in E: O(E)   O(VE)
        relax(u, v) O(1)
for (u, v) in E:
    if $v.d > u.d + w(u, v)$:   O(E)
        report that a negative-weight cycle exists

Try it yourself..!







DAG Algorithm

Shortest Paths in a DAG

Directed Acyclic Graph: No cycles; vertices must occur on shortest paths in an order consistent with a topological sort; negative weights not a problem.

Similar to Bellman-Ford, but fewer iterations:

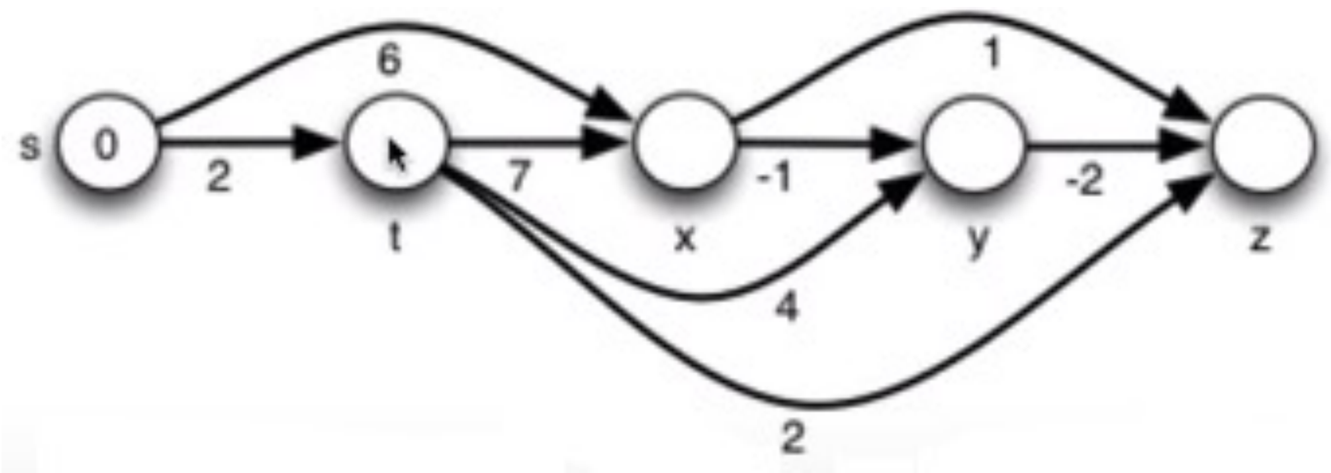
```
DAG-SHORTEST-PATHS(G, w, s)
topologically sort the vertices of G
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex u, taken in topologically sorted order
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Single-Source Shortest Paths in DAGs

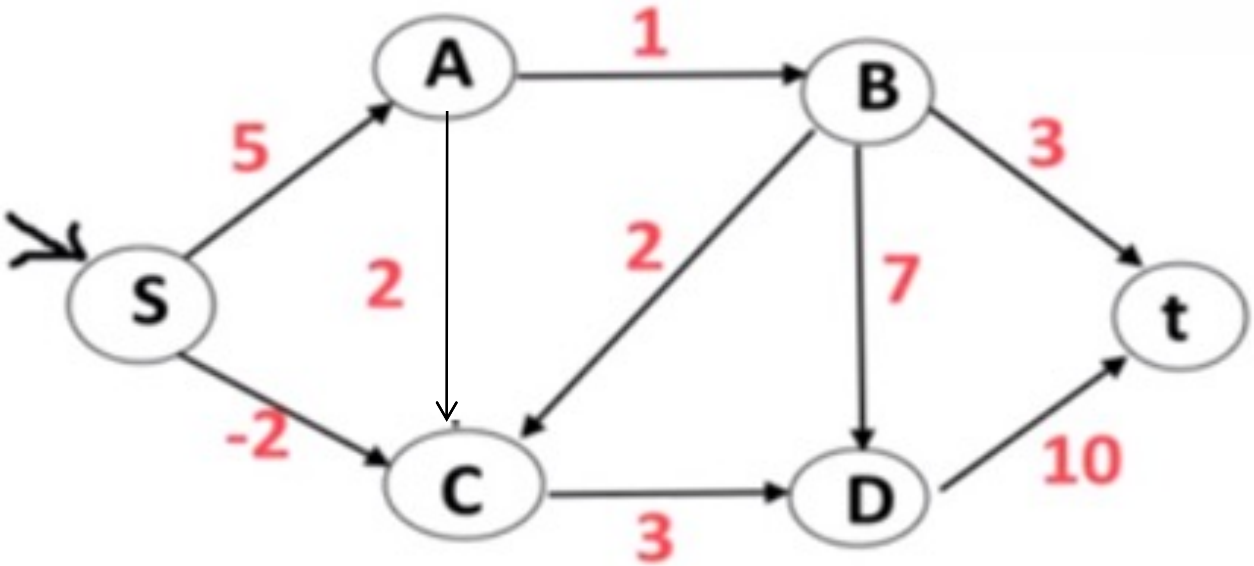
Runs in linear time: $\Theta(V+E)$

- topological sort: $\Theta(V+E)$
- initialization: $\Theta(V+E)$
- *for-loop:* $\Theta(V+E)$
  each vertex processed exactly once
  => each edge processed exactly once: $\Theta(V+E)$

Example 1



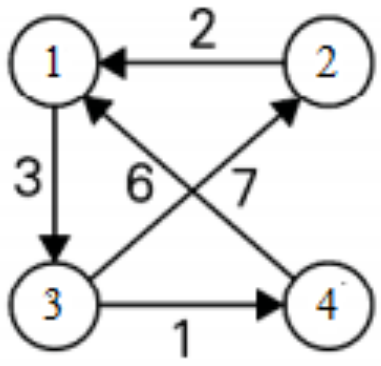
Example 2



All Pair Shortest Path Algorithm

- Shortest path between every pair of vertices.
- Floyd-Warshall algorithm.
- Dynamic programming approach.
- Input: Weighted graph without negative cycles.

Example 1:

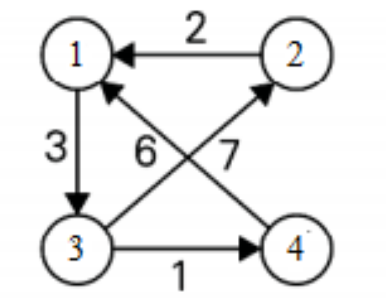


Step 1: Write the adjacency matrix of the graph.

$$
D^0 = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}
\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\
\begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}
$$

Step 2: To write the D^1 matrix keep 1^{st} row, 1^{st} column unaltered and use the below formula to fill the other values of matrix.

$\mathbf{D^{k}[i,j] \leftarrow min\ [\ D^{k-1}[i,j],\ D^{k-1}[i,k] + D^{k-1}[k,j]\ ]}$ where k varies from 1 to n which is the number of vertices.



$$D^0 = \begin{array}{cc} & \begin{array}{cccc}1 & 2 & 3 & 4\end{array} \\ \begin{array}{c}1\\2\\3\\4\end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix}\end{array} \qquad D^1 = \begin{array}{cc} & \begin{array}{cccc}1 & 2 & 3 & 4\end{array} \\ \begin{array}{c}1\\2\\3\\4\end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix}\end{array}$$

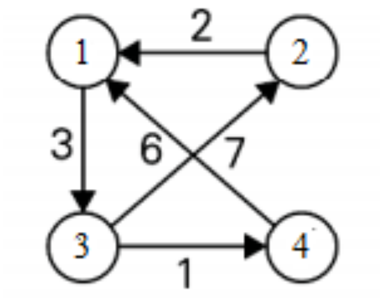
$\mathbf{D^{k}[i,j] \leftarrow min\ [\ D^{k-1}[i,j],\ D^{k-1}[i,k] + D^{k-1}[k,j]\ ]}$

$D^1[2,3] \leftarrow \min [\ D^{1-1}[2,3],\ D^{1-1}[2,1] + D^{1-1}[1,3]\ ]$

$\min [\ \infty, 5] = 5$

Similarly find the other values of matrix which is D^1

Step 3: To write the D^2 matrix keep 2^{nd} row, 2^{nd} column unaltered and use the below formula to fill the other values of matrix.



$$D^0 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$D^1 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

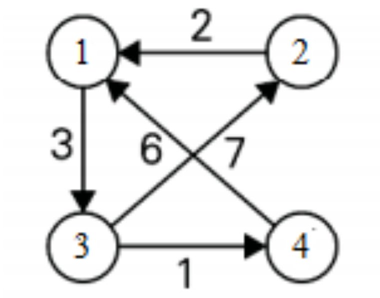
$$\mathbf{D^k[i,j] \leftarrow \min [\, D^{k-1}[i,j],\ D^{k-1}[i,k] + D^{k-1}[k,j] \,]}$$

$$D^2[3,1] \leftarrow \min [\, D^{2-1}[3,1],\ D^{2-1}[3,2] + D^{2-1}[2,1] \,]$$

$$\min [\, \infty, 9] = 9$$

$$D^2 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ 9 & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

Step 3: To write the D^3 matrix keep 3^{nd} row, 3^{nd} column unaltered and use the below formula to fill the other values of matrix.



$$D^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$D^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

$$D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ 9 & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

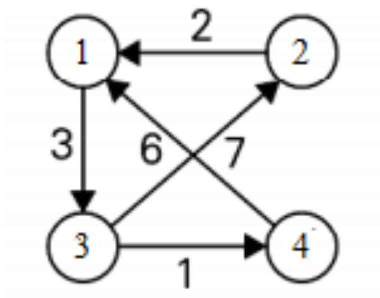
$D^k[i,j] \leftarrow \min[\, D^{k-1}[i,j],\ D^{k-1}[i,k] + D^{k-1}[k,j] \,]$

$D^3[1,2] \leftarrow \min[\, D^{3-1}[1,2],\ D^{3-1}[1,3] + D^{3-1}[3,2] \,]$

$\min[\, \infty, 10] = 10$

$$D^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

Step 4: To write the D^4 matrix keep 4^{th} row, 4^{th} column unaltered and use the below formula to fill the other values of matrix.



$$D^0 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$D^1 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

$$D^2 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ 9 & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix} \end{array}$$

$$D^3 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 9 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

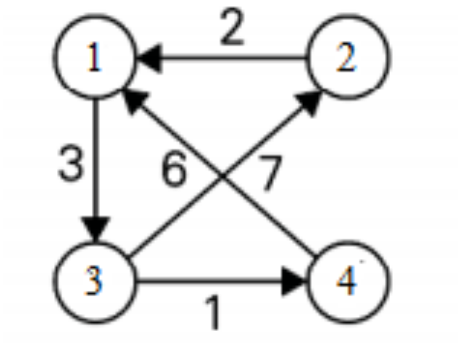
$D^k[i,j] \leftarrow \min [\, D^{k-1}[i,j], D^{k-1}[i,k] + D^{k-1}[k,j] \,]$

$D^4[3,1] \leftarrow \min [\, D^{4-1}[3,1], D^{4-1}[3,4] + D^{4-1}[4,1] \,]$

$\min [\, 9, 7] = 7$

$$D^4 = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

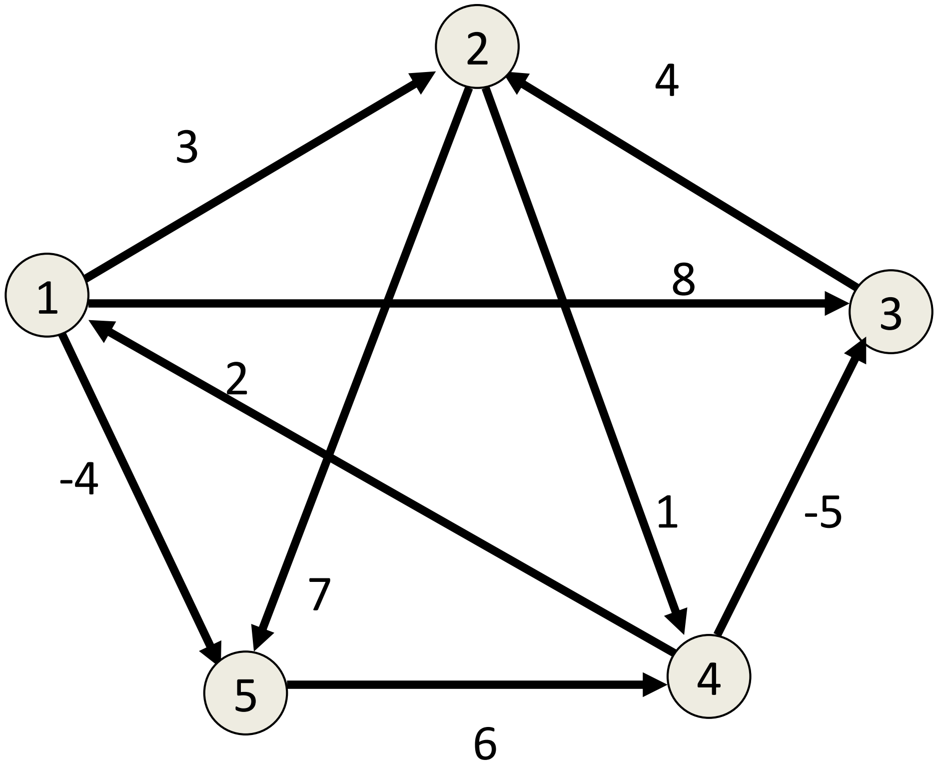
When the value of k =4, algorithm will stop working since there are only 4 vertices.



$$D^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$D = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

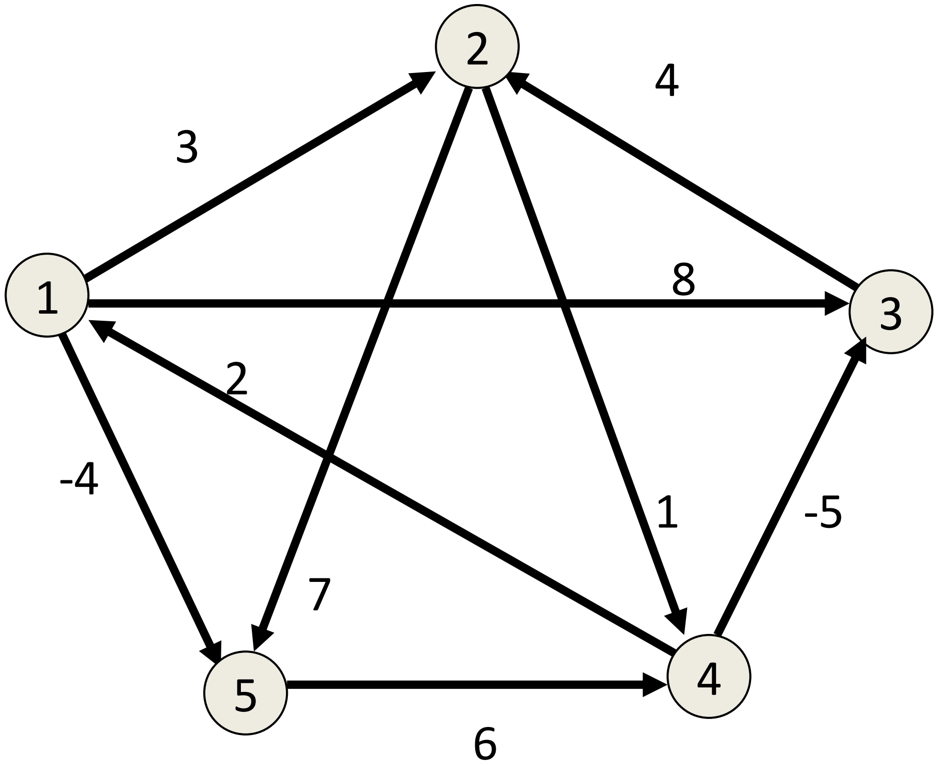
Example



$$D^{(0)} = W$$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

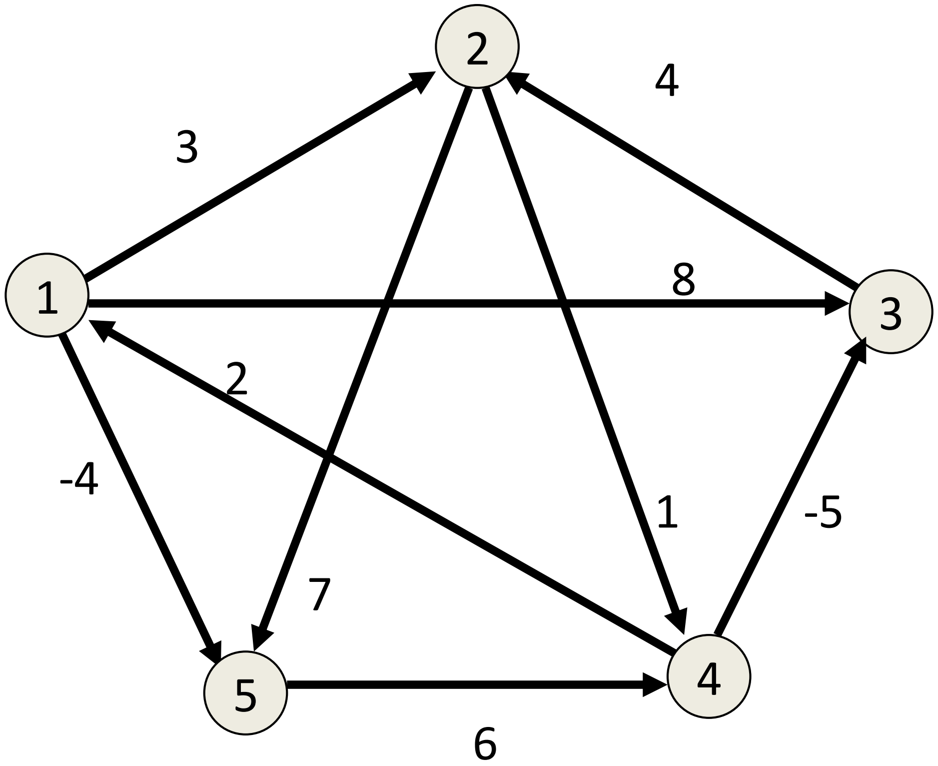
Example



$$D^{(1)}$$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

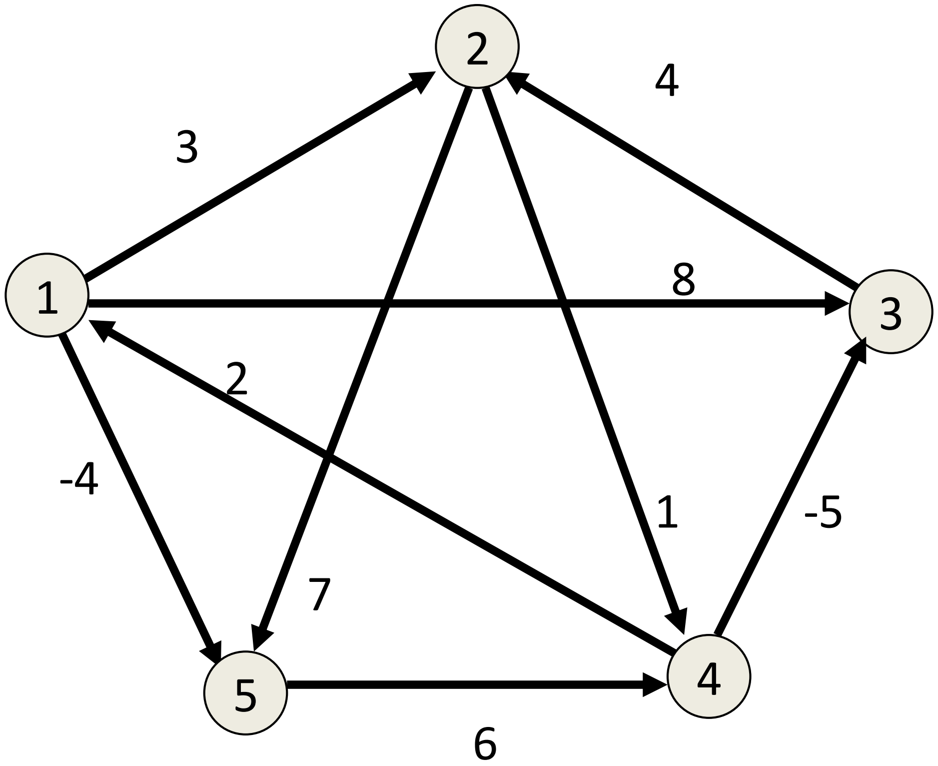
Example



$$D^{(2)}$$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

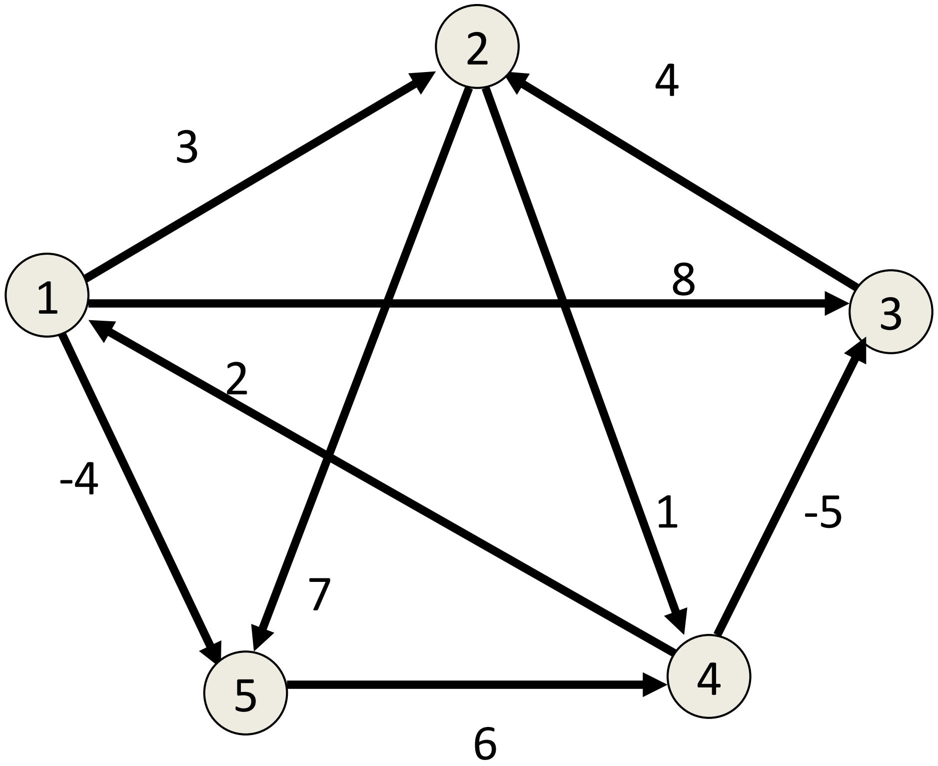
Example



$$D^{(3)}$$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

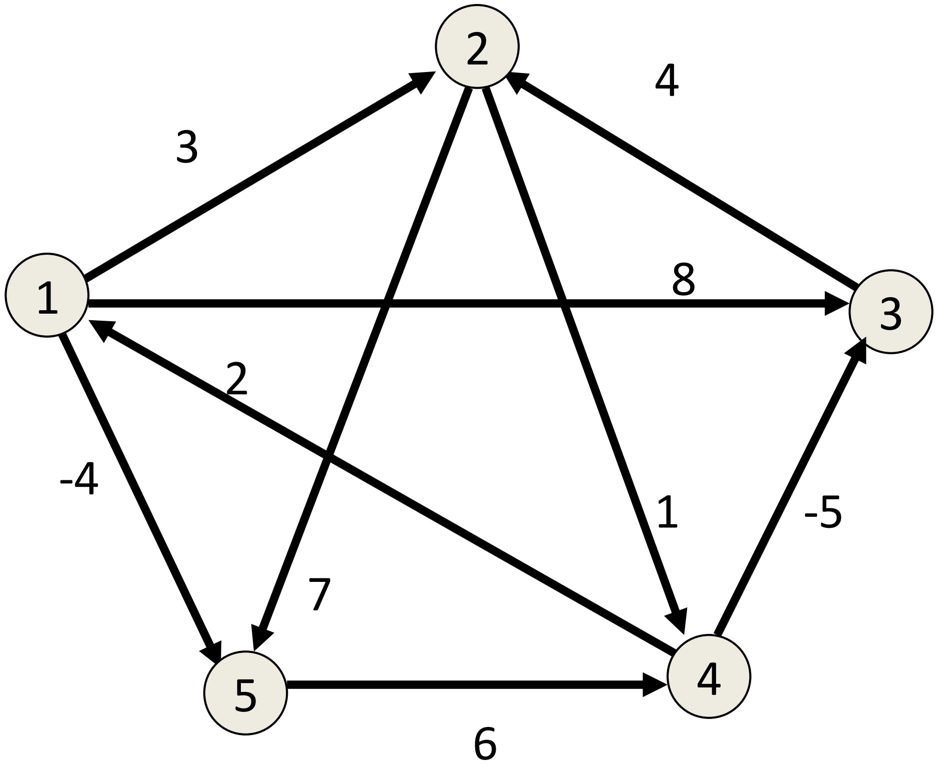
Example



$D^{(4)}$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

Example



$D^{(5)}$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

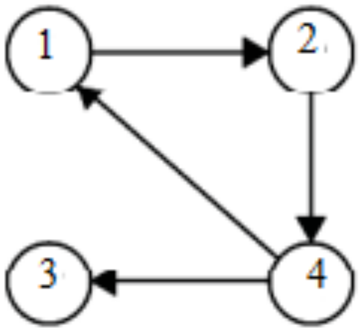
Algorithm Floyd Warshall

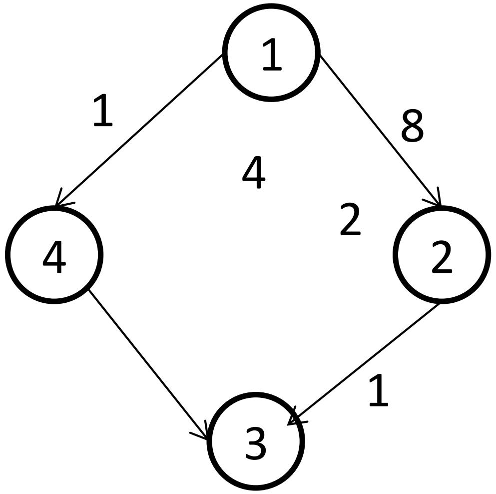
```
for i← 1 to n do
    for j← 1 to n do
        if (i==j) then
            D[i,j] ← 0
        else if ((v_i, v_j) is an edge in graph) then
            D[i,j] ← W[i,j]
        else
            D[i,j] ← ∞
    end for
end for

for k← 1 to n do
    for i← 1 to n do
        for j← 1 to n do
            D^k [i,j] ← min [ D^(k-1)[i,j], D^(k-1)[i,k] + D^(k-1)[k,j] ]
        end for
    end for
end for
```

Complexity: $T(n) = O(n^2) + O(n^3) = \mathbf{O(n^3)}$

Example 2: Try yourself..!





Example 3: Try yourself..!