Single Source Shortest Paths
Book: Cormen; Chapter: 24

- DAG Shortest Paths: Section: 24.2
- Dijsktra: Section: 24.3
- Bellman-Ford: Section: 24.1

Shortest Paths: Notations

- **Weight of a path** $p = \langle v_0, v_1, \ldots, v_k \rangle$ is the sum of the weights of its constituent edges:

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

- **Shortest-path weight** from u to v:

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{if there is a path from } u \text{ to } v, \\ \infty & \text{otherwise.} \end{cases}$$

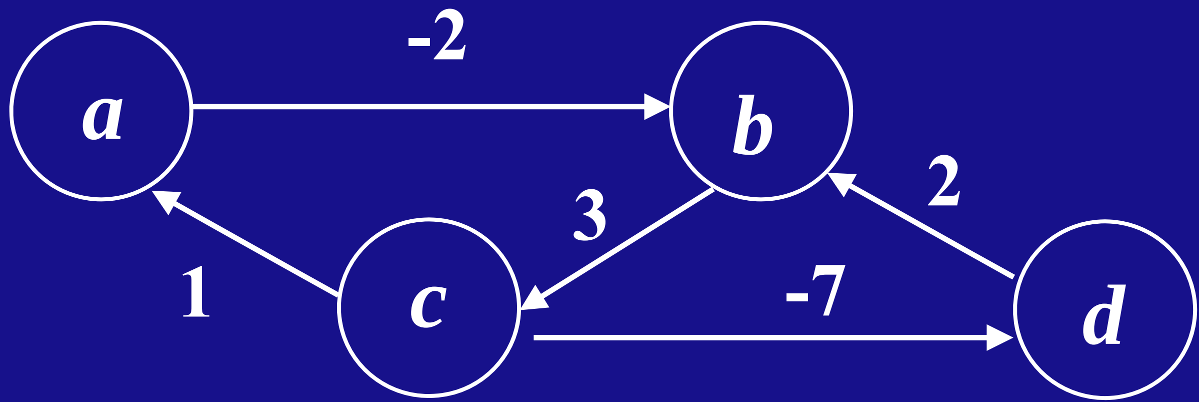
- **A shortest path** from vertex u to v is any path p with weight:

$$w(p) = \delta(u, v)$$

- BFS is a shortest path algorithm that works on unweighted graphs.

Negative Weight

- Negative-weight edges
- Negative-weight cycles



- Algorithms allow negative-weight edges, but disallow (or detect) negative-weight cycles.

Optimal substructure of a shortest path

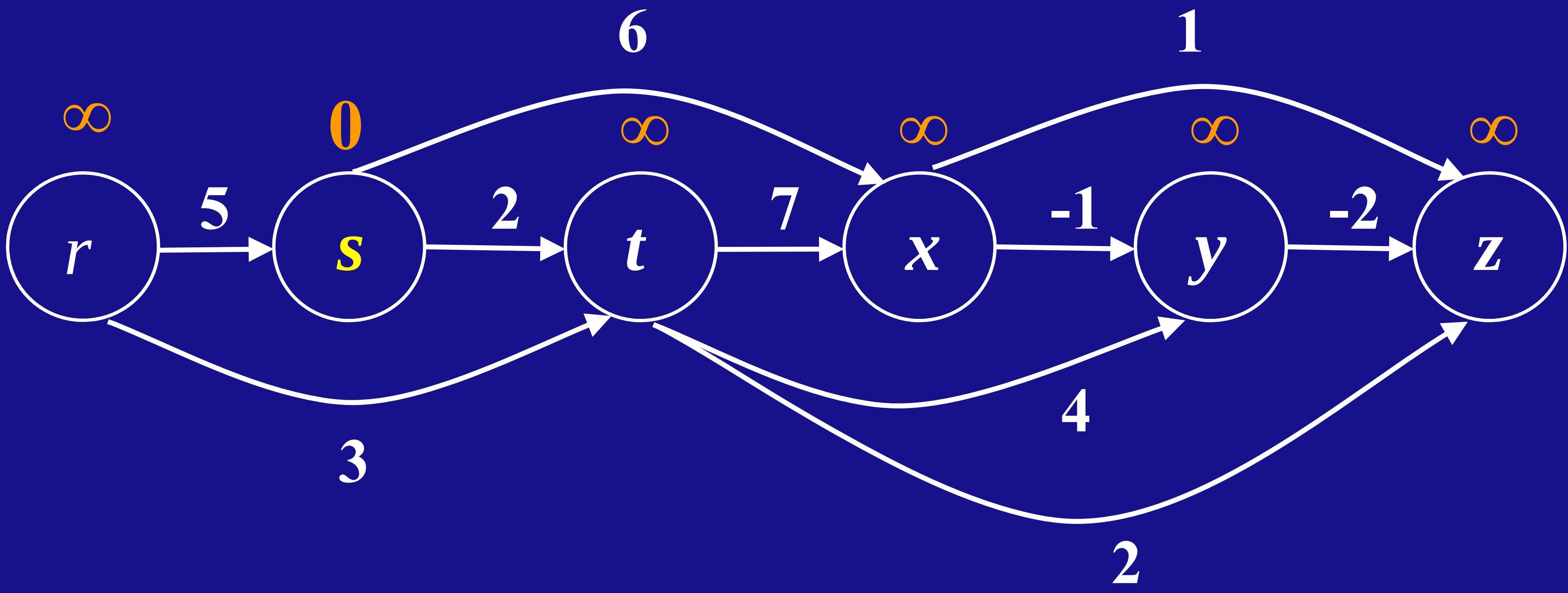
- Sub-paths of shortest paths are shortest paths
  - Given a weighted, directed graph $G = (V, E; w)$, let $p = \langle v_1, v_2, \ldots, v_k \rangle$ be a shortest path from vertex v_1 to vertex v_k and, for any i and j such that $1 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ be the sub-path of p from vertex v_i to vertex v_j. Then, p_{ij} is a shortest path from v_i to v_j.
  - Why?
  - **Page - 645**

Optimal substructure of a shortest path
Proof

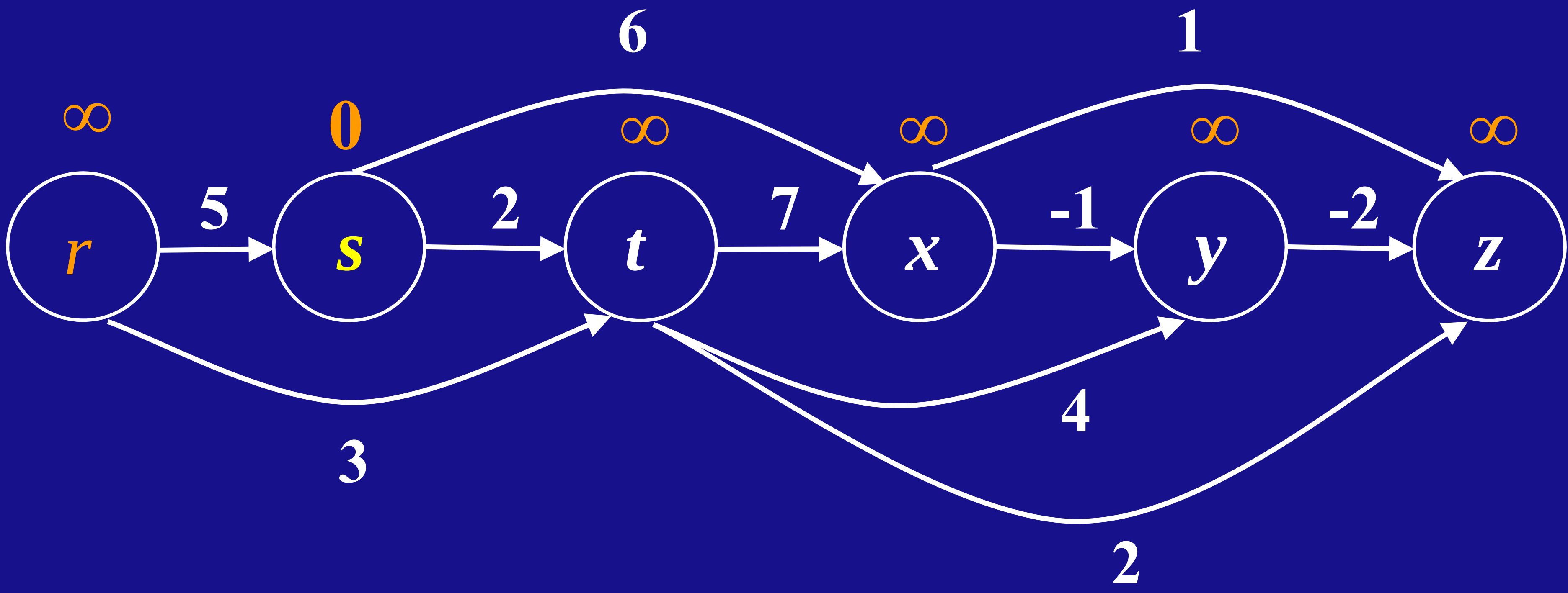
Single-Source Shortest Paths Problem

- Input: A weighted directed graph $G=(V, E, w)$, and a source vertex s.
- Output: Shortest-path weight from s to each vertex v in V, and a shortest path from s to each vertex v in V if v is reachable from s.

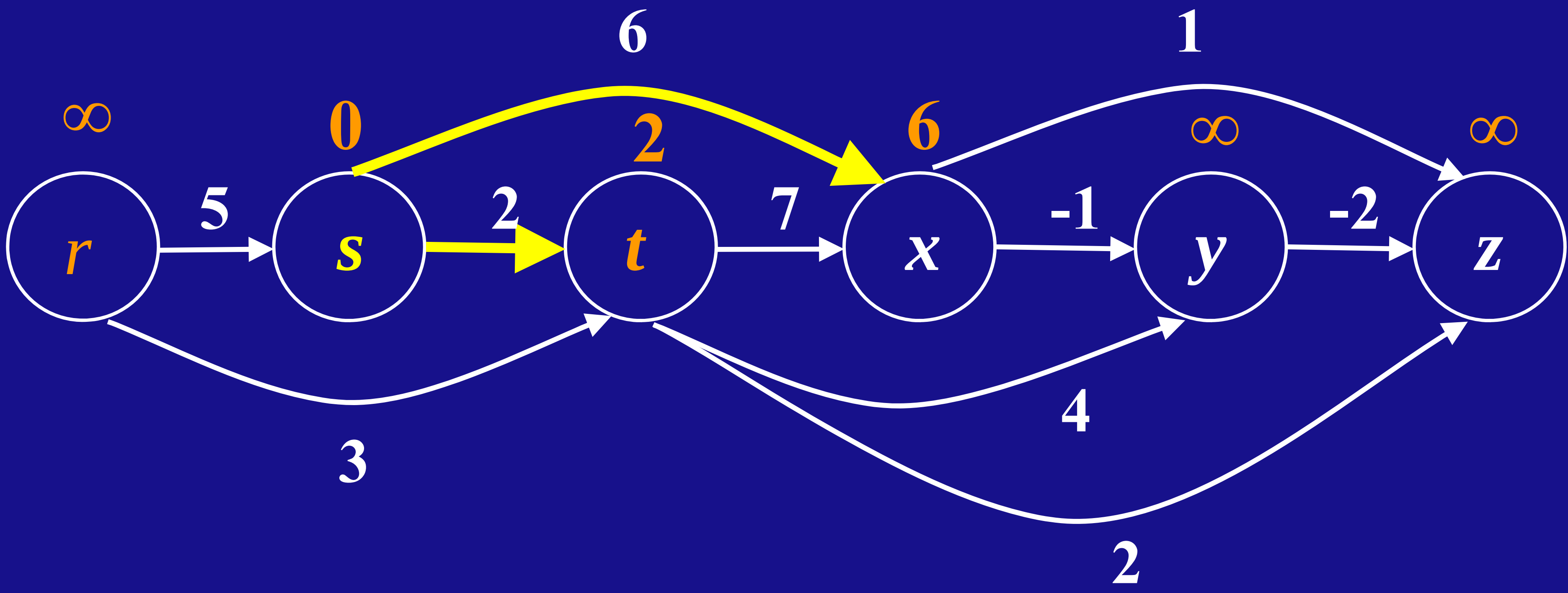
Example (Page-656): DAG Shortest Paths (1)



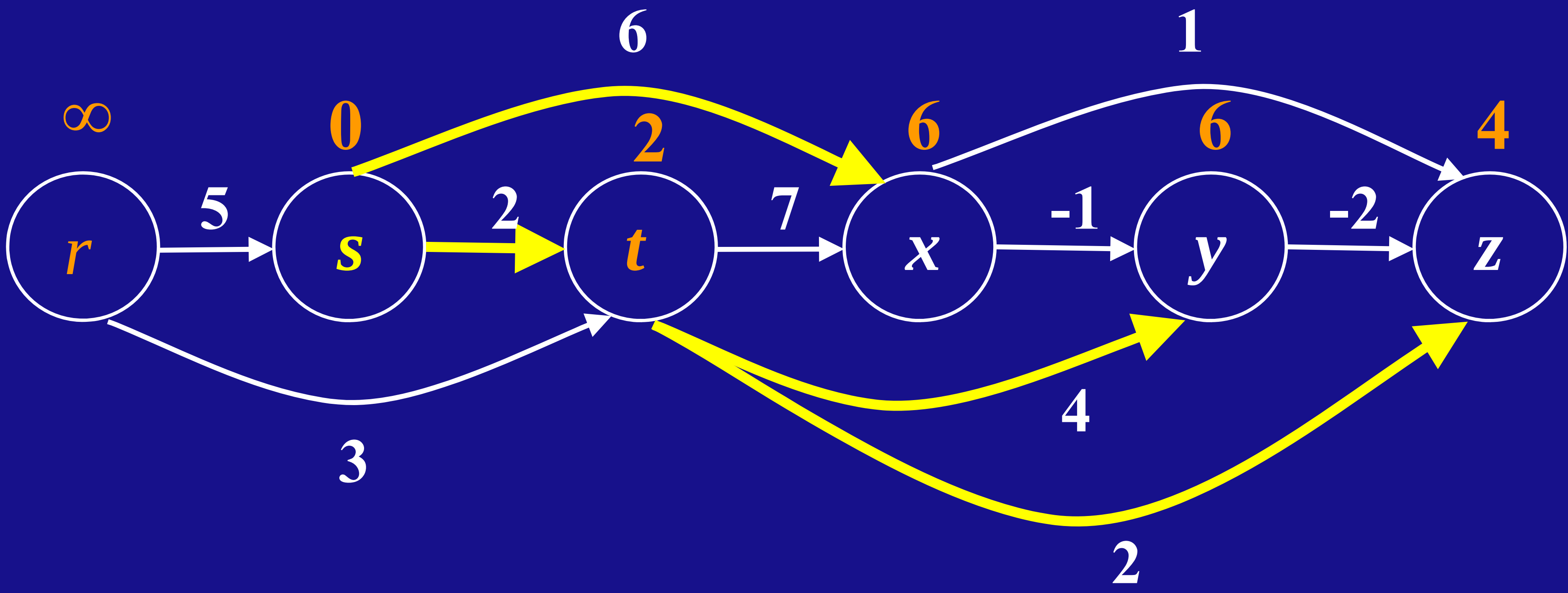
Example (Page-656): DAG Shortest Paths (2)



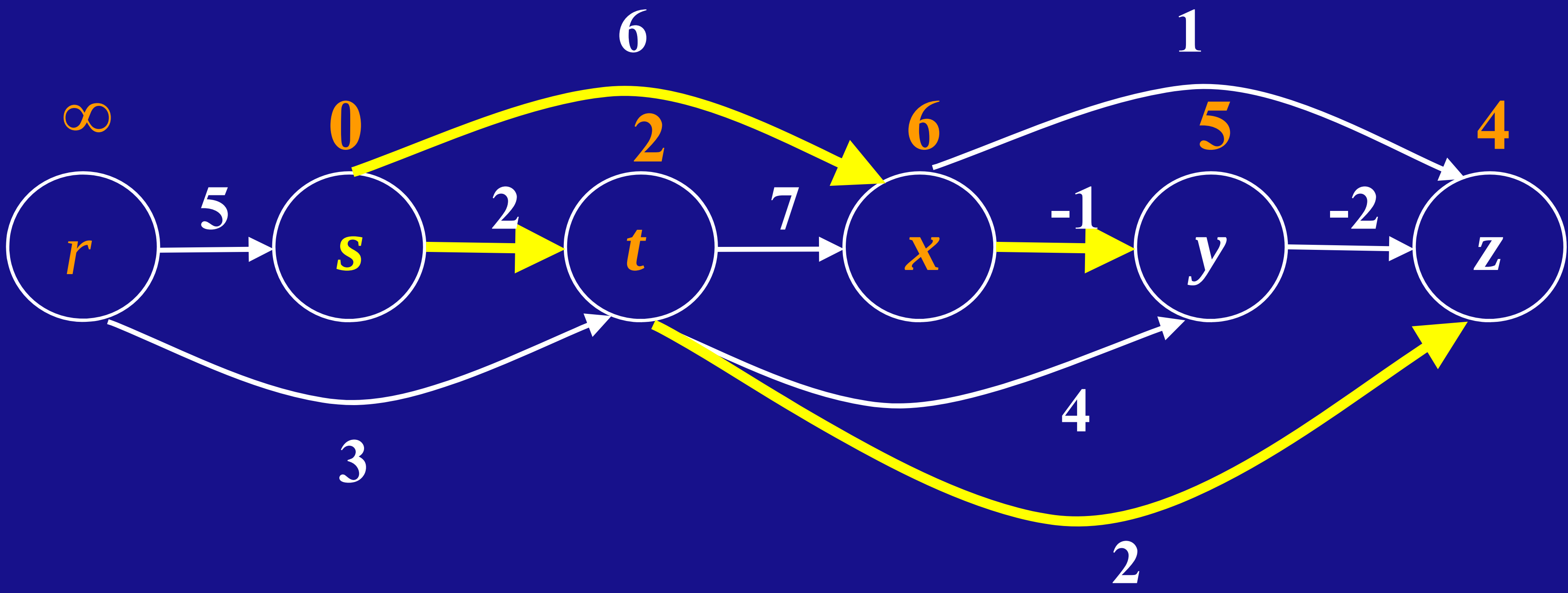
Example (Page-656): DAG Shortest Paths (3)



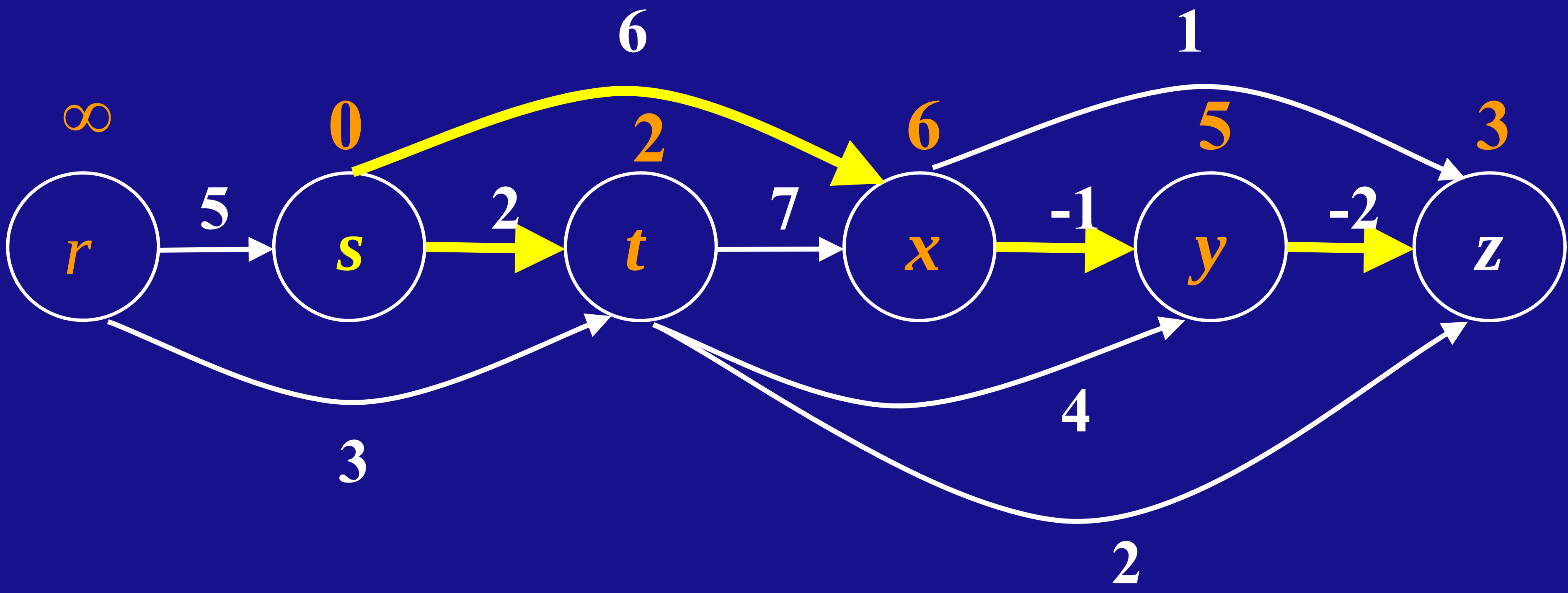
Example (Page-656): DAG Shortest Paths (4)



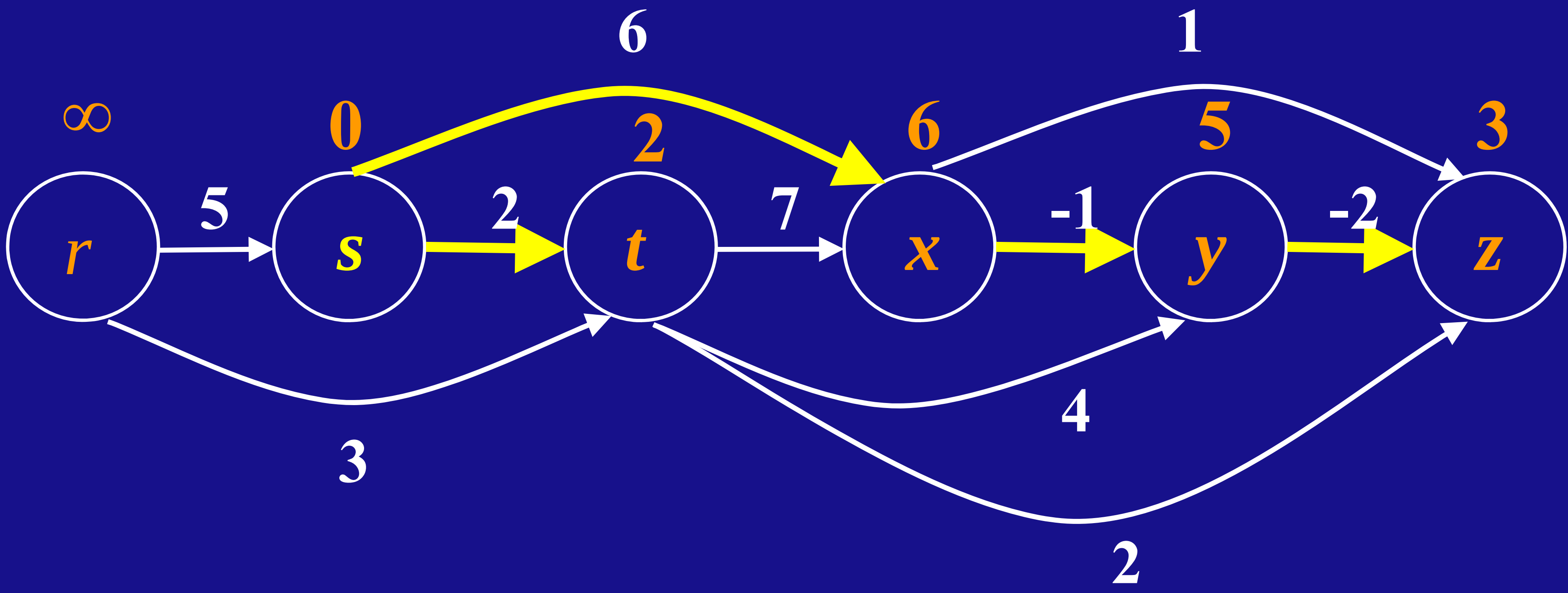
Example (Page-656): DAG Shortest Paths (5)



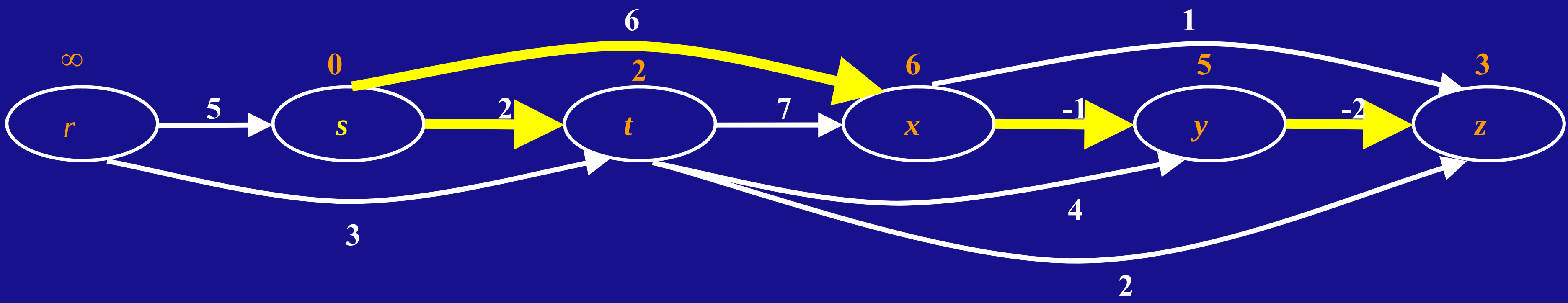
Example (Page-656): DAG Shortest Paths (6)



Example (Page-656): DAG Shortest Paths (7)

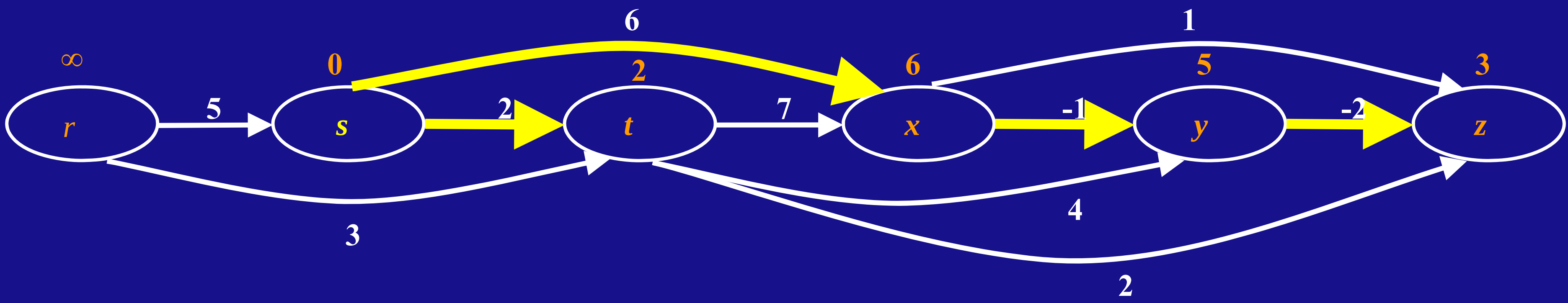


Example (Page-656): DAG Shortest Paths (8)



Selected Node	r	s	t	x	y	z
Initialize	∞ / Nil	0 / Nil	∞ / Nil	∞ / Nil	∞ / Nil	∞ / Nil
r	∞ / Nil	0 / Nil				
s	∞ / Nil	0 / Nil				
t	∞ / Nil	0 / Nil				
x	∞ / Nil	0 / Nil				
y	∞ / Nil	0 / Nil				
z	∞ / Nil	0 / Nil	2 / s	6 / s	5 / x	3/ y

Example (Page-656): DAG Shortest Paths (9)



Source Node	r	s	t	x	y	z
s	∞ / Nil	0 / Nil	2 / s	6 / s	5 / x	3/ y

Source Node	Destination Node	Paths	Cost
s	r	×	∞
	t	s → t	2
	x	s → x	6
	y	s → x → y	5
	z	s → x → y → z	3

Algorithm (Page - 655): DAG Shortest Paths

```
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
      do for each vertex v ∈ Adj[u]
            do RELAX(u, v, w)
```

Initial Estimate (Page - 648)

INITIALIZE-SINGLE-SOURCE(G, s)

1 **for** each vertex $v \in V[G]$
2 **do** $d[v] \leftarrow \infty$
3 $\pi[v] \leftarrow$ NIL
4 $d[s] \leftarrow 0$

Edge Relaxation (Page - 649)

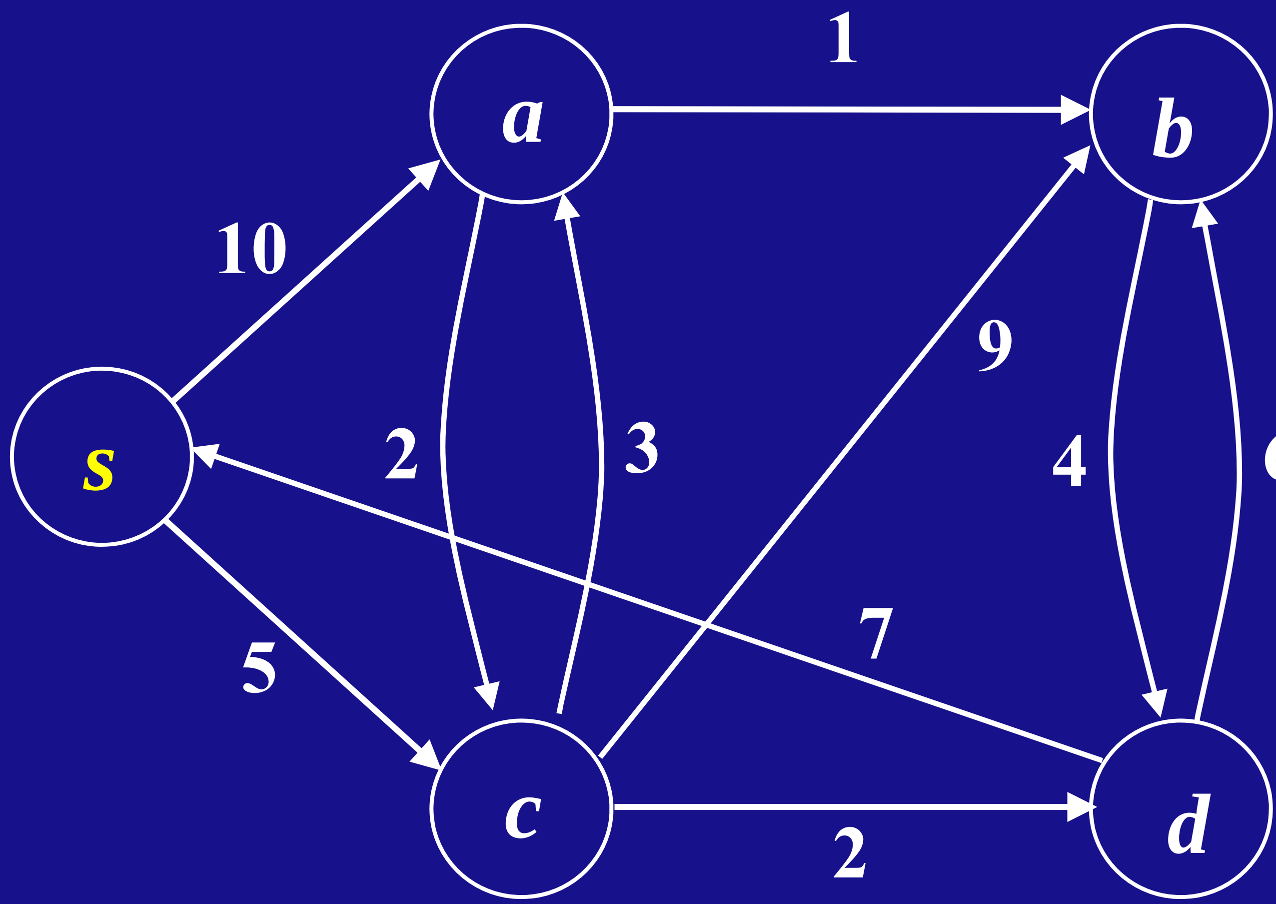
RELAX(u, v, w)

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

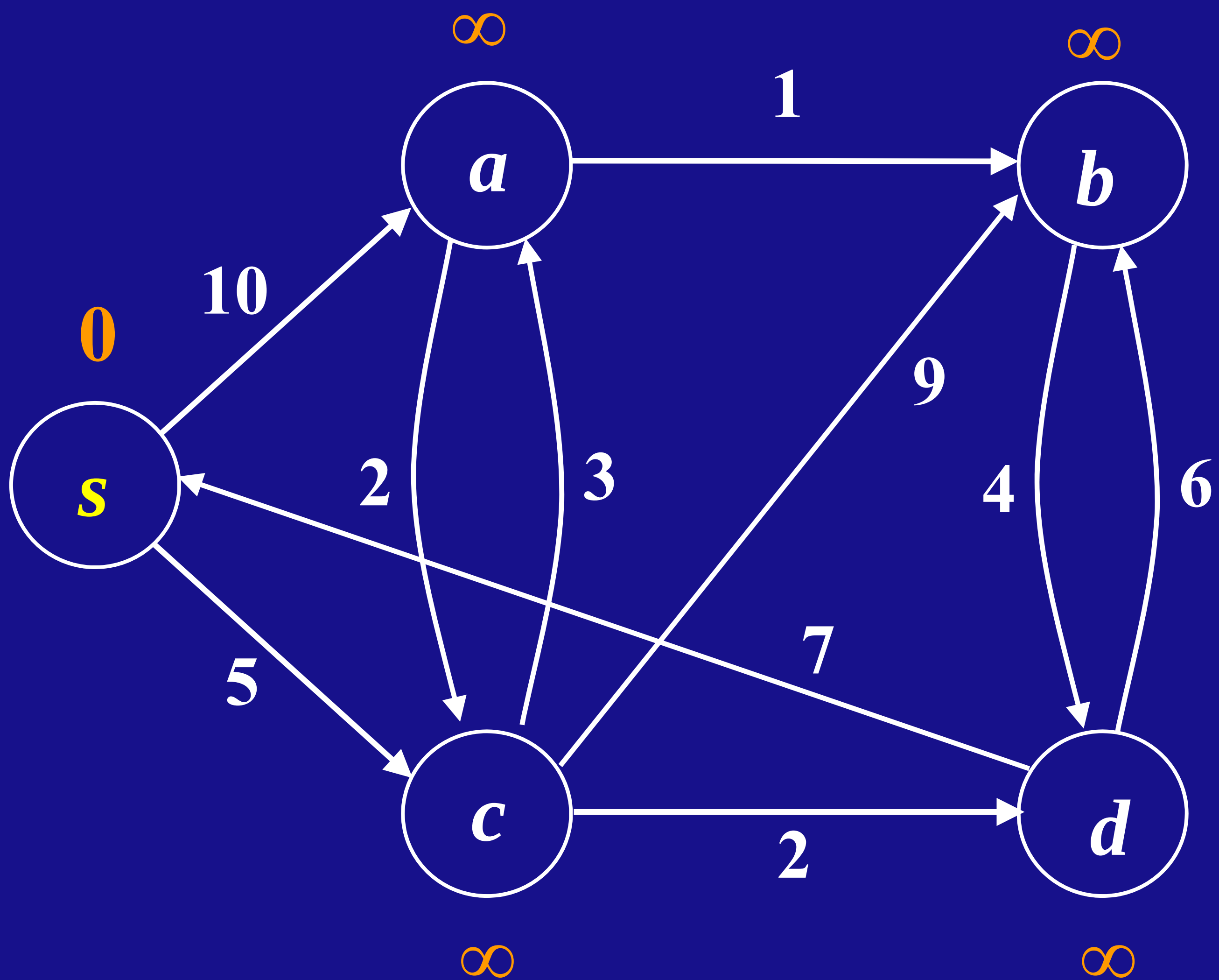
2 **then** $d[v] \leftarrow d[u] + w(u, v)$

3 $\pi[v] \leftarrow u$

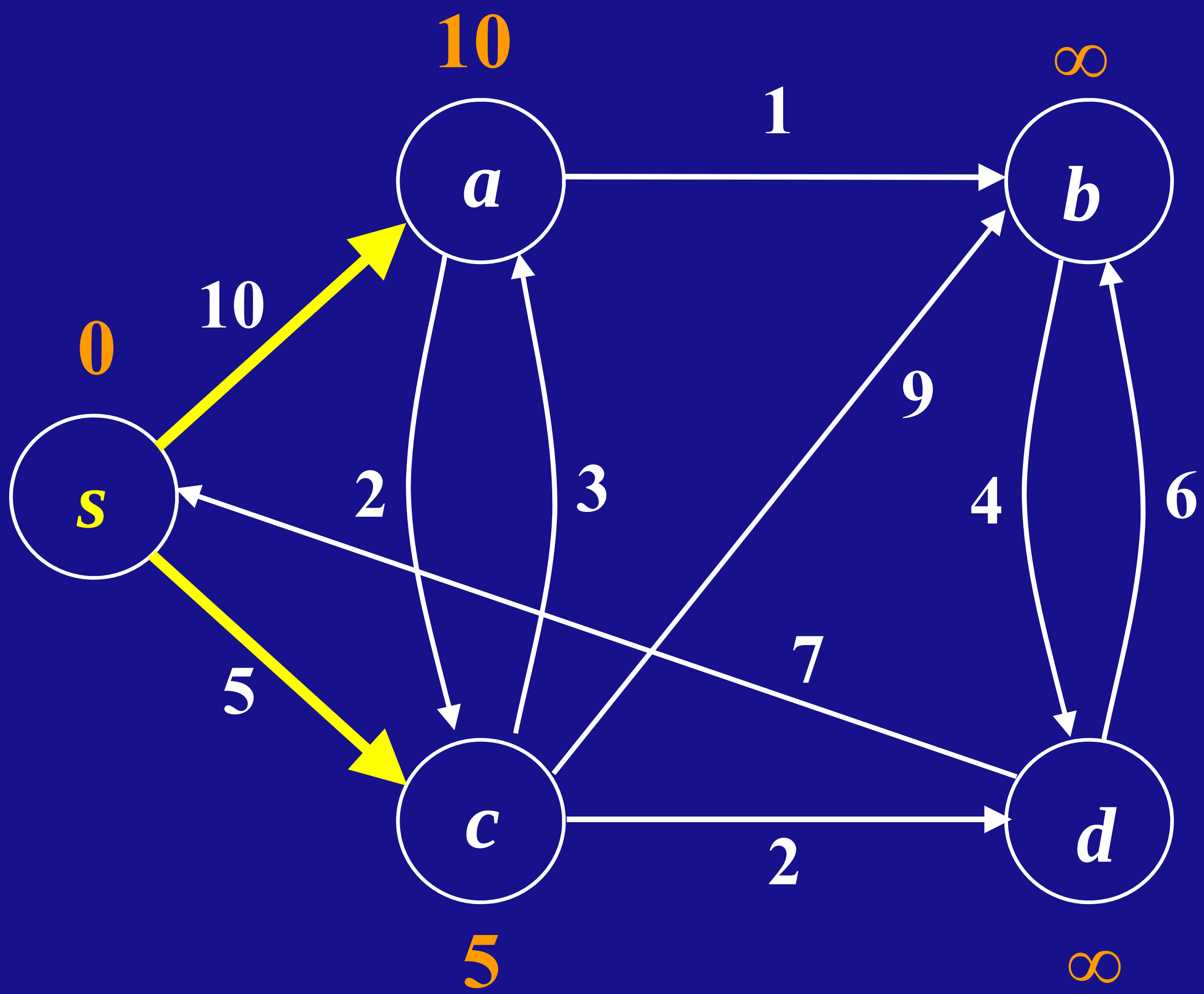
Example (Page-659): Dijkstra (1)



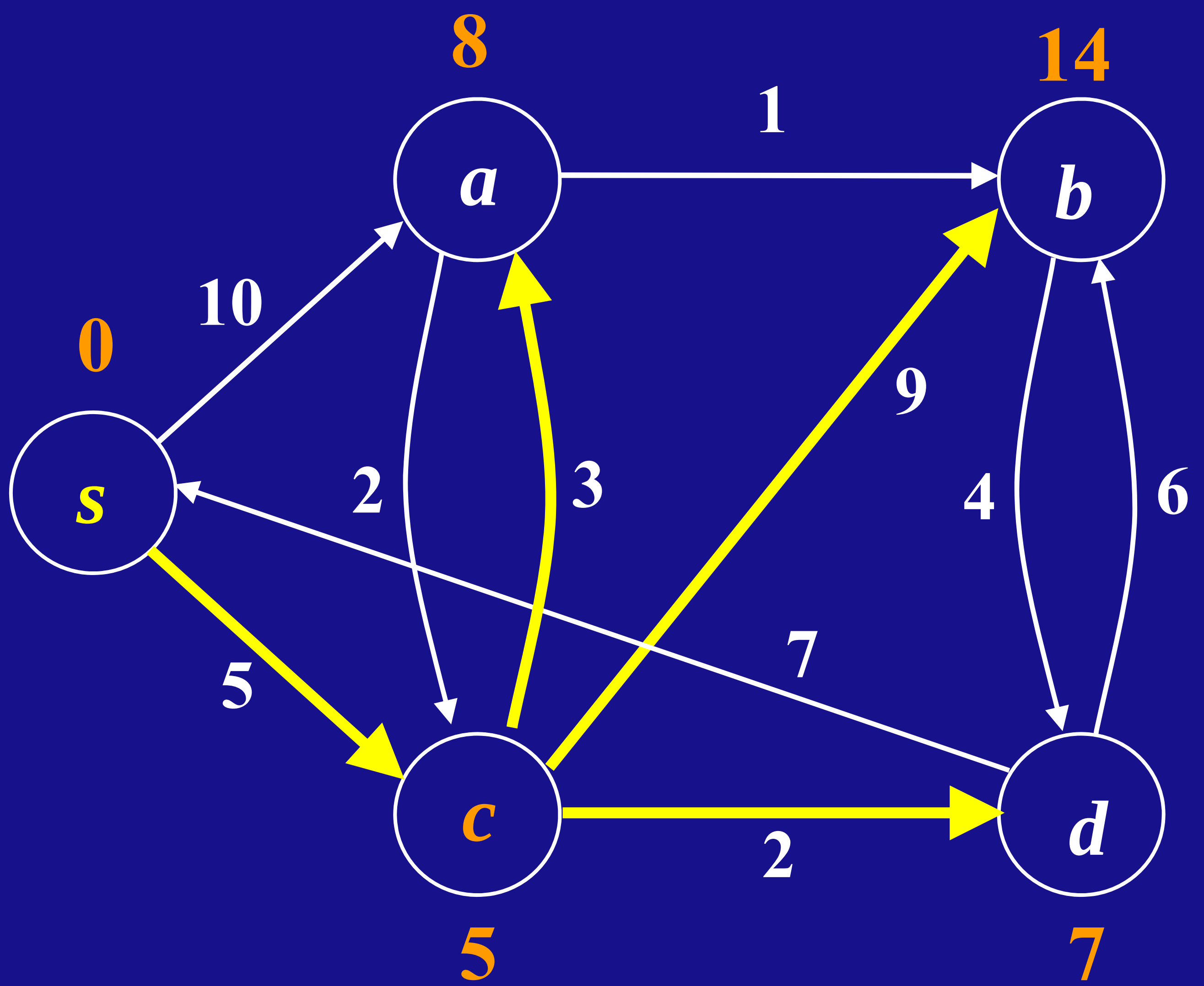
Example (Page-659): Dijkstra (2)



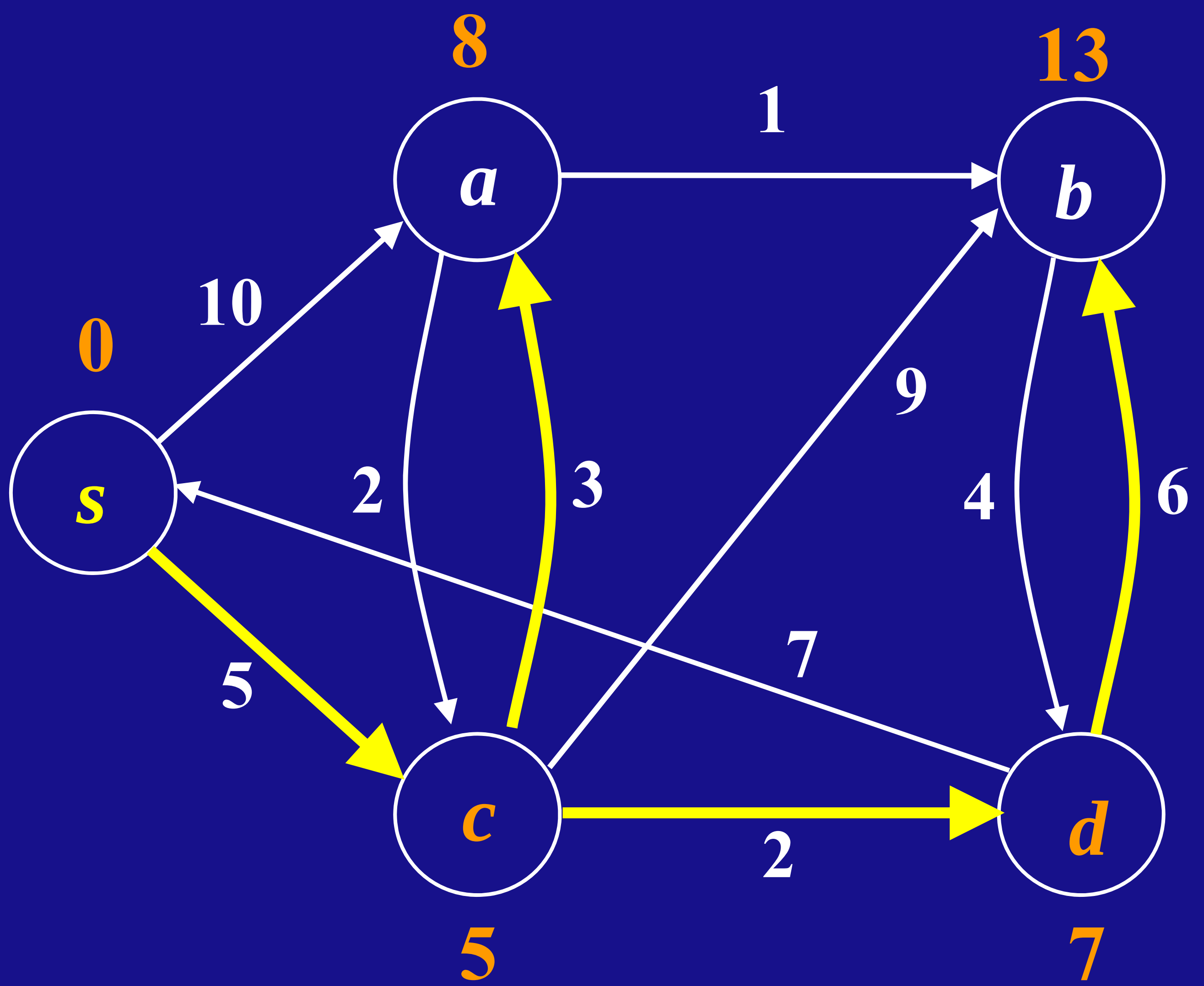
Example (Page-659): Dijkstra (3)



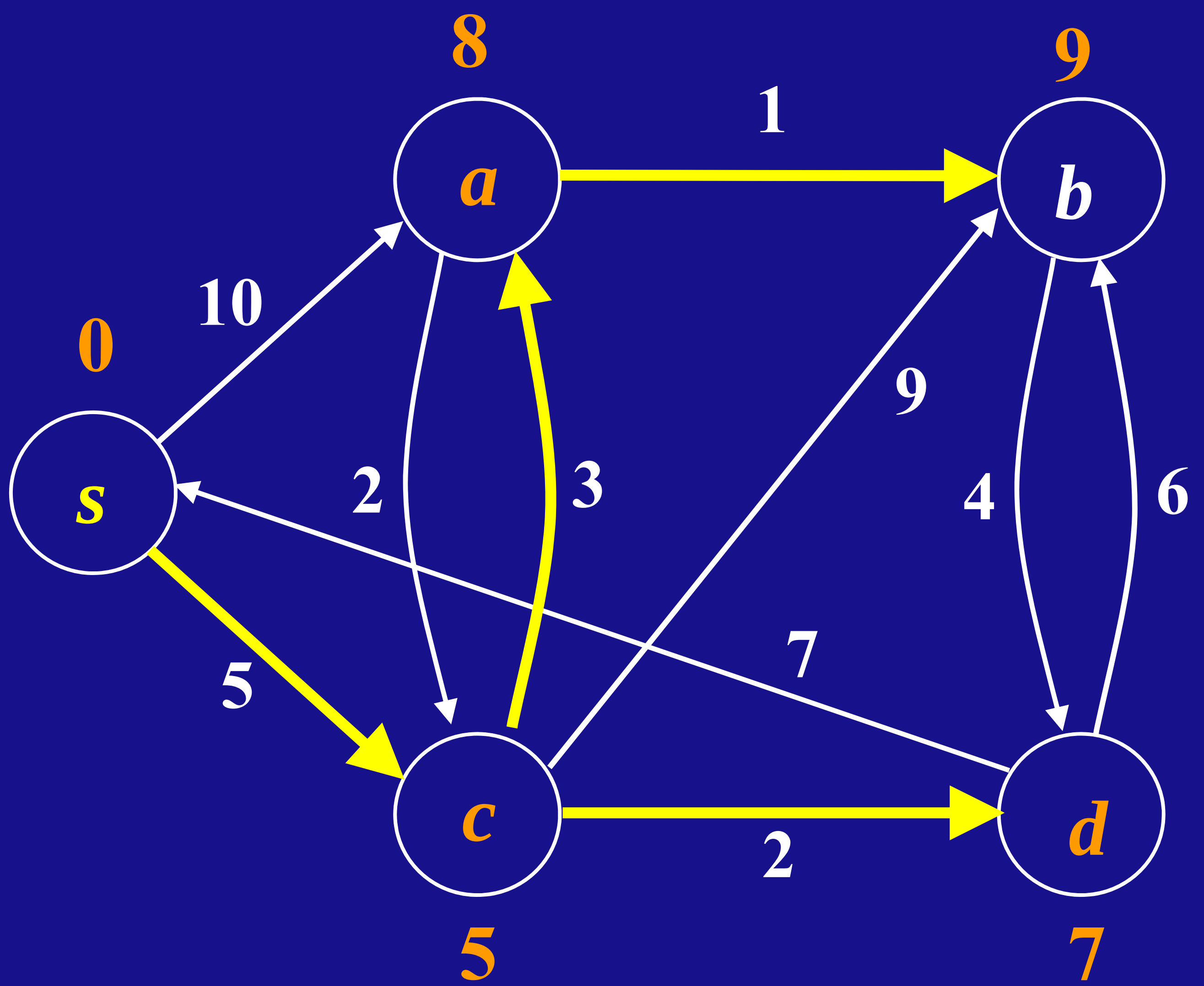
Example (Page-659): Dijkstra (4)



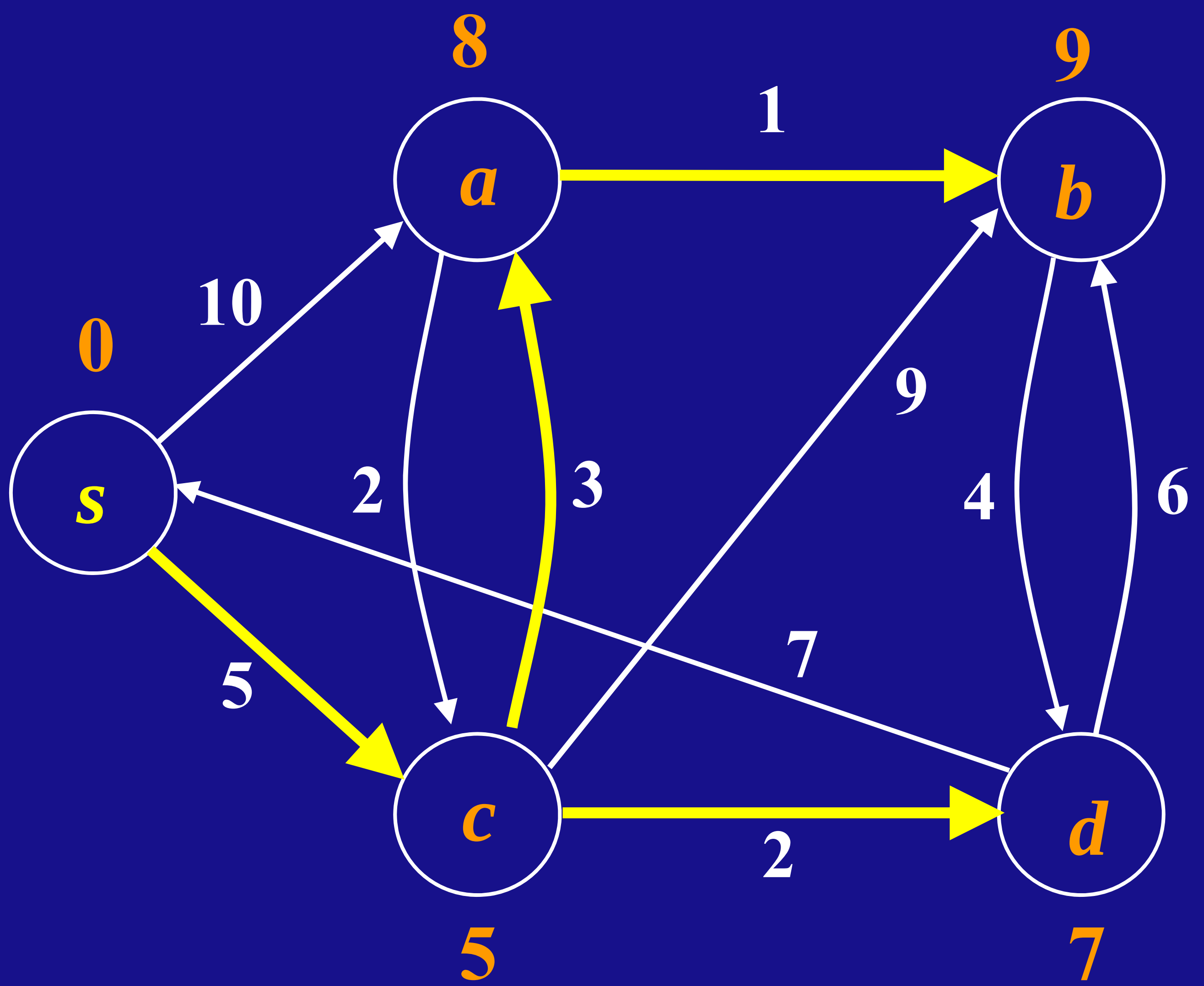
Example (Page-659): Dijkstra (5)



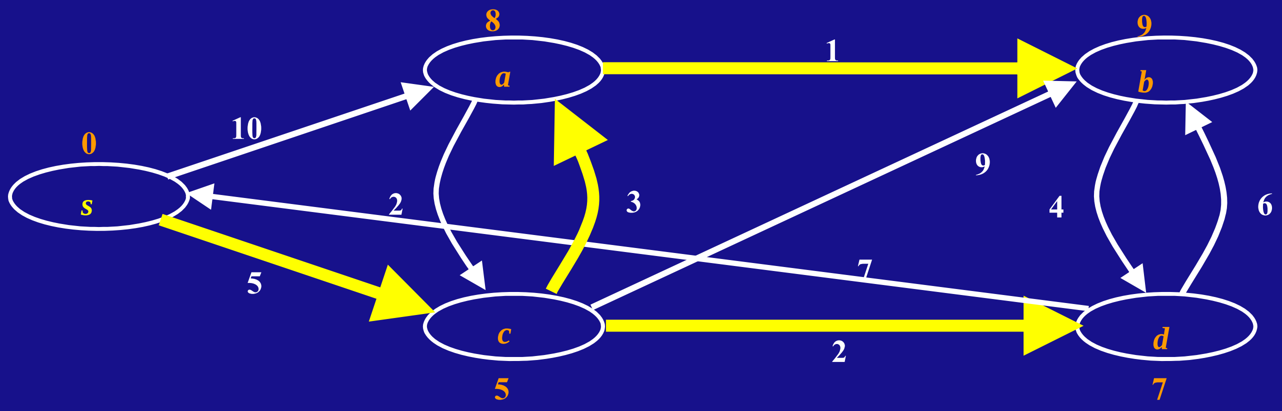
Example (Page-659): Dijkstra (6)



Example (Page-659): Dijkstra (7)

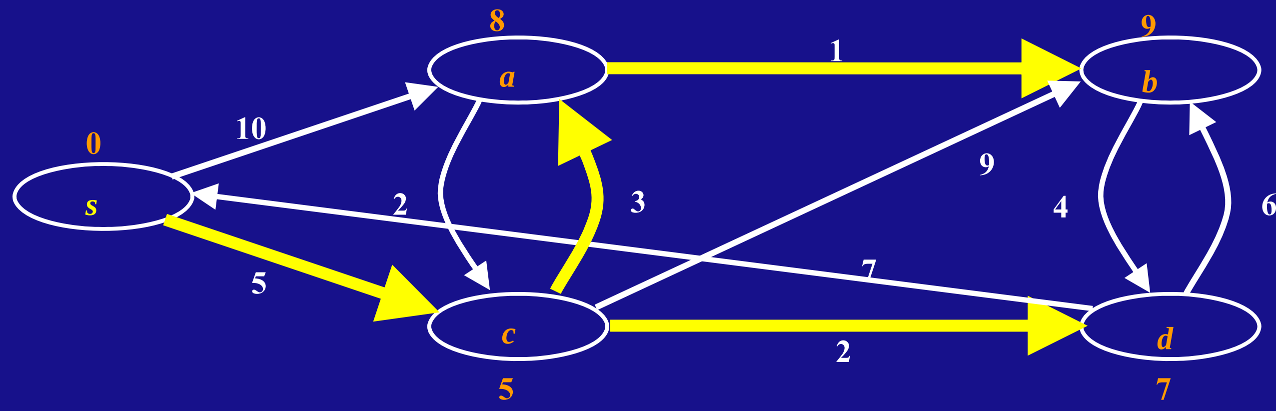


Example (Page-659): Dijkstra (8)



Selected Node	s	a	b	c	d
Initialize	0 / Nil	∞ / Nil	∞ / Nil	∞ / Nil	∞ / Nil
s	0 / Nil				
a	0 / Nil				
b	0 / Nil				
c	0 / Nil				
d	0 / Nil	8 / c	9 / a	5 / s	7/ c

Example (Page-659): Dijkstra (9)



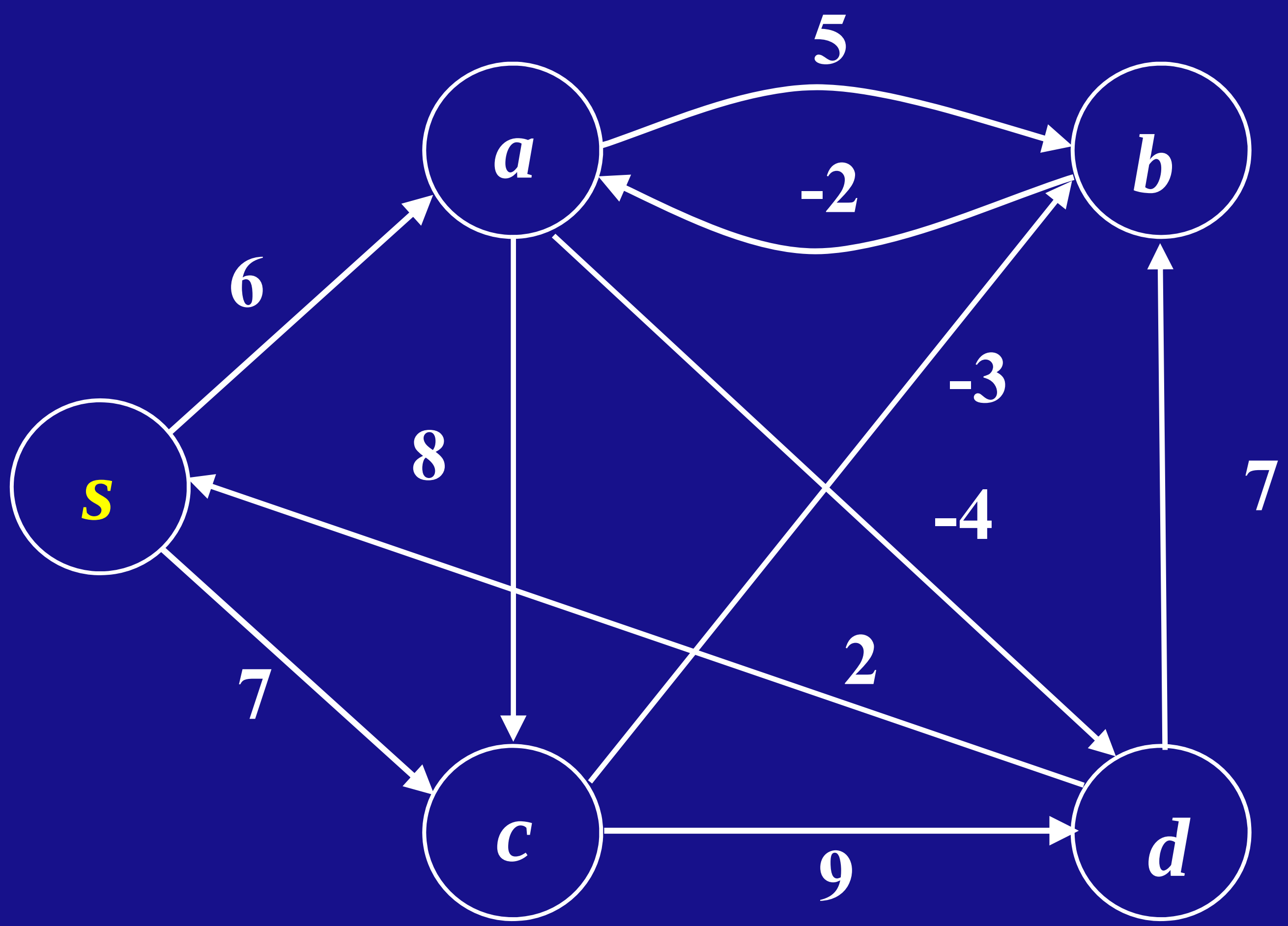
Source Node	a	b	c	d
s	8 / c	9 / a	5 / s	7/ c

Source Node	Destination Node	Paths	Cost
s	a	s → → a	
	b	s → → b	
	c	s → → c	
	d	s → → d	

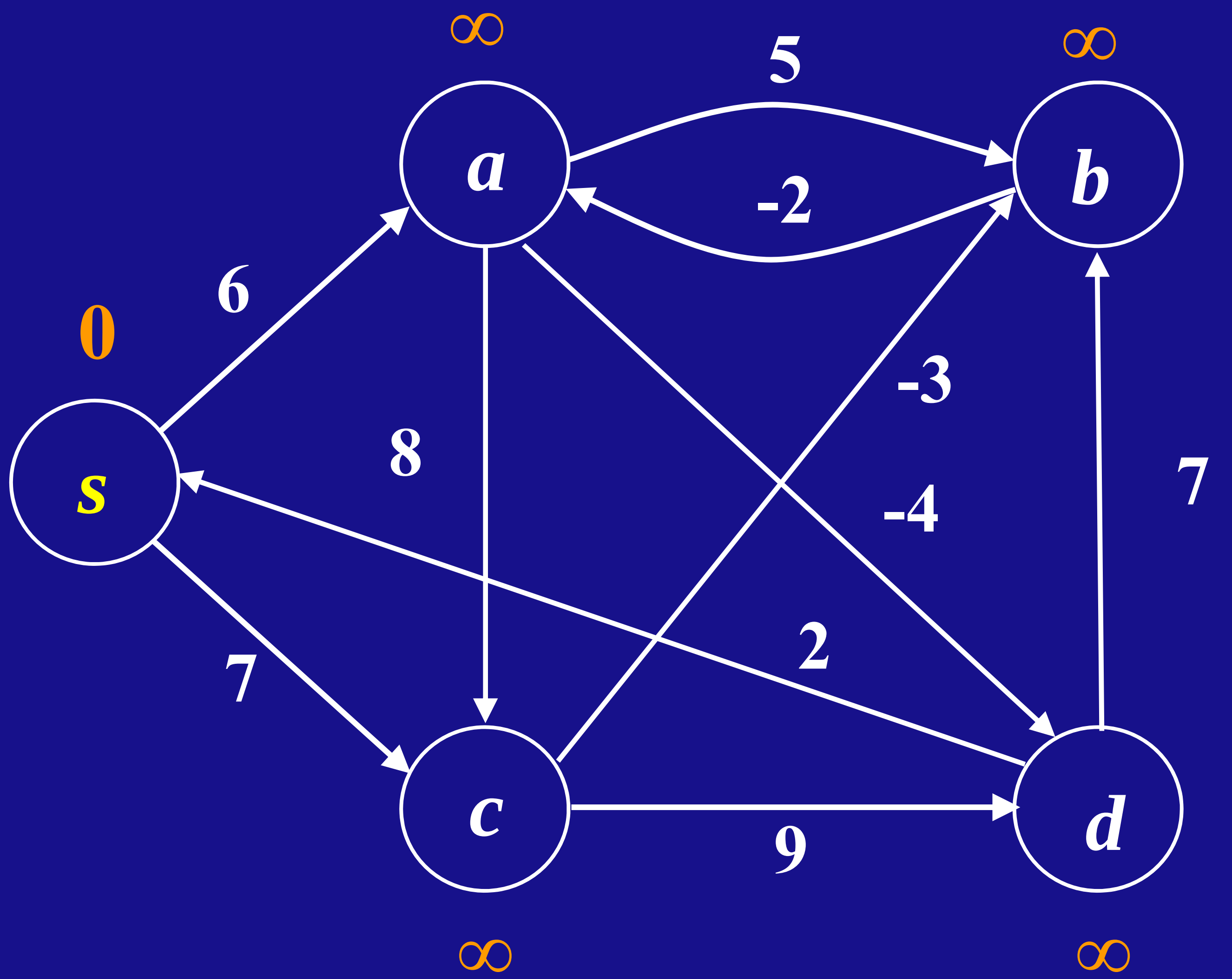
Algorithm (Page-658): Dijkstra

```
DIJKSTRA(G, w, s)
  INITIALIZE-SINGLE-SOURCE(G, s)
  S ← ∅
  Q ← V[G]
  while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
           S ← S ∪ {u}
           for each vertex v ∈ Adj[u]
               do RELAX(u, v, w)
```

Example (Page-652): Bellman-Ford (1)



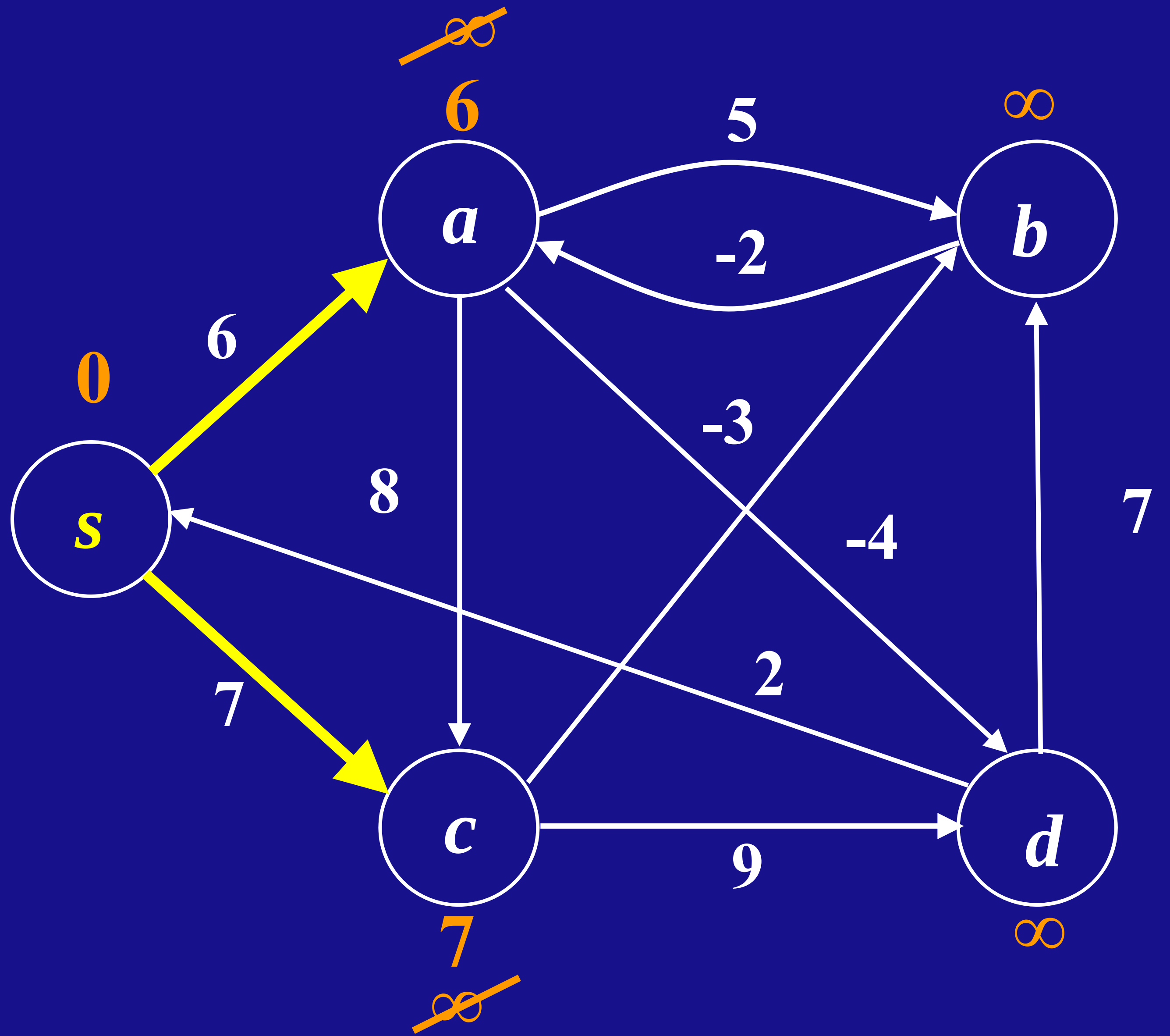
Example (Page-652): Bellman-Ford (2)



Edge order

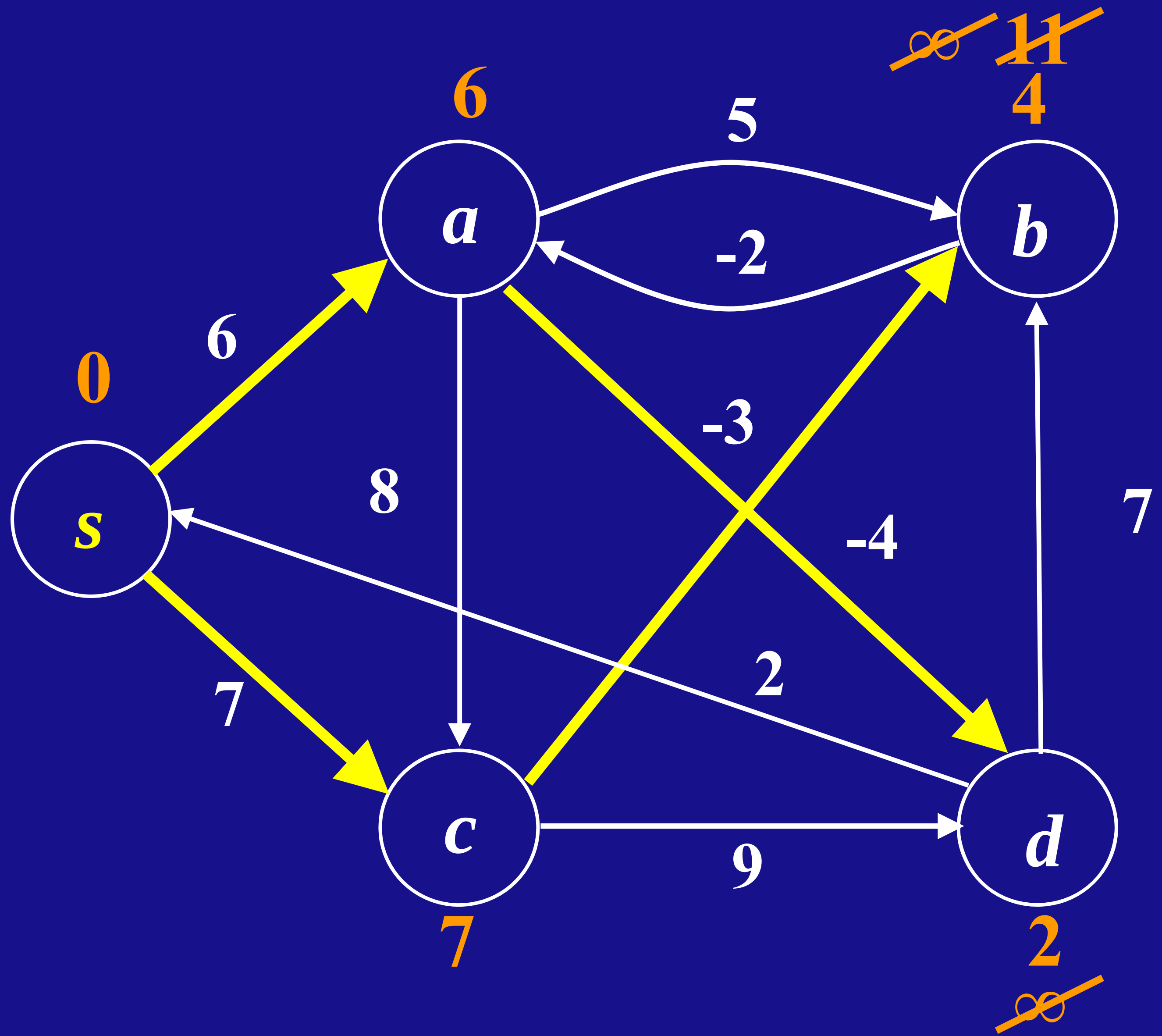
(a,b)
(a,c)
(a,d)
(b,a)
(c,b)
(c,d)
(d,s)
(d,b)
(s,a)
(s,c)

Example (Page-652): Bellman-Ford (3)



Edge order
(a,b)
(a,c)
(a,d)
(b,a)
(c,b)
(c,d)
(d,s)
(d,b)
(s,a)
(s,c)

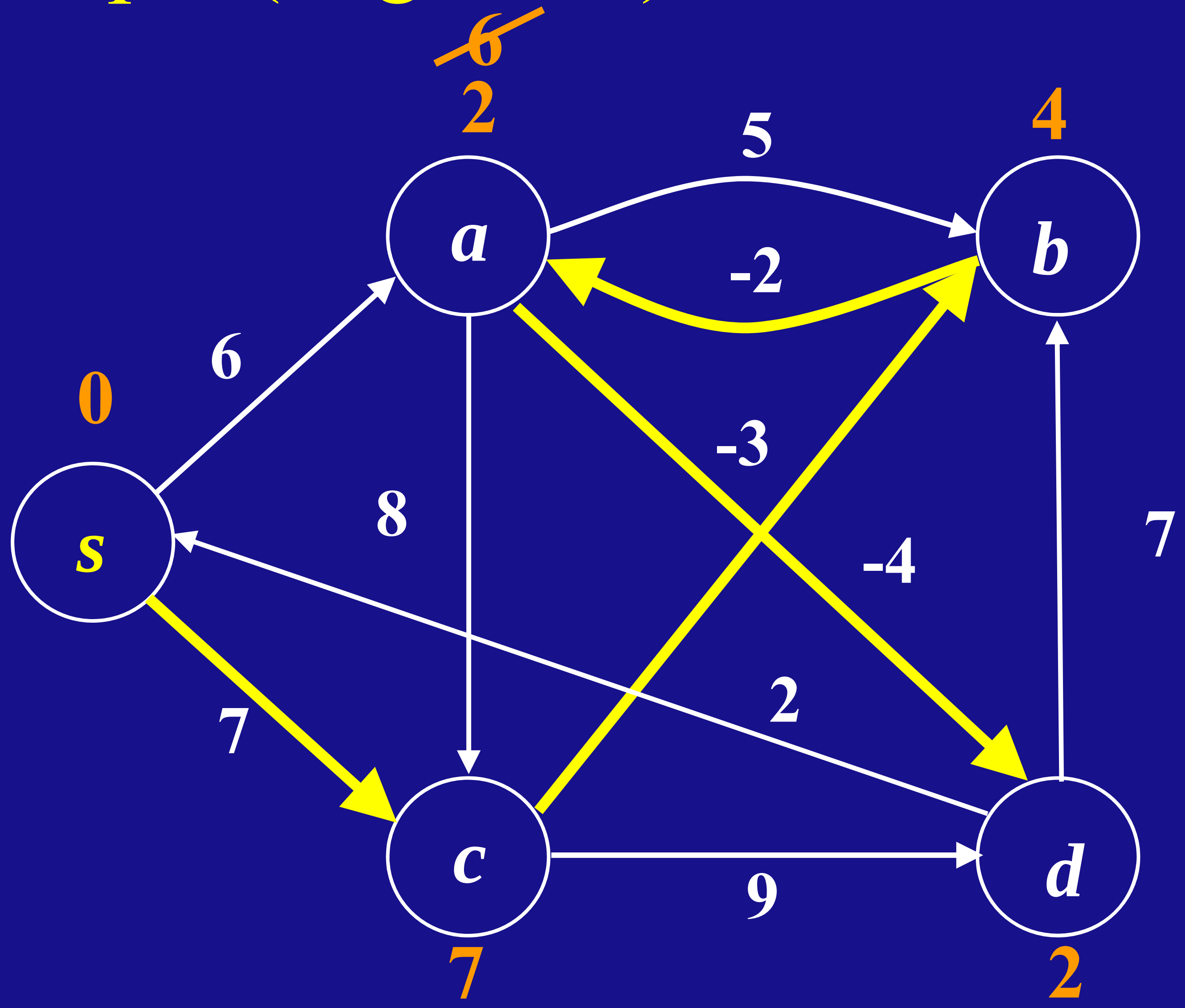
Example (Page-652): Bellman-Ford (4)



Edge order

(*a*,*b*)
(*a*,*c*)
(*a*,*d*)
(*b*,*a*)
(*c*,*b*)
(*c*,*d*)
(*d*,*s*)
(*d*,*b*)
(*s*,*a*)
(*s*,*c*)

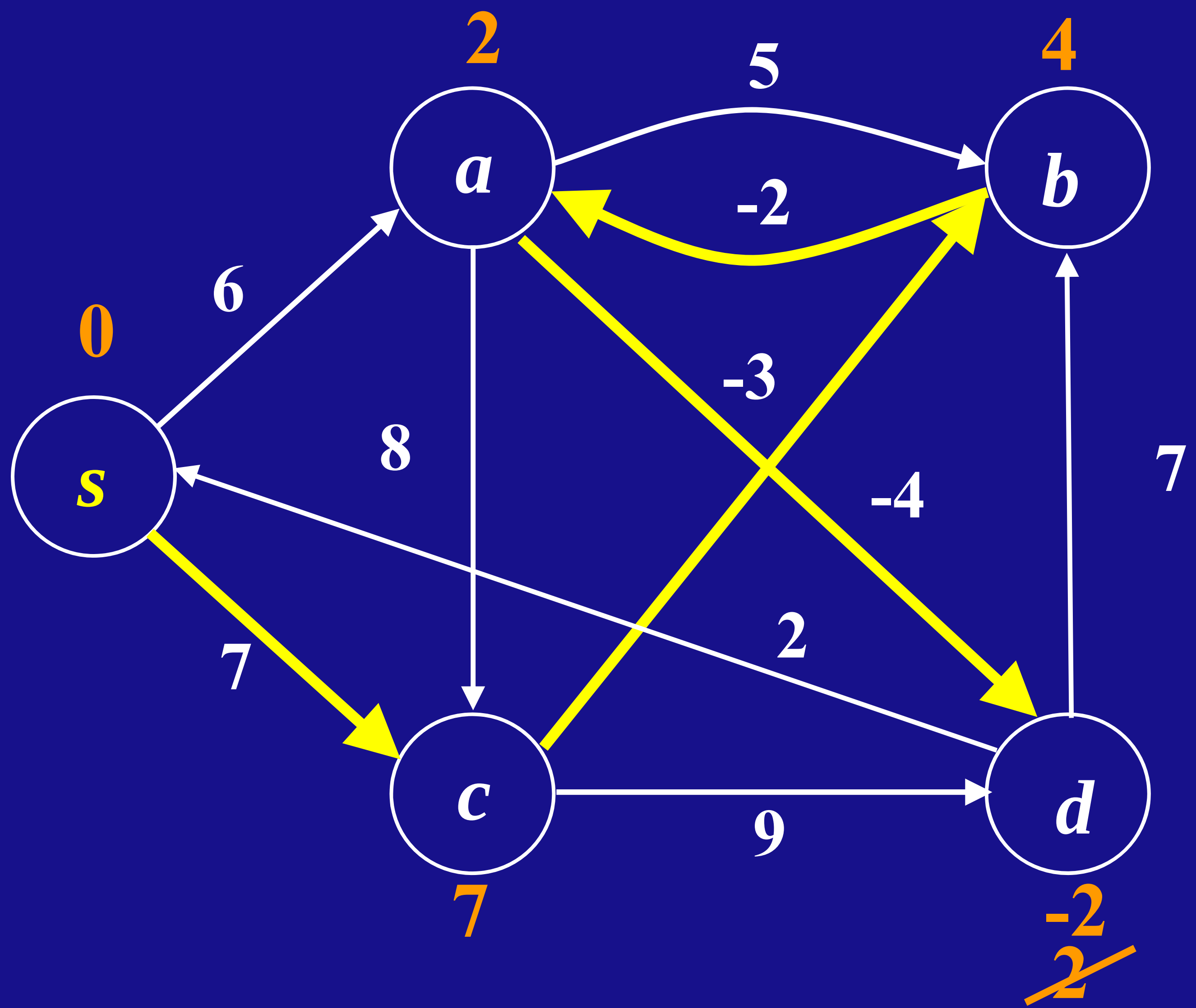
Example (Page-652): Bellman-Ford (5)



Edge order

(a,b)
(a,c)
(a,d)
(b,a)
(c,b)
(c,d)
(d,s)
(d,b)
(s,a)
(s,c)

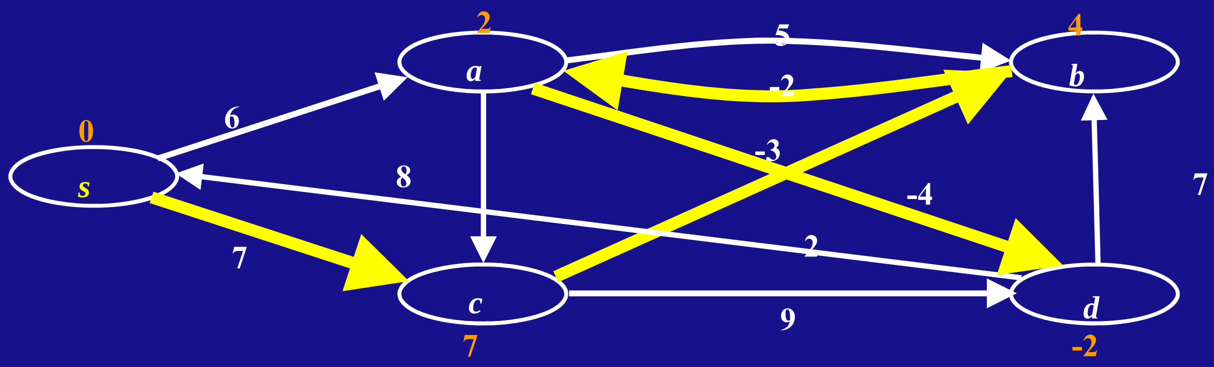
Example (Page-652): Bellman-Ford (6)



Edge order

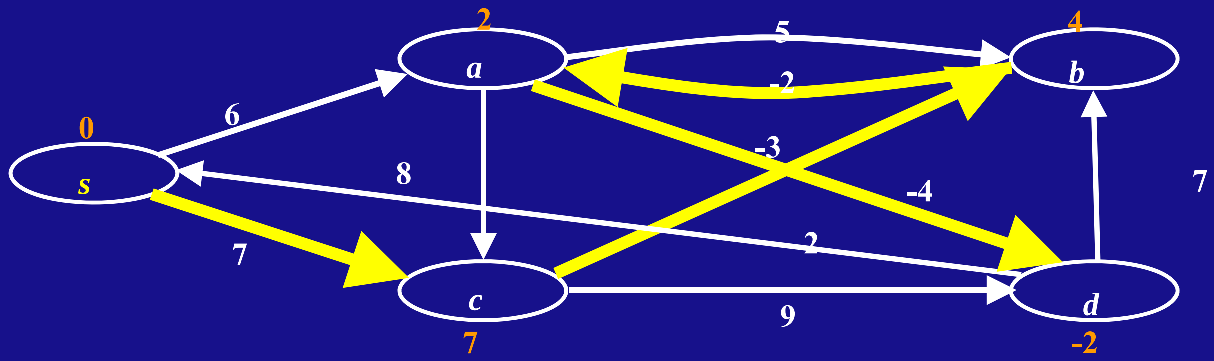
(a,b)
(a,c)
(a,d)
(b,a)
(c,b)
(c,d)
(d,s)
(d,b)
(s,a)
(s,c)

Example (Page-652): Bellman-Ford (7)



Pass Number	s	a	b	c	d
Initialize	0 / Nil	∞ / Nil	∞ / Nil	∞ / Nil	∞ / Nil
1	0 / Nil				
2	0 / Nil				
3	0 / Nil				
4	0 / Nil	2 / b	4 / c	7 / s	-2/ a
5	0 / Nil	2 / b	4 / c	7 / s	-2/ a

Example (Page-652): Bellman-Ford (8)



Source Node	a	b	c	d
s	2 / b	4 / c	7 / s	-2/ a

Source Node	Destination Node	Paths	Cost
s	a	s → → a	
	b	s → → b	
	c	s → → c	
	d	s → → d	

Algorithm (Page-651): Bellman-Ford

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

Thank You

Stay Safe