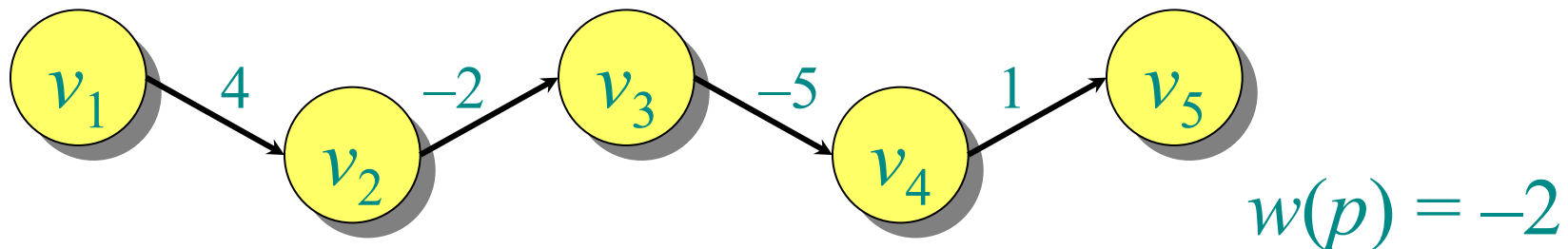# Shortest Paths 1

Paths in graphs

Consider a digraph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathbb{R}$. The **weight** of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is defined to be

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

Introduction to Algorithms

Paths in graphs

Consider a digraph $G = (V, E)$ with edge-weight function $w : E \to \mathbb{R}$. The **weight** of path $p = v_1 \to v_2 \to \cdots \to v_k$ is defined to be

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

Example:

$$w(p) = -2$$

Shortest paths

A ***shortest path*** from u to v is a path of minimum weight from u to v. The ***shortest-path weight*** from u to v is defined as

$$\delta(u, v) = \min\{w(p) : p \text{ is a path from } u \text{ to } v\}.$$

Note: $\delta(u, v) = \infty$ if no path from u to v exists.

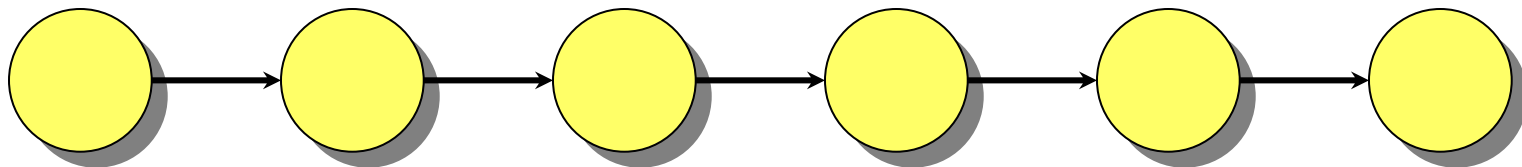# Optimal substructure

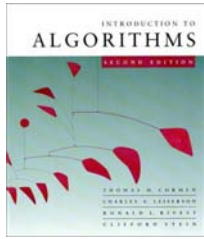**Theorem.** A subpath of a shortest path is a shortest path.

Optimal substructure

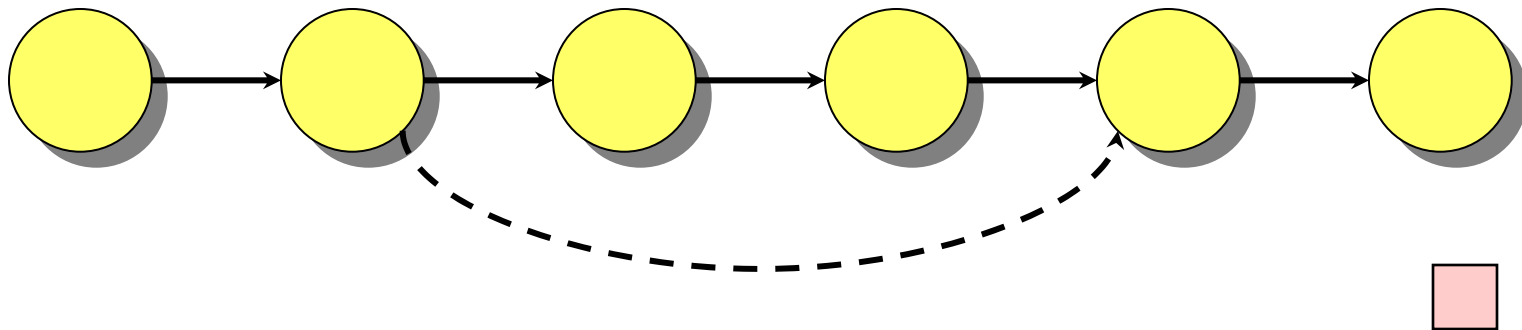**Theorem.** A subpath of a shortest path is a shortest path.
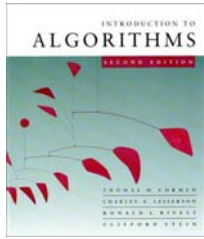
Proof. Cut and paste:

Optimal substructure

Theorem. A subpath of a shortest path is a shortest path.
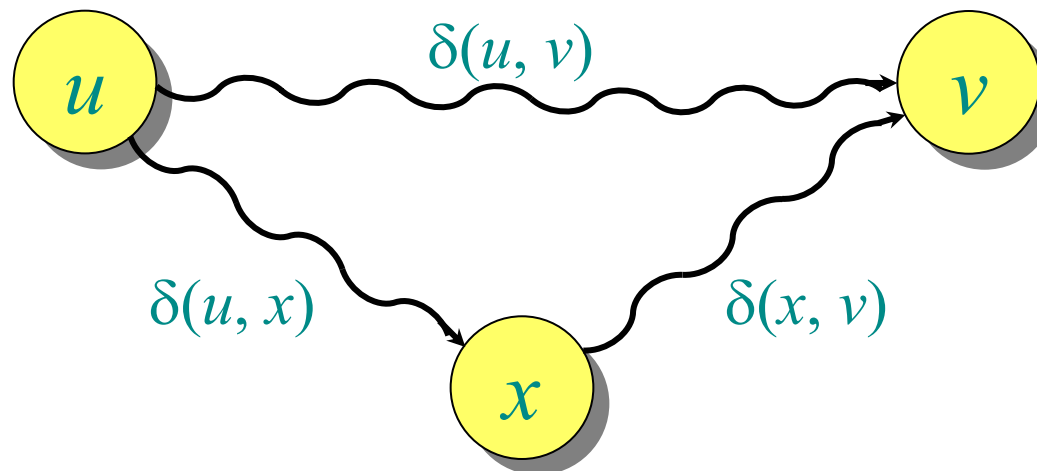
Proof. Cut and paste:

Introduction to Algorithms

Triangle inequality

Theorem. For all $u, v, x \in V$, we have
$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

Introduction to Algorithms

Triangle inequality

Theorem. For all $u, v, x \in V$, we have
$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

Proof.

Well-definedness of shortest paths

If a graph G contains a negative-weight cycle, then some shortest paths may not exist.

Introduction to Algorithms
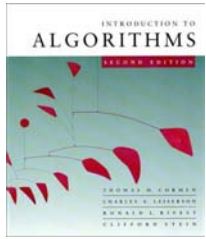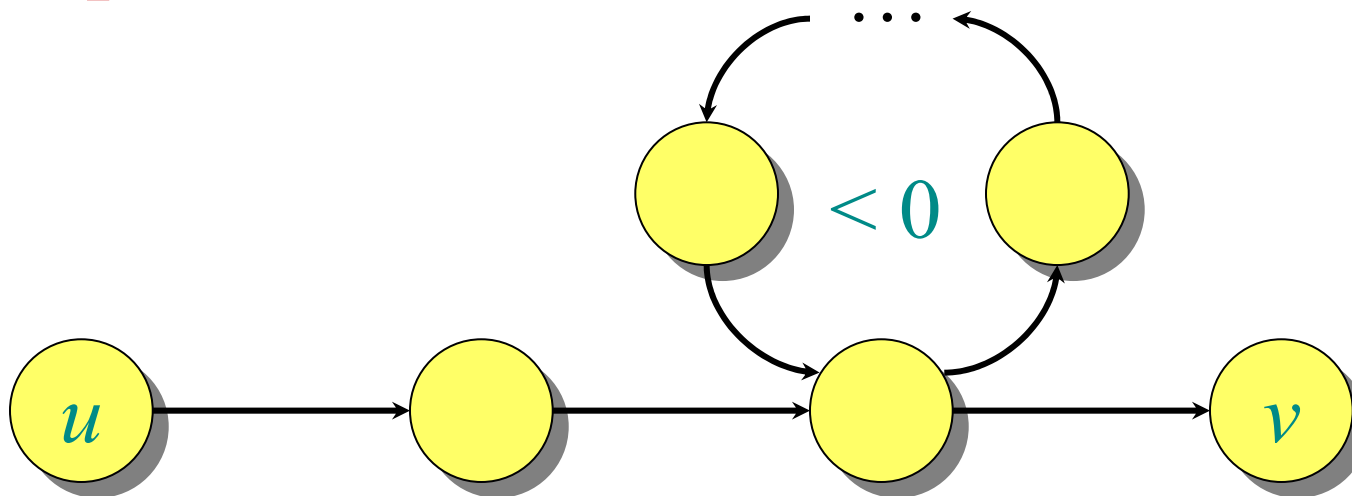
Well-definedness of shortest paths

If a graph G contains a negative-weight cycle, then some shortest paths may not exist.

Example:

Single-source shortest paths

Problem. From a given source vertex $s \in V$, find the shortest-path weights $\delta(s, v)$ for all $v \in V$.

If all edge weights $w(u, v)$ are *nonnegative*, all shortest-path weights must exist.

IDEA: Greedy.

1. Maintain a set S of vertices whose shortest-path distances from s are known.
2. At each step add to S the vertex $v \in V - S$ whose distance estimate from s is minimal.
3. Update the distance estimates of vertices adjacent to v.

Dijkstra's algorithm

$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
$S \leftarrow \varnothing$
$Q \leftarrow V$ $\triangleleft$ Q is a priority queue maintaining $V - S$

Introduction to Algorithms

Dijkstra's algorithm

$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
    do $d[v] \leftarrow \infty$
$S \leftarrow \varnothing$
$Q \leftarrow V$ $\triangleleft$ Q is a priority queue maintaining $V - S$
while $Q \neq \varnothing$
    do $u \leftarrow$ EXTRACT-MIN(Q)
        $S \leftarrow S \cup \{u\}$
        for each $v \in Adj[u]$
            do if $d[v] > d[u] + w(u, v)$
                then $d[v] \leftarrow d[u] + w(u, v)$

Dijkstra's algorithm

$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
    do $d[v] \leftarrow \infty$
$S \leftarrow \varnothing$
$Q \leftarrow V$ $\triangleleft$ Q is a priority queue maintaining $V - S$
while $Q \neq \varnothing$
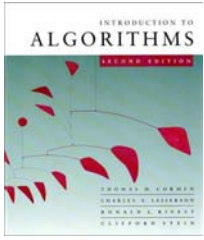   **do** $u \leftarrow \text{EXTRACT-MIN}(Q)$
        $S \leftarrow S \cup \{u\}$
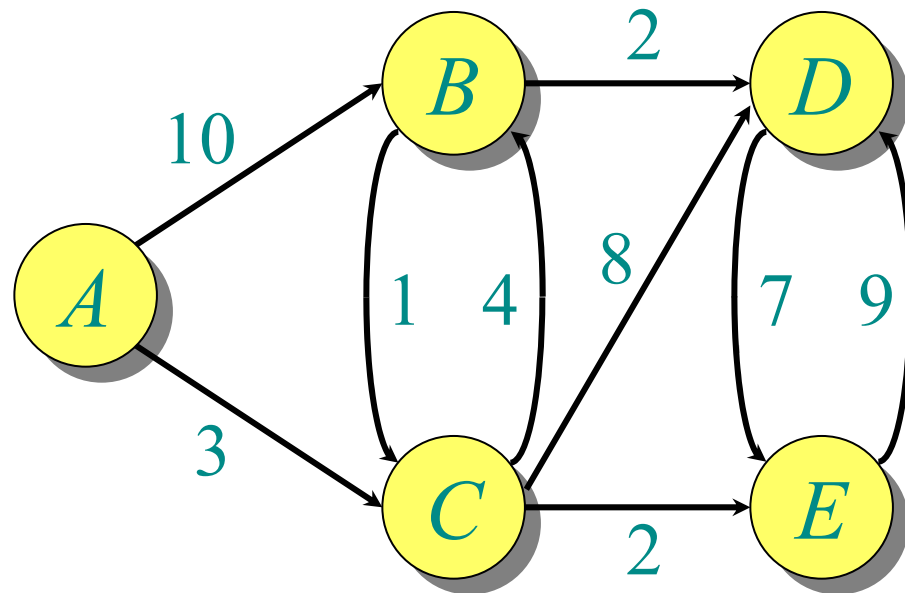   **for** each $v \in Adj[u]$
            do **if** $d[v] > d[u] + w(u, v)$ *relaxation*
                then $d[v] \leftarrow d[u] + w(u, v)$ *step*
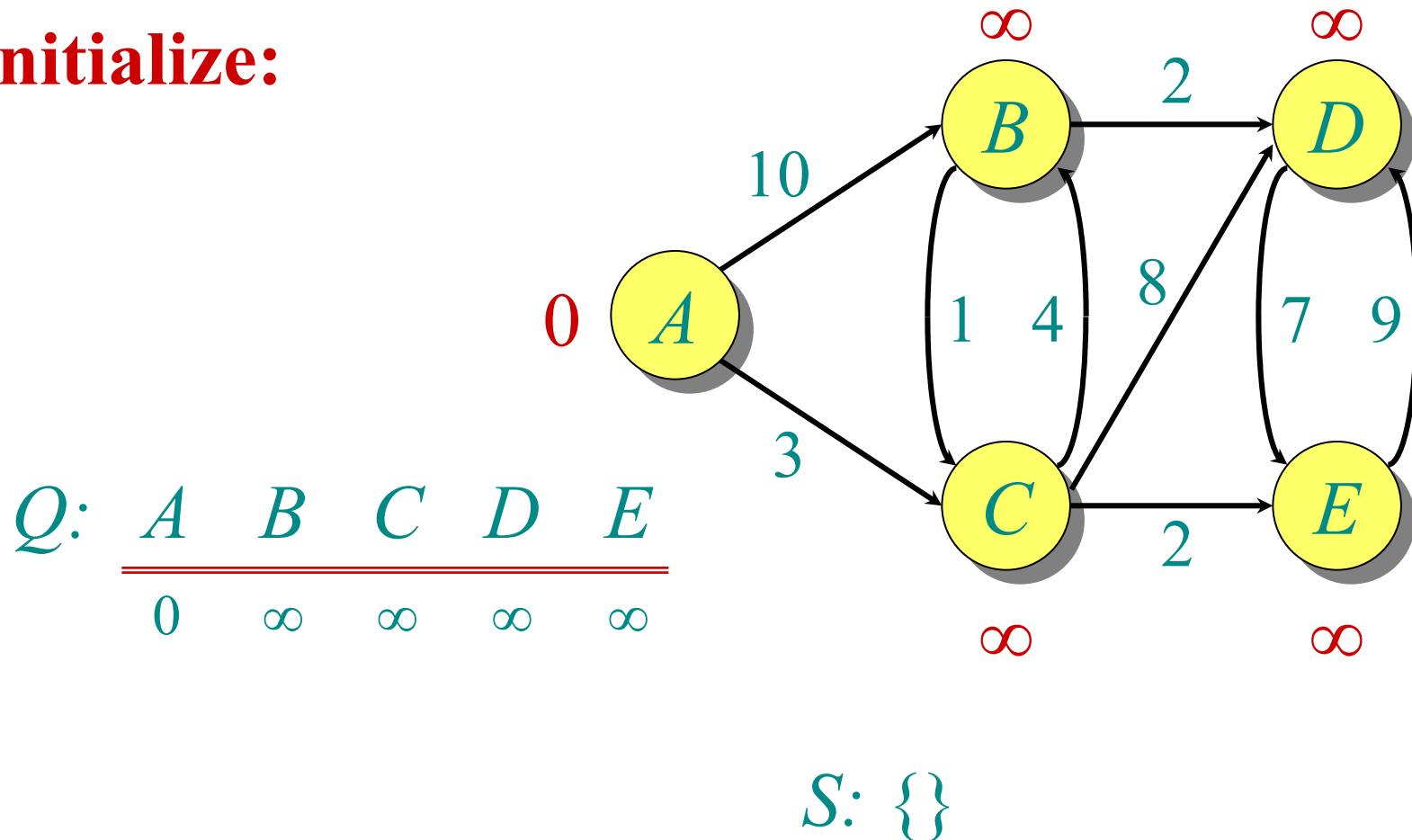
Implicit DECREASE-KEY

Introduction to Algorithms

Example of Dijkstra's algorithm

Graph with nonnegative edge weights:

Example of Dijkstra's algorithm

Initialize:

Q: A B C D E
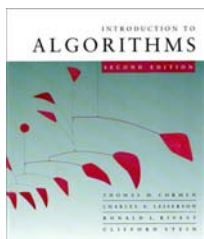
0 ∞ ∞ ∞ ∞

S: {}

Introduction to Algorithms

Example of Dijkstra's algorithm

"A" ← EXTRACT-MIN(Q):
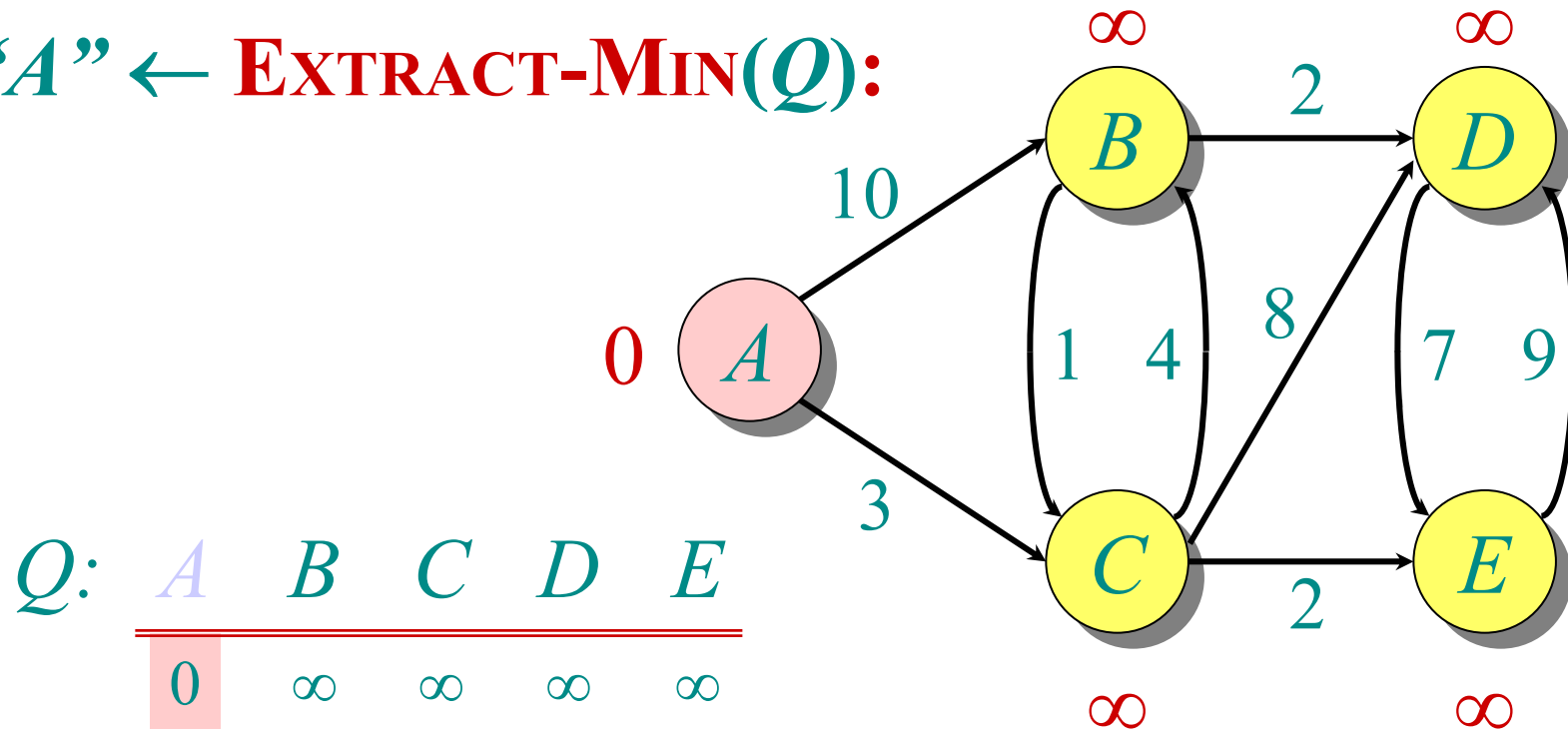
Q: A B C D E

A	B	C	D	E
0	∞	∞	∞	∞

$S: \{ A \}$

Introduction to Algorithms

Example of Dijkstra's algorithm

Relax all edges leaving A:

$$Q: \quad A \quad B \quad C \quad D \quad E$$

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞

$S: \{ A \}$

Introduction to Algorithms
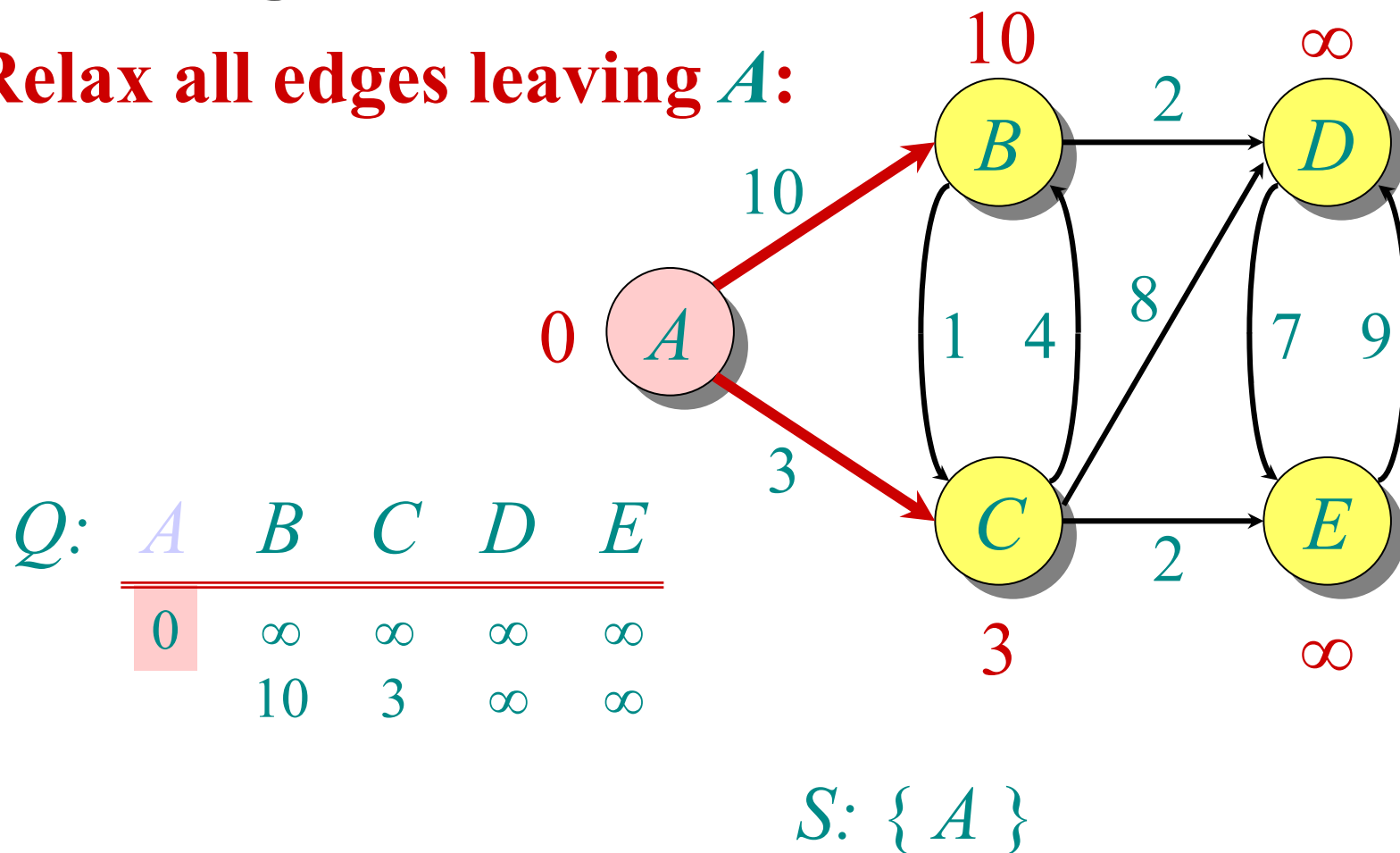
Example of Dijkstra's algorithm

"*C*" ← **EXTRACT-MIN(*Q*)**:
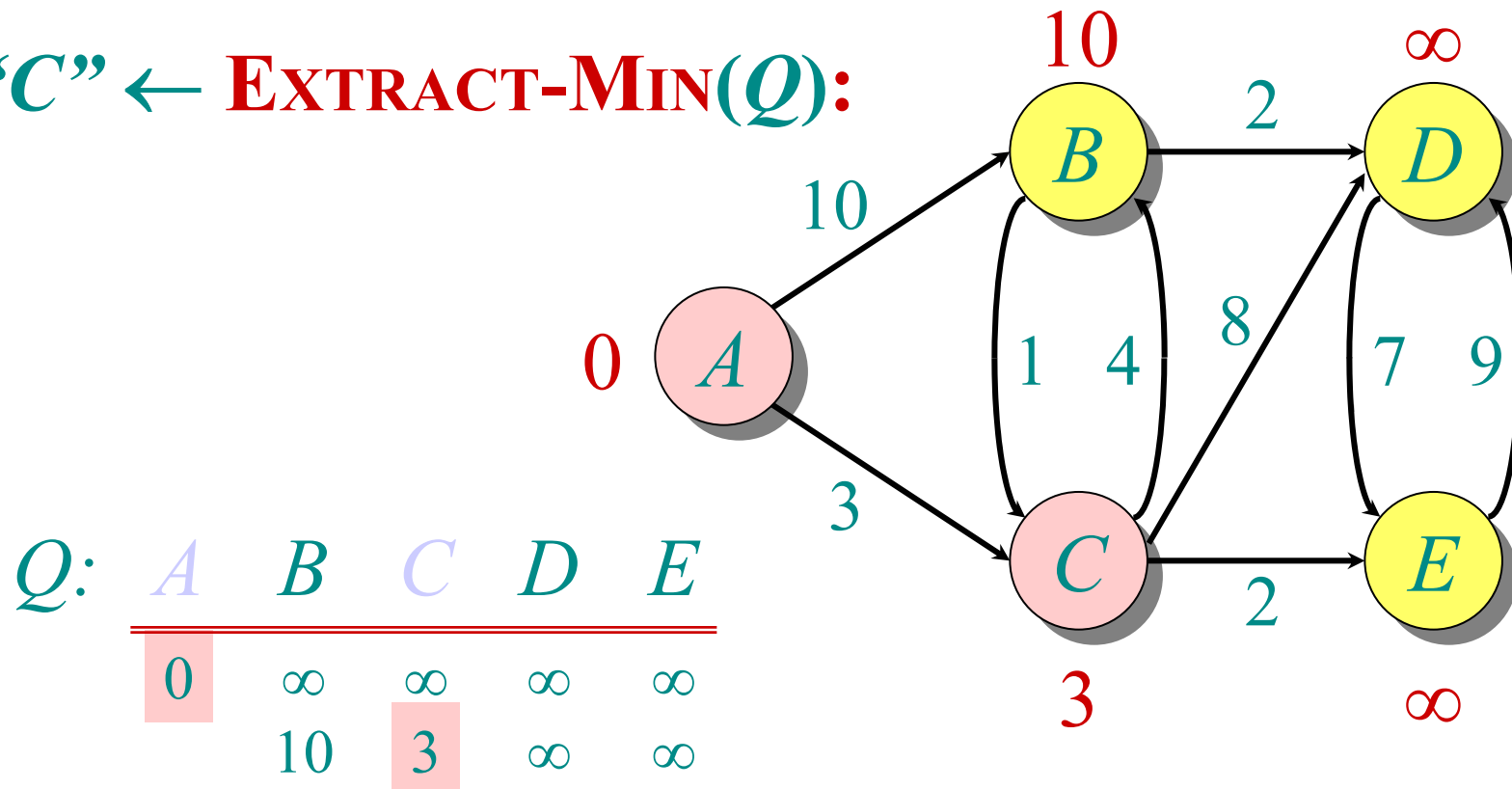
Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞

S: { A, C }

Introduction to Algorithms

Example of Dijkstra's algorithm

Relax all edges leaving C:

Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5

S: { A, C }

Introduction to Algorithms
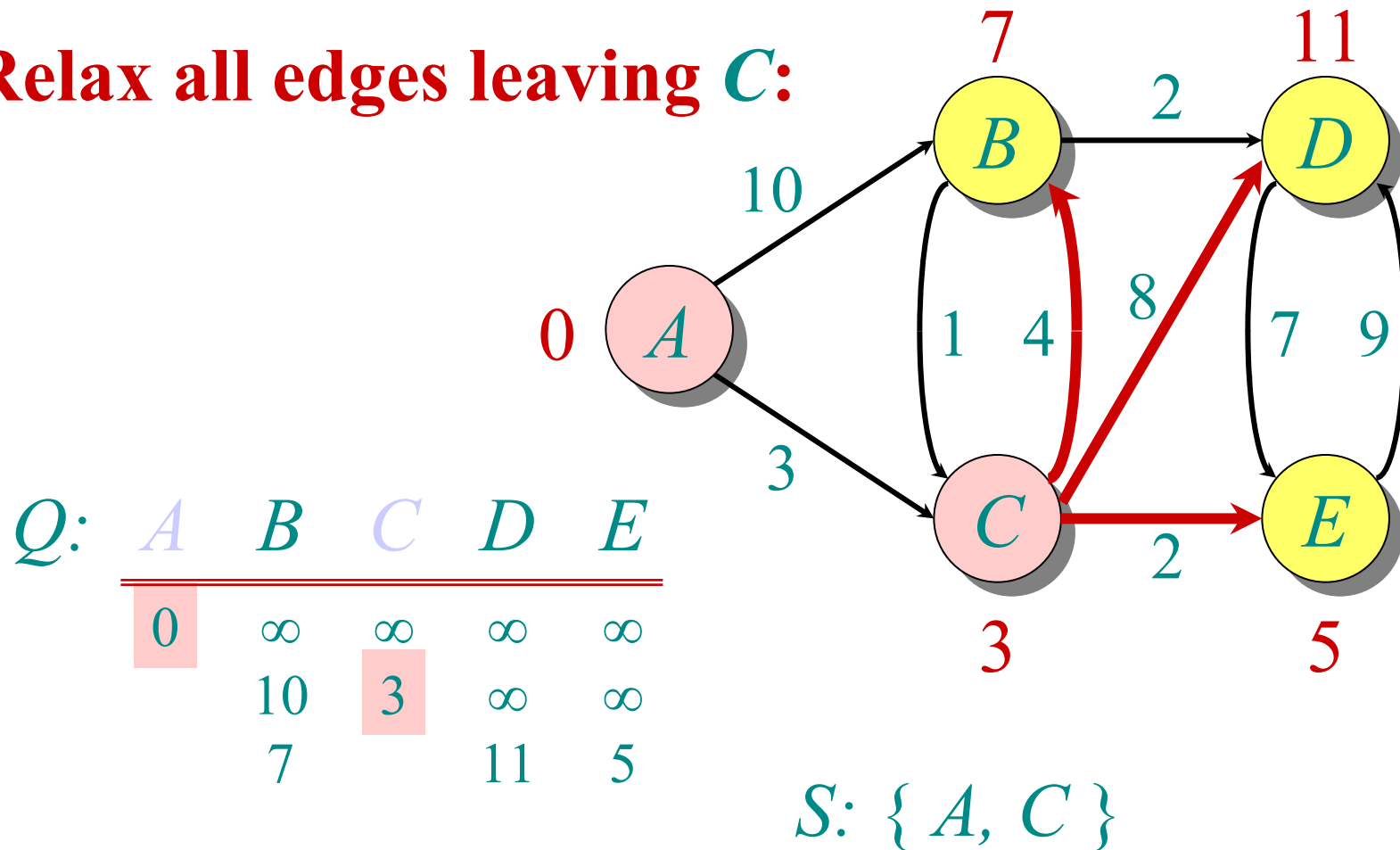
Example of Dijkstra's algorithm

"*E*" ← **EXTRACT-MIN(*Q*):**
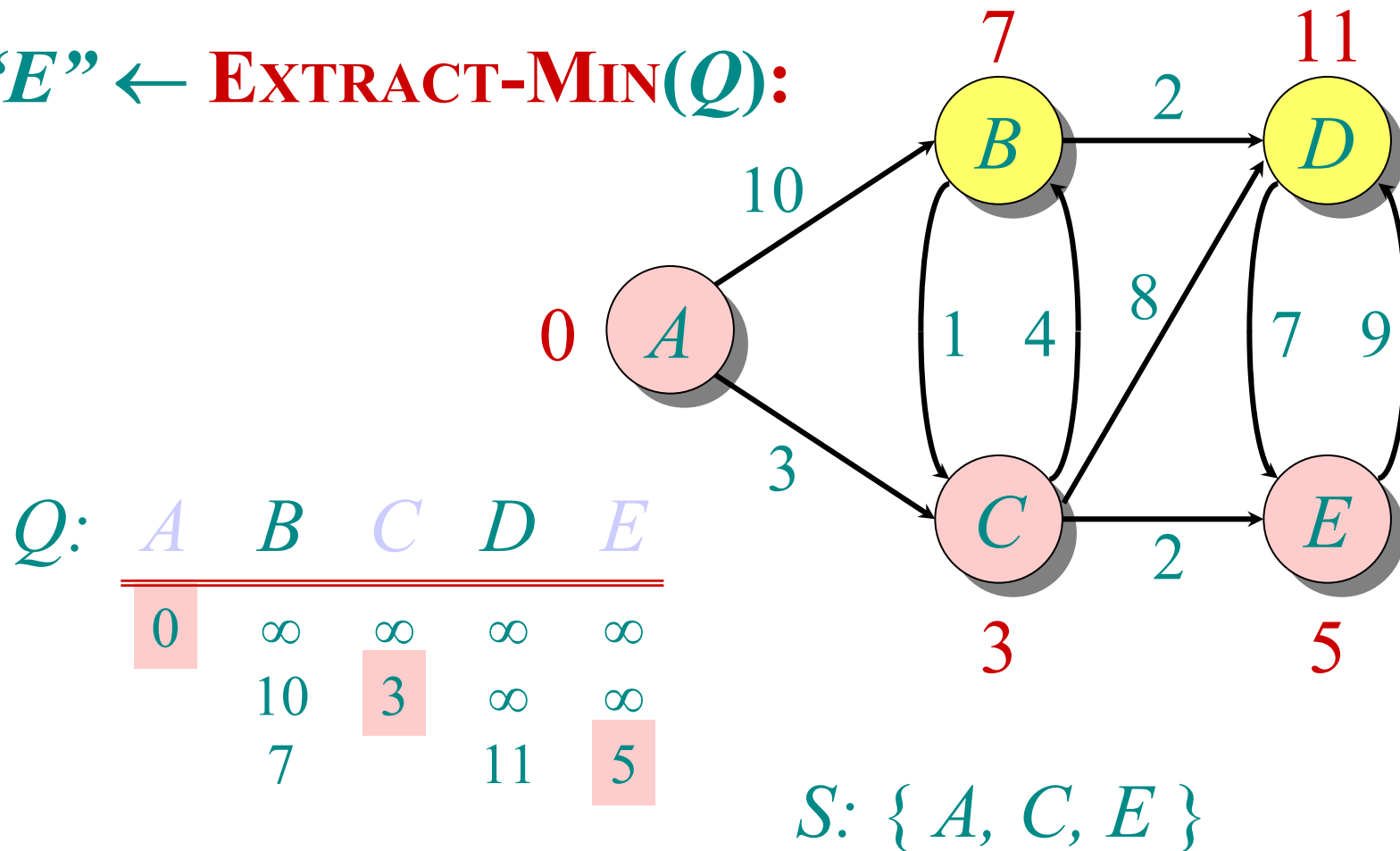
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C, E }

Introduction to Algorithms   November 1, 2004

Example of Dijkstra's algorithm

Relax all edges leaving E:

$$Q: \quad A \quad B \quad C \quad D \quad E$$

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	

$S: \{\, A, C, E \,\}$

Introduction to Algorithms
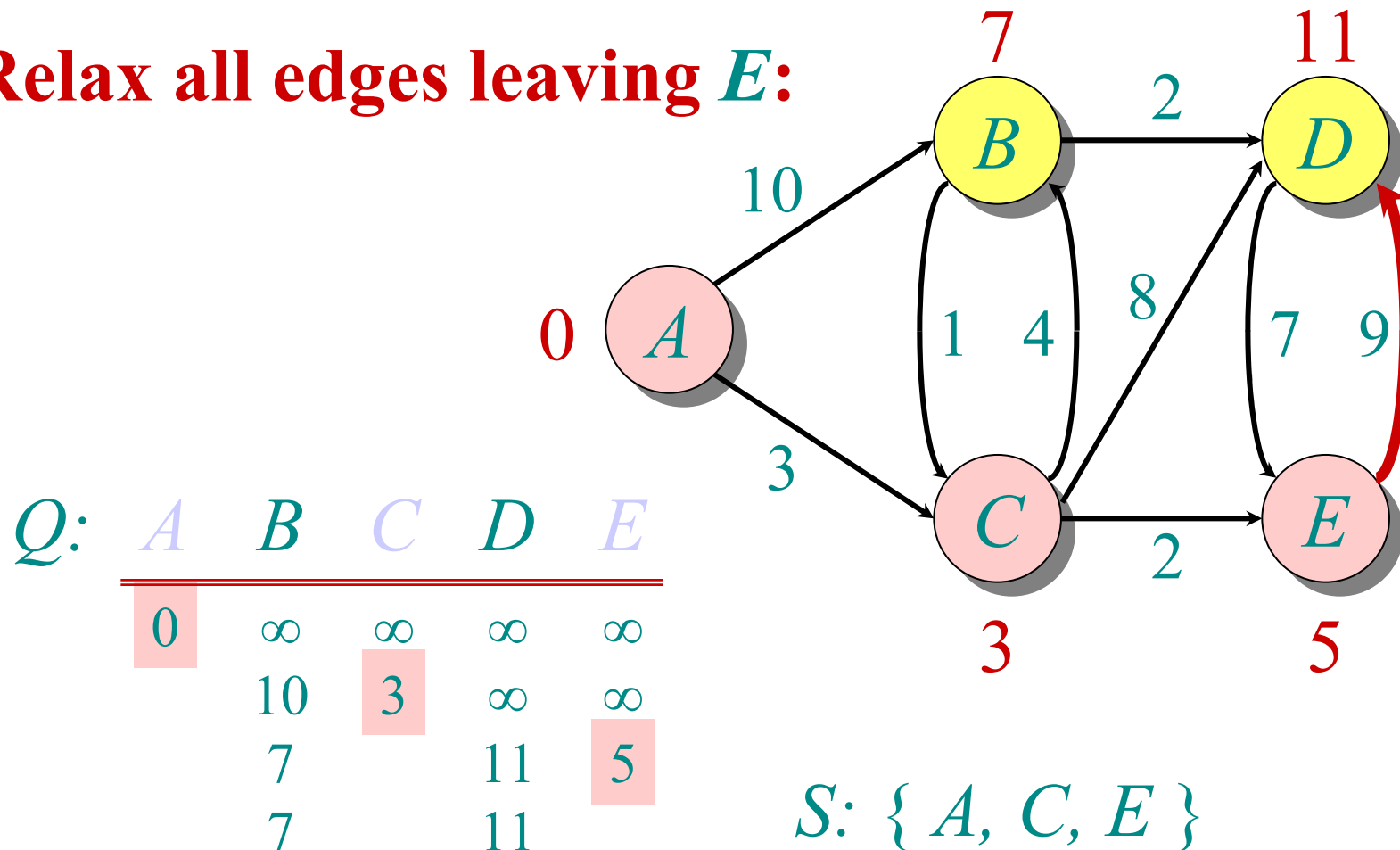
Example of Dijkstra's algorithm

"*B*" ← **EXTRACT-MIN**(*Q*):
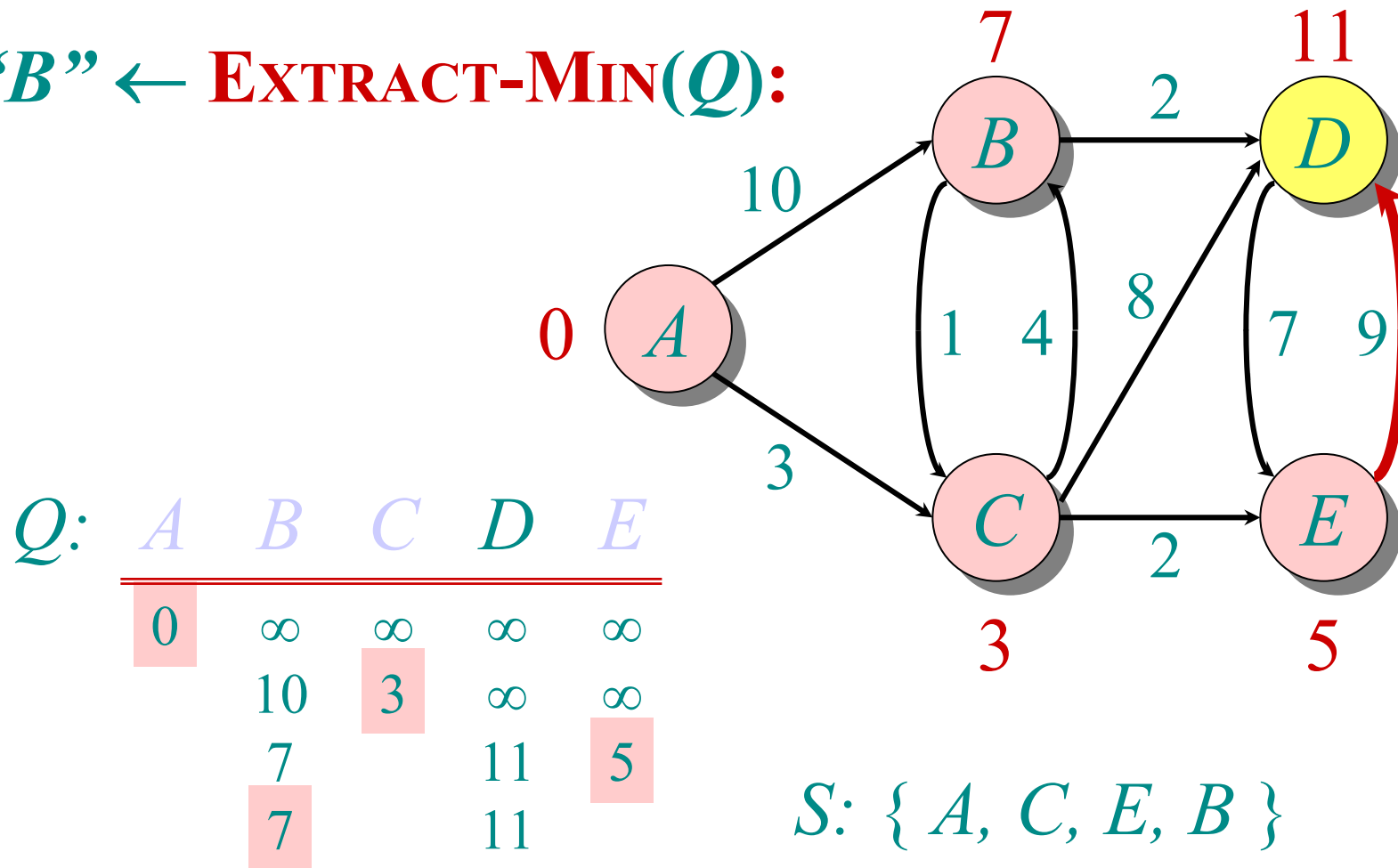
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { *A, C, E, B* }

Introduction to Algorithms

Example of Dijkstra's algorithm

Relax all edges leaving B:

Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	
			9	

$S: \{ A, C, E, B \}$

Introduction to Algorithms
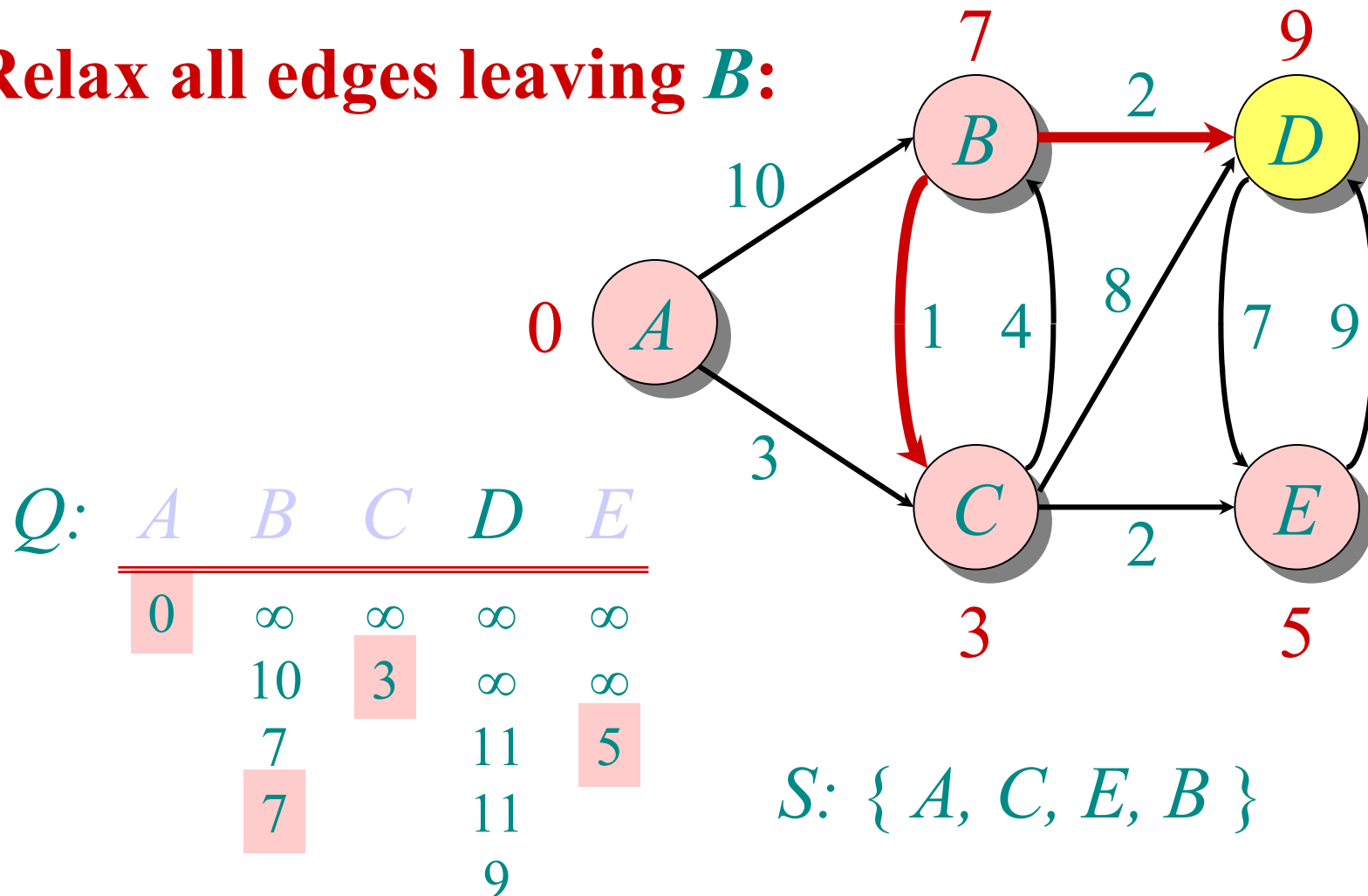
Example of Dijkstra's algorithm

"*D*" ← EXTRACT-MIN(*Q*):

Q:

A	B	C	D	E
0	∞	∞	∞	∞
	10	3	∞	∞
	7		11	5
	7		11	
			9	

$S: \{ A, C, E, B, D \}$

Introduction to Algorithms
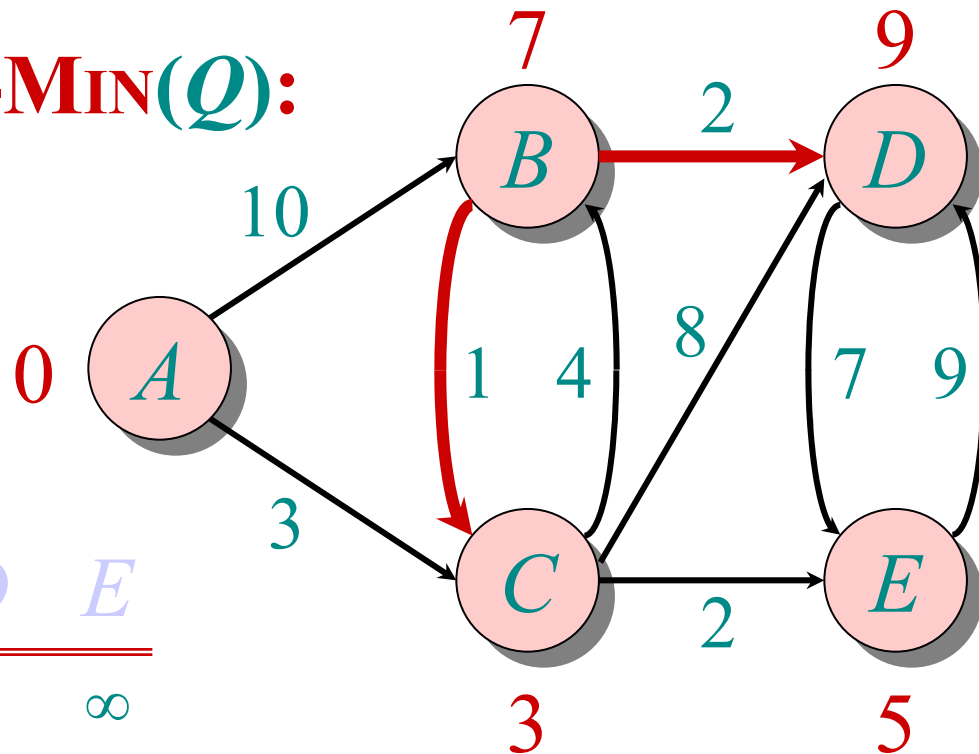
Correctness

Lemma. Let u be v's predecessor on a shortest path from s to v. Then, if $d[u] = \delta(s, u)$ and edge (u, v) is relaxed, we have $d[v] = \delta(s, v)$ after the relaxation.

Proof. Observe that $\delta(s, v) = \delta(s, u) + w(u, v)$. Suppose that $d[v] > \delta(s, v)$ before the relaxation. (Otherwise, we're done.) Then, the test $d[v] > d[u] + w(u, v)$ succeeds, because $d[v] > \delta(s, v) = \delta(s, u) + w(u, v) = d[u] + w(u, v)$, and the algorithm sets $d[v] = d[u] + w(u, v) = \delta(s, v)$. $\square$

Correctness

Theorem. Dijkstra's algorithm terminates with $d[v] = \delta(s, v)$ for all $v \in V$.

Proof. It suffices to show that $d[v] = \delta(s, v)$ for every $v \in V$ when v is added to S. Suppose u is the first vertex added to S for which $d[u] > \delta(s, u)$. Let y be the first vertex in $V - S$ along a shortest path from s to u, and let x be its predecessor:
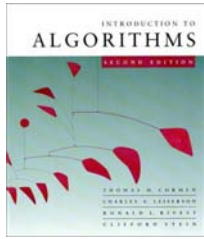
S, just before adding u.
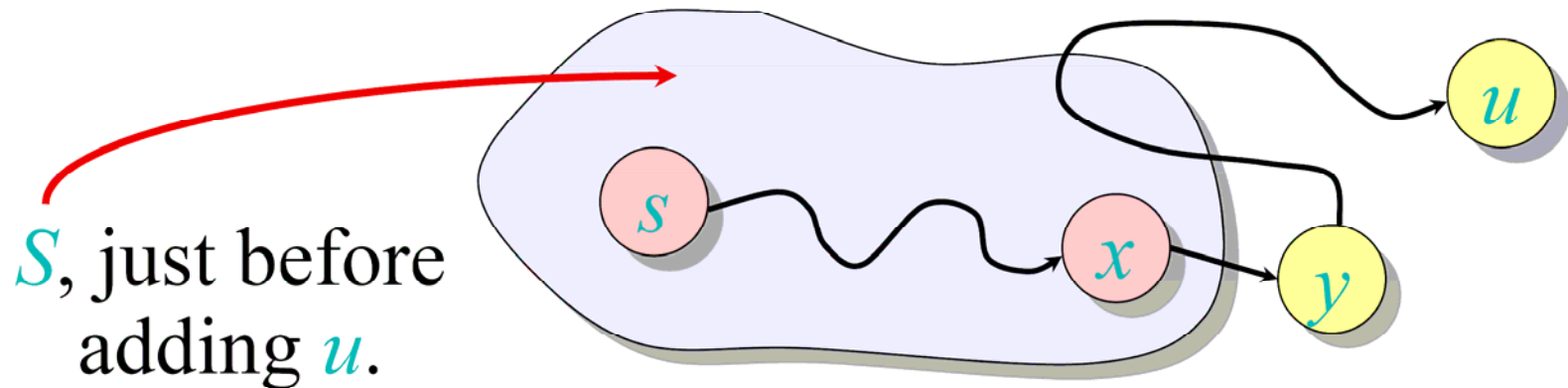
Correctness
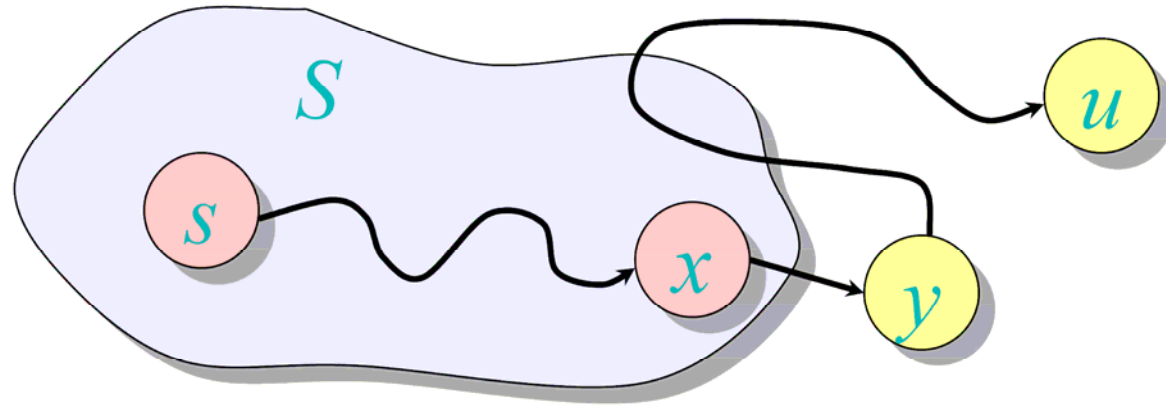
Since u is the first vertex violating the claimed invariant, we have $d[x] = \delta(s, x)$. When x was added to S, the edge (x, y) was relaxed, which implies that $d[y] = \delta(s, y) \le \delta(s, u) < d[u]$. But, $d[u] \le d[y]$ by our choice of u. Contradiction. □
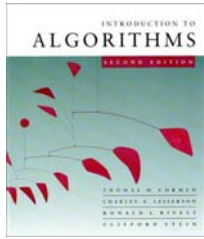
Analysis of Dijkstra

while $Q \neq \varnothing$
    do $u \leftarrow \text{EXTRACT-MIN}(Q)$
    $S \leftarrow S \cup \{u\}$
    for each $v \in Adj[u]$
        do if $d[v] > d[u] + w(u, v)$
            then $d[v] \leftarrow d[u] + w(u, v)$

Analysis of Dijkstra

$|V|$
times
$\Bigg\{$

while $Q \neq \varnothing$
    do $u \leftarrow \text{EXTRACT-MIN}(Q)$
        $S \leftarrow S \cup \{u\}$
        for each $v \in Adj[u]$
            do if $d[v] > d[u] + w(u, v)$
                then $d[v] \leftarrow d[u] + w(u, v)$

Introduction to Algorithms

Analysis of Dijkstra

$|V|$ times

$degree(u)$ times

$$\textbf{while } Q \neq \varnothing$$
$$\textbf{do } u \leftarrow \text{E{\scriptsize XTRACT}-M{\scriptsize IN}}(Q)$$
$$S \leftarrow S \cup \{u\}$$
$$\textbf{for } \text{each } v \in Adj[u]$$
$$\textbf{do if } d[v] > d[u] + w(u, v)$$
$$\textbf{then } d[v] \leftarrow d[u] + w(u, v)$$

Introduction to Algorithms

Analysis of Dijkstra

$|V|$ times
$\left\{\rule{0pt}{5em}\right.$

while $Q \neq \varnothing$
   do $u \leftarrow$ EXTRACT-MIN(Q)
   $S \leftarrow S \cup \{u\}$

$degree(u)$ times
$\left\{\rule{0pt}{3em}\right.$

     for each $v \in Adj[u]$
      do if $d[v] > d[u] + w(u, v)$
        then $d[v] \leftarrow d[u] + w(u, v)$

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY's.

Introduction to Algorithms

Analysis of Dijkstra

$$\text{while } Q \neq \varnothing$$
$$\text{do } u \leftarrow \text{EXTRACT-MIN}(Q)$$
$$S \leftarrow S \cup \{u\}$$
$$\text{for each } v \in Adj[u]$$
$$\text{do if } d[v] > d[u] + w(u, v)$$
$$\text{then } d[v] \leftarrow d[u] + w(u, v)$$

$|V|$ times

$degree(u)$ times

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY's.

Time $= \Theta(V \cdot T_{\text{EXTRACT-MIN}} + E \cdot T_{\text{DECREASE-KEY}})$

Note: Same formula as in the analysis of Prim's minimum spanning tree algorithm.

Analysis of Dijkstra (continued)

$$\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total

Introduction to Algorithms

Analysis of Dijkstra (continued)

$$Time = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$

Introduction to Algorithms

Analysis of Dijkstra (continued)

$$\text{Time} = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

Introduction to Algorithms

Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$.
Can Dijkstra's algorithm be improved?

Introduction to Algorithms

Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$.
Can Dijkstra's algorithm be improved?

- Use a simple FIFO queue instead of a priority queue.

Introduction to Algorithms

Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$. Can Dijkstra's algorithm be improved?

- Use a simple FIFO queue instead of a priority queue.

Breadth-first search

> **while** $Q \neq \varnothing$
> > **do** $u \leftarrow$ DEQUEUE(Q)
> > > **for** each $v \in Adj[u]$
> > > > **do if** $d[v] = \infty$
> > > > > **then** $d[v] \leftarrow d[u] + 1$
> > > > > ENQUEUE(Q, v)

Introduction to Algorithms

Unweighted graphs

Suppose that $w(u, v) = 1$ for all $(u, v) \in E$. Can Dijkstra's algorithm be improved?

- Use a simple FIFO queue instead of a priority queue.

Breadth-first search

> **while** $Q \neq \varnothing$
> > **do** $u \leftarrow \text{DEQUEUE}(Q)$
> > > **for** each $v \in Adj[u]$
> > > > **do if** $d[v] = \infty$
> > > > > **then** $d[v] \leftarrow d[u] + 1$
> > > > > $\text{ENQUEUE}(Q, v)$

Analysis: Time $= O(V + E)$.
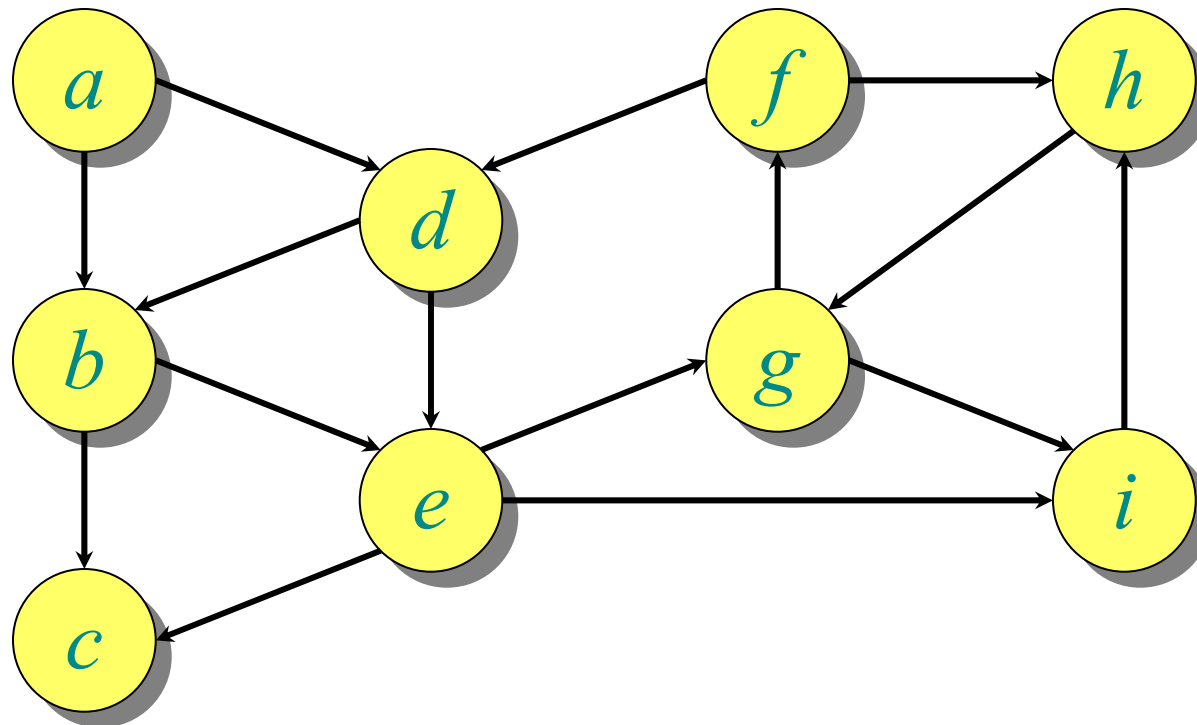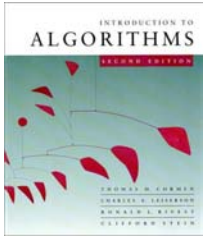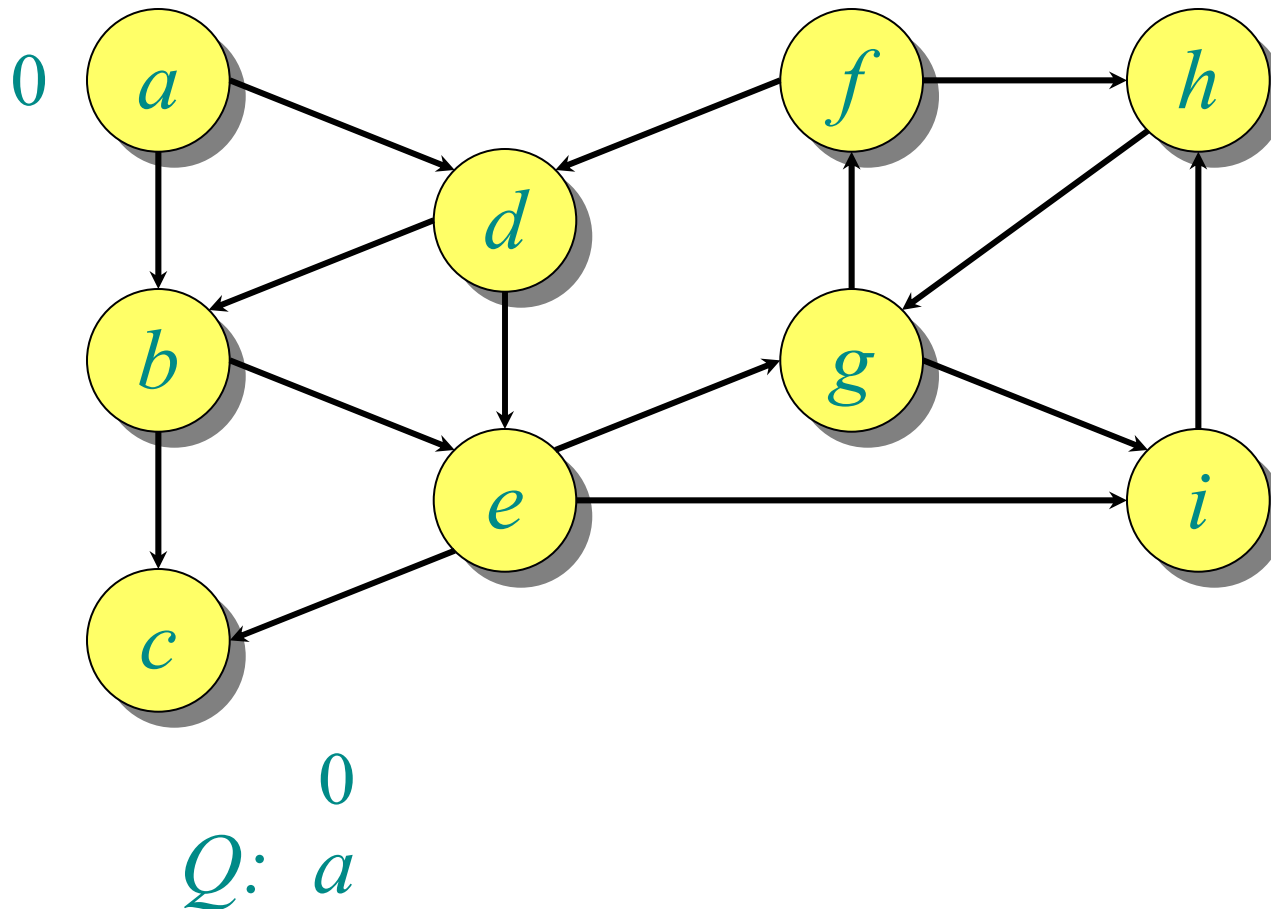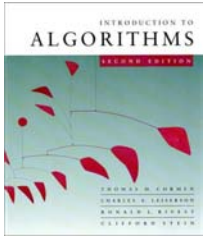
Introduction to Algorithms

Example of breadth-first search

Q:

Introduction to Algorithms November 1, 2004 L14.47

Example of breadth-first search

0

Q: a

Introduction to Algorithms

Example of breadth-first search

0   *a*

1   *d*

1   *b*

1   1

Q: a b d

Example of breadth-first search

Q: a b d c e

Introduction to Algorithms

Example of breadth-first search

0   *a*

1   *d*

f   *h*

1   *b*

g

e

i

2   *c*   2

2   2

Q: *a b d c e*

    Introduction to Algorithms   November 1, 2004   L14.51

Example of breadth-first search

Q: *a b d c e*

Introduction to Algorithms

Example of breadth-first search

0 a
1 d
1 b
2 c
1 (top of d)
f
h
3 g
2 e
i
3 (i)

$Q:$ a b d c e g i
3 3

Introduction to Algorithms

Example of breadth-first search

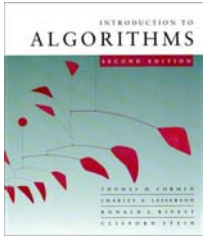Q: a b d c e g i f

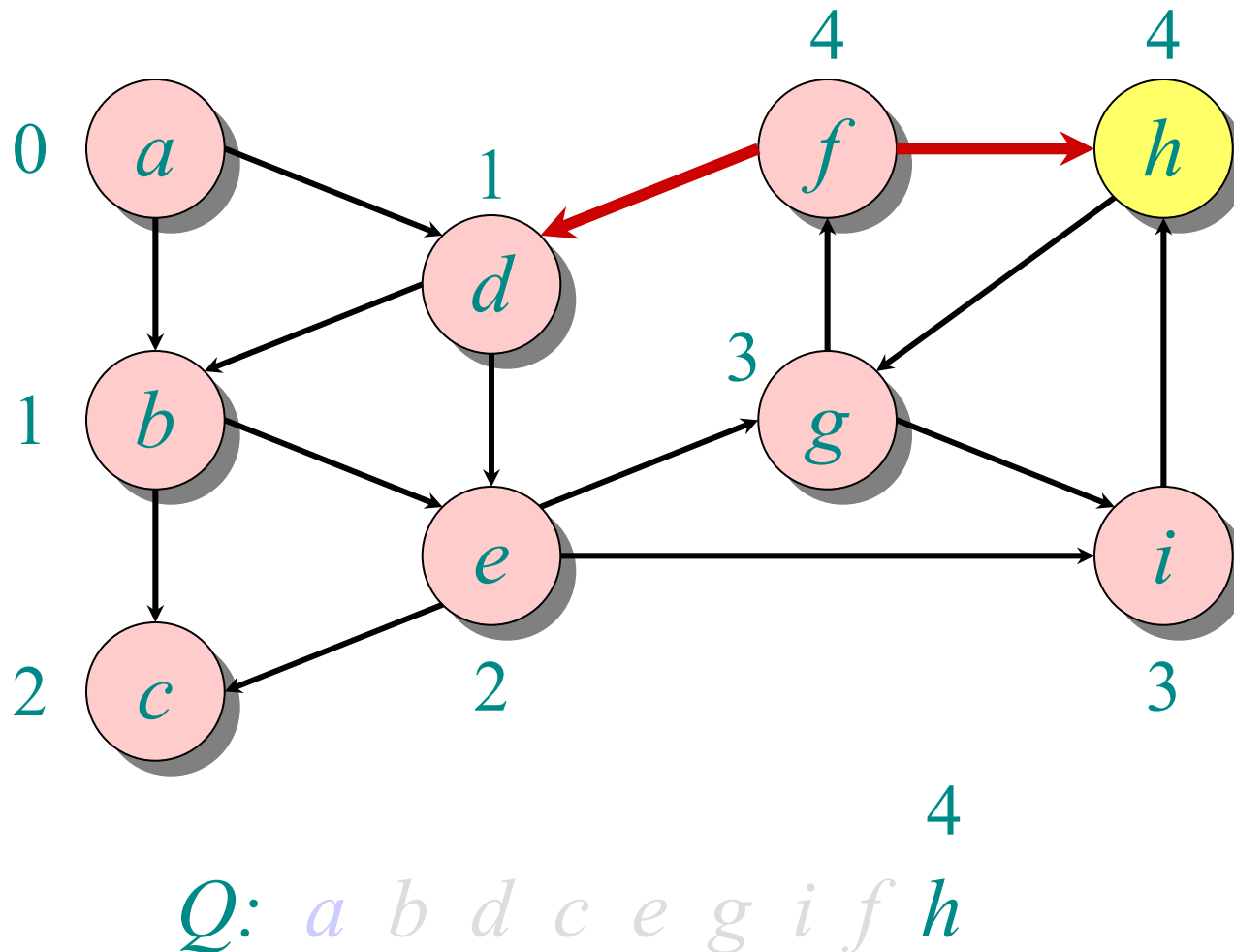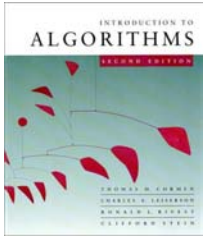*Introduction to Algorithms*

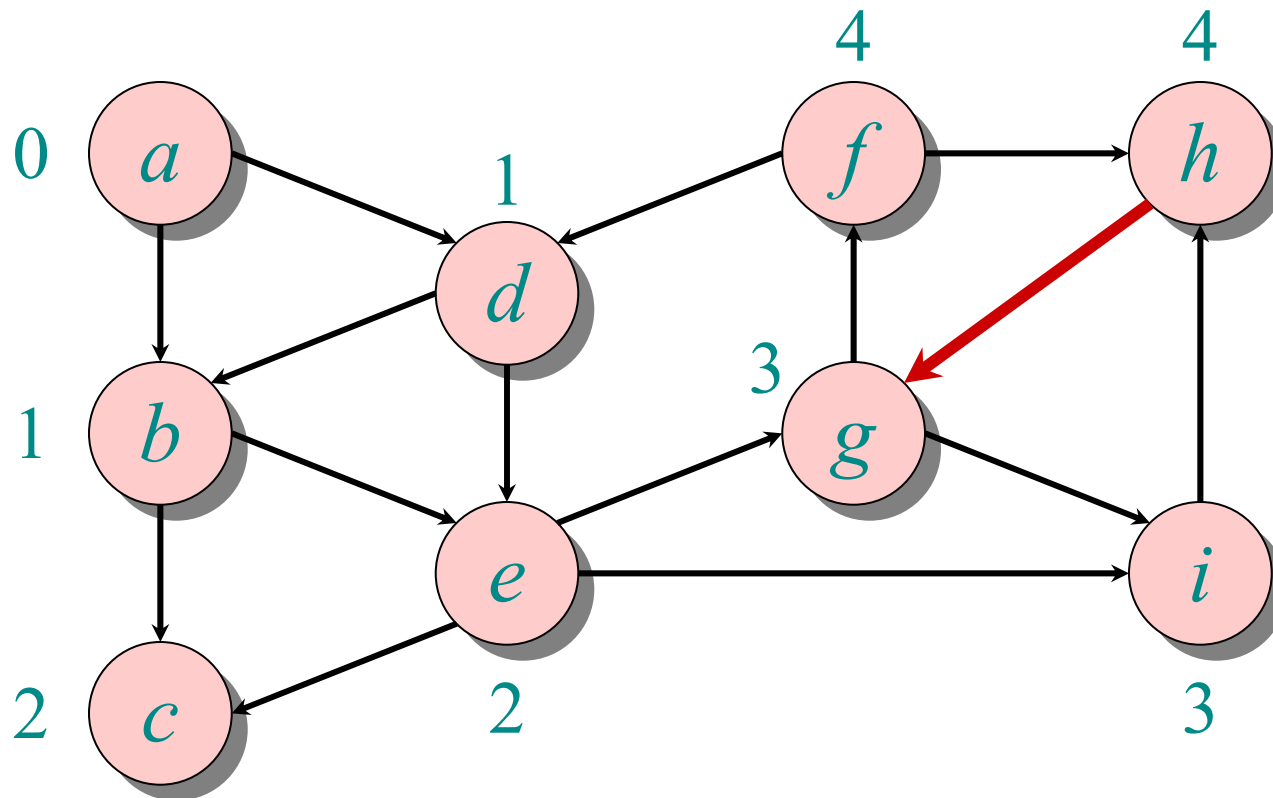# Example of breadth-first search

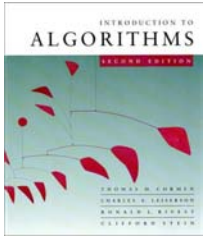# Example of breadth-first search

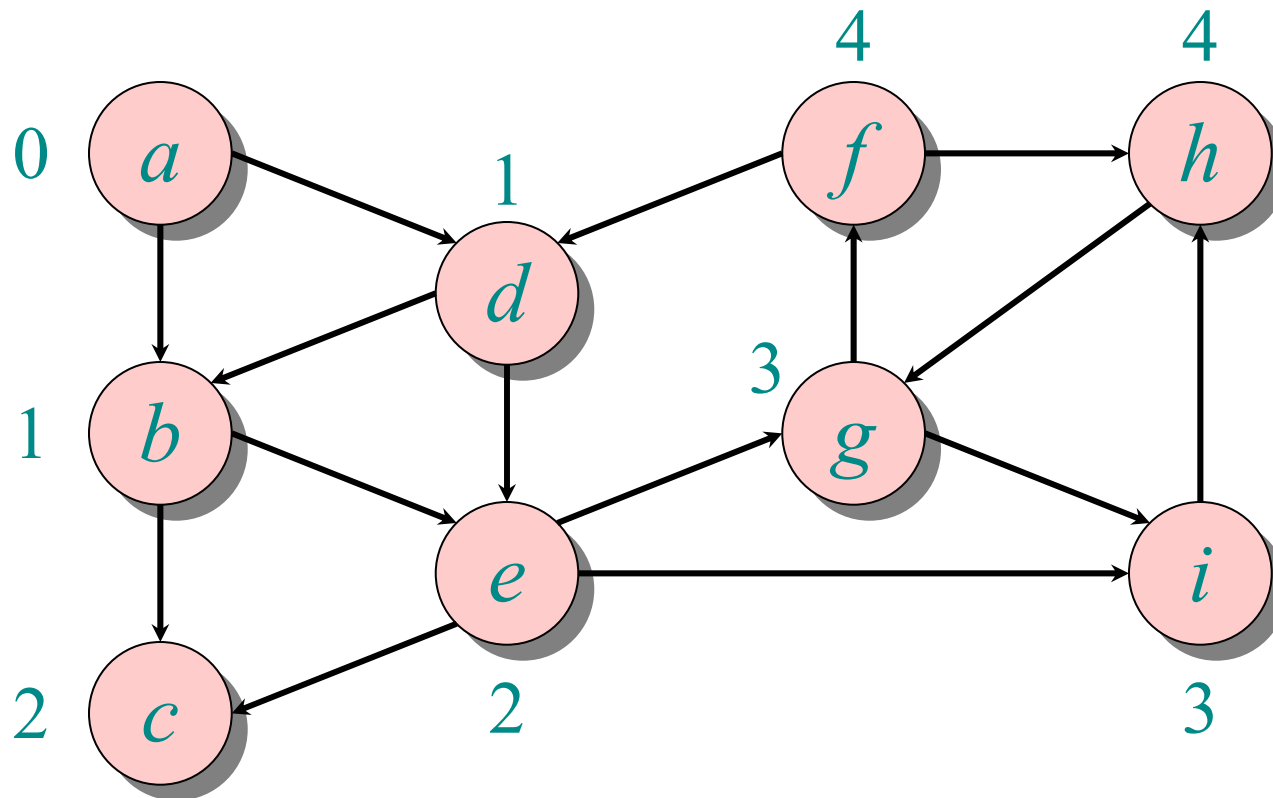# Example of breadth-first search

Q: a b d c e g i f h

Introduction to Algorithms

Example of breadth-first search

$Q: a\ b\ d\ c\ e\ g\ i\ f\ h$

Introduction to Algorithms
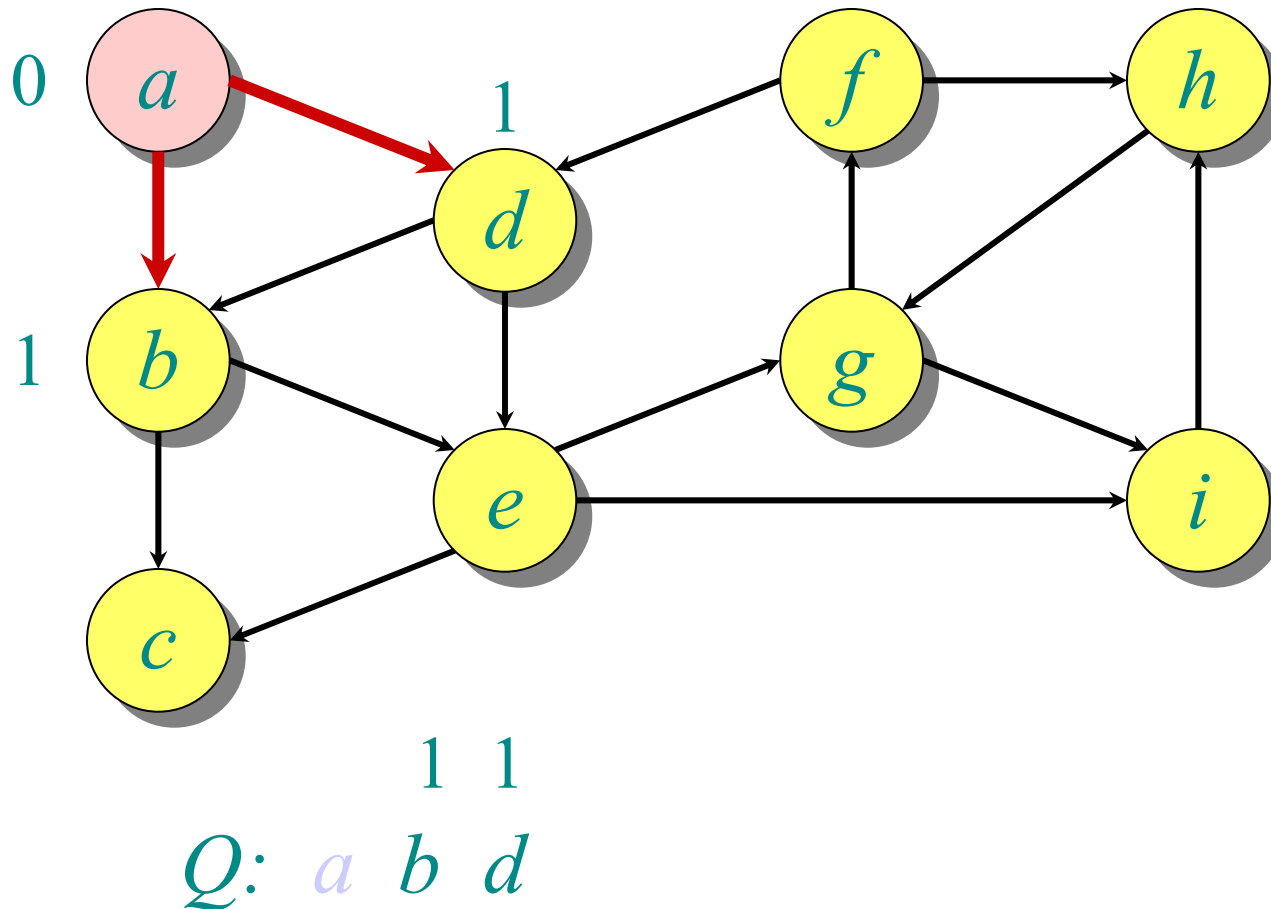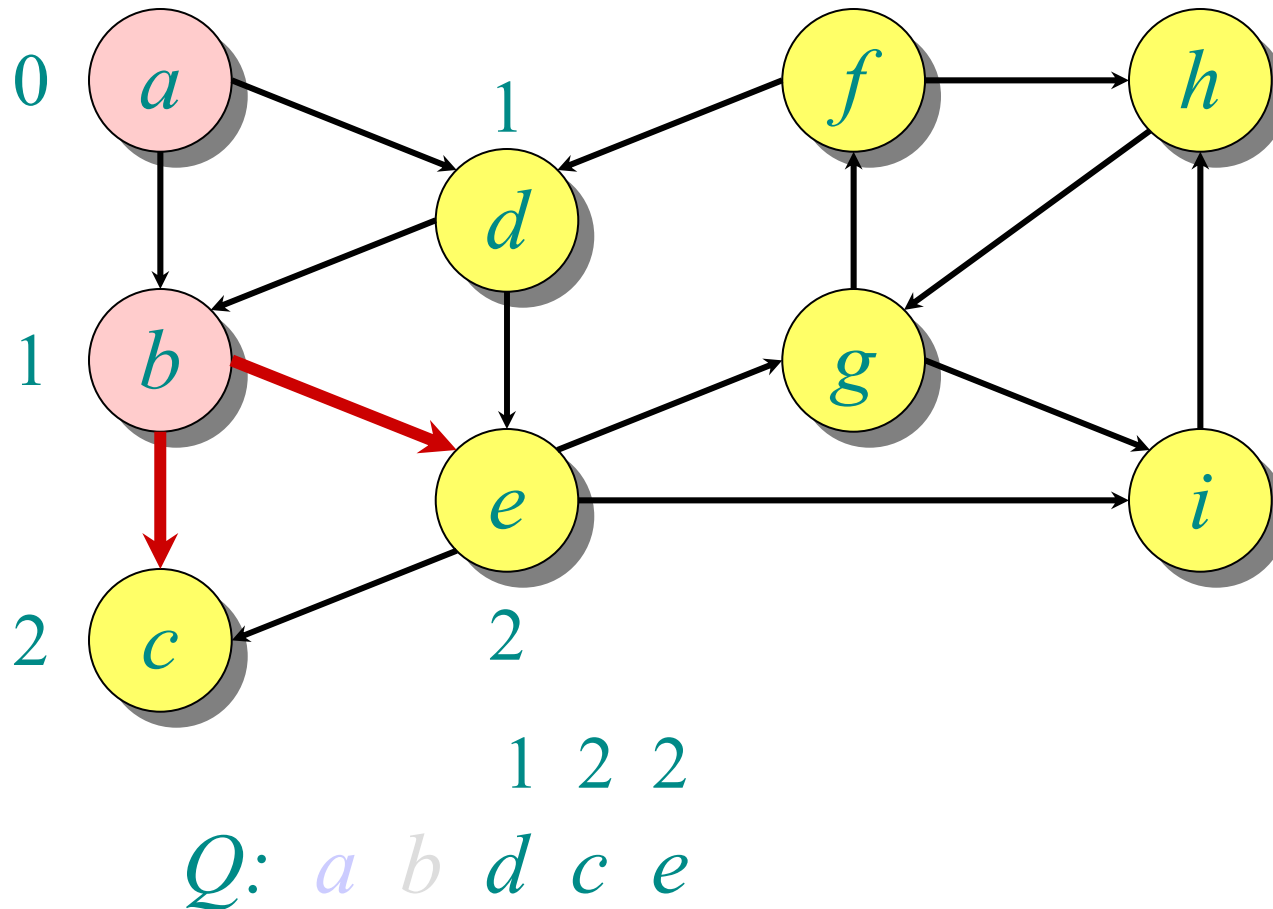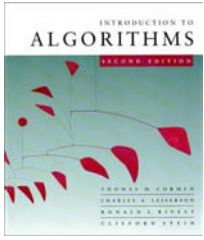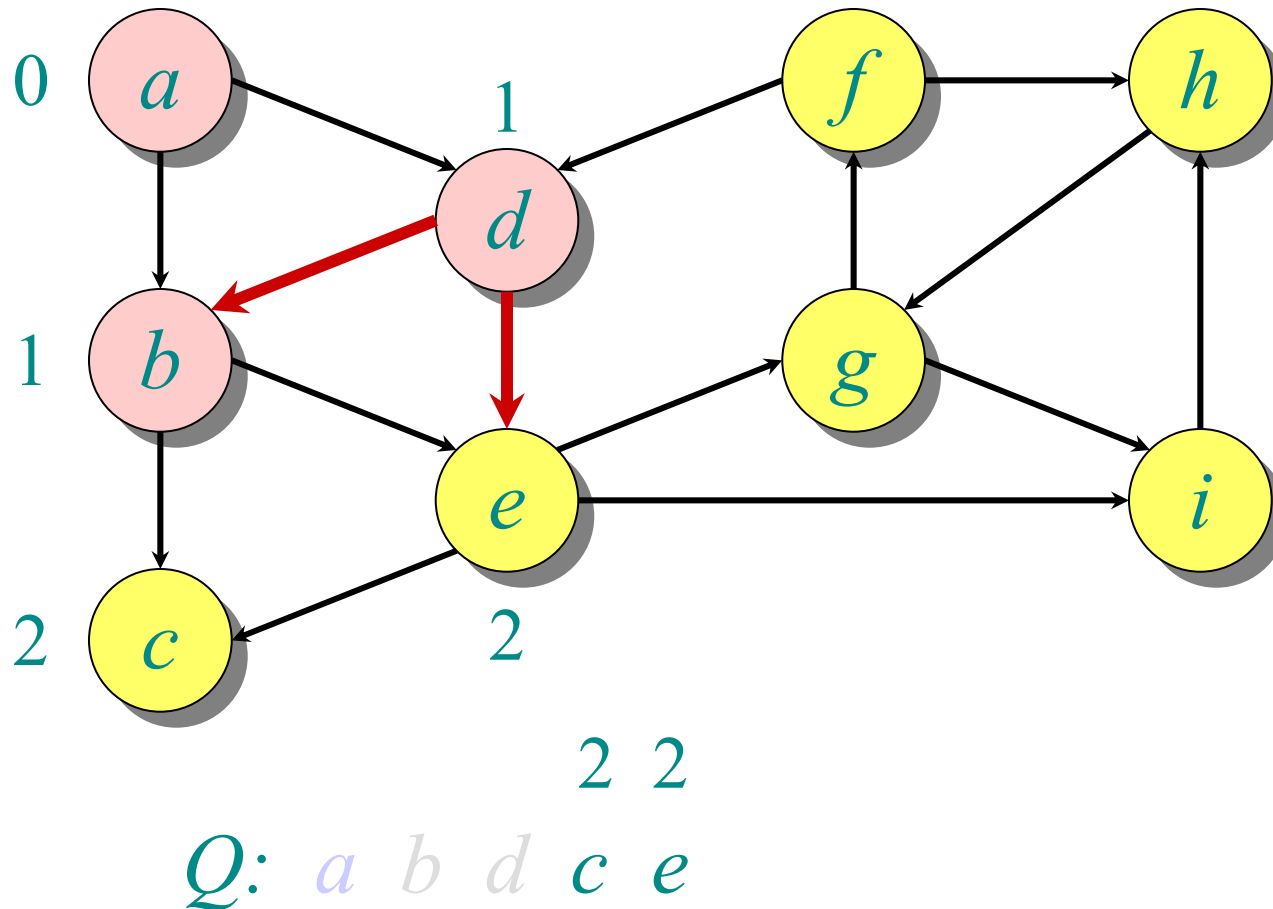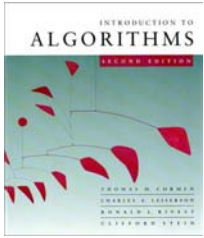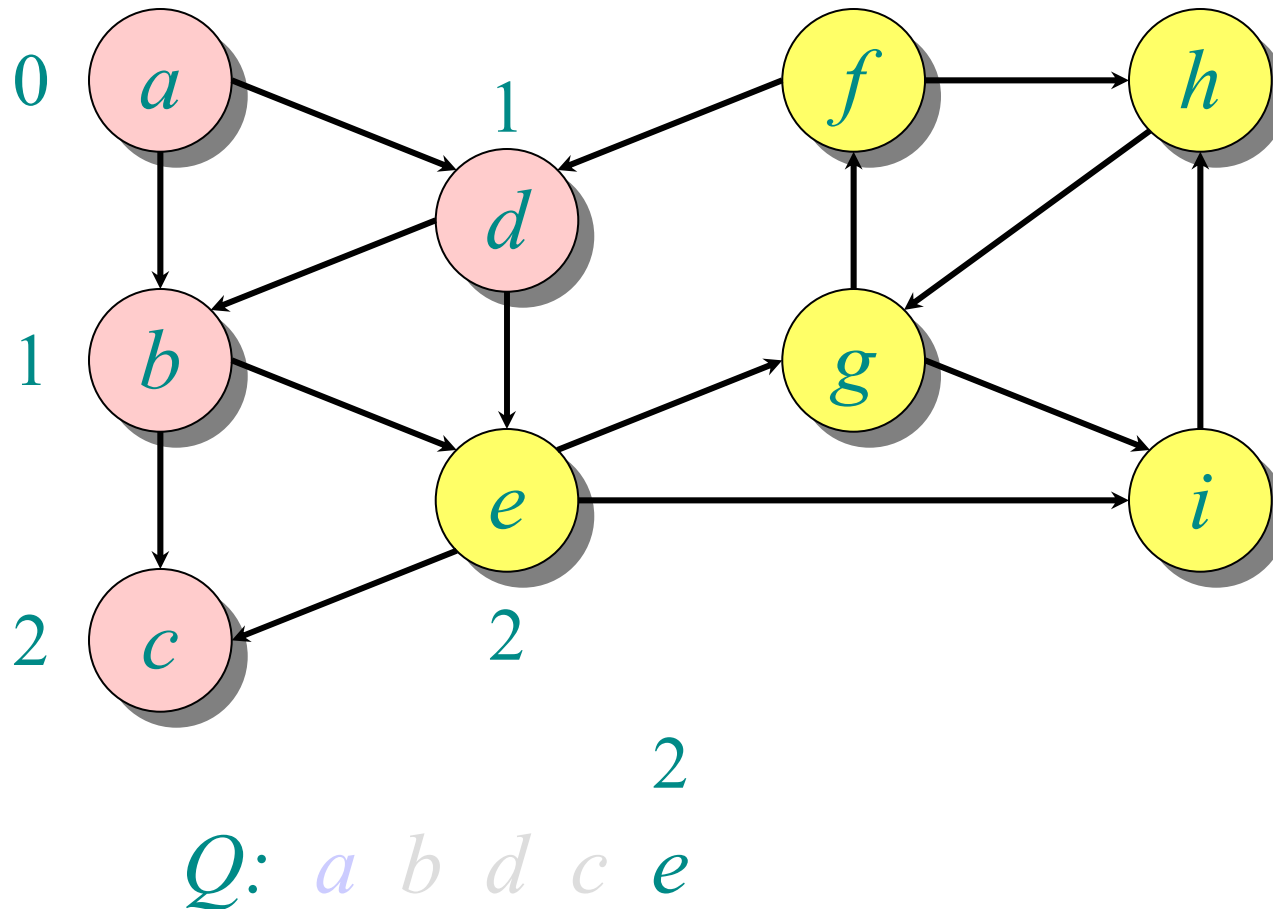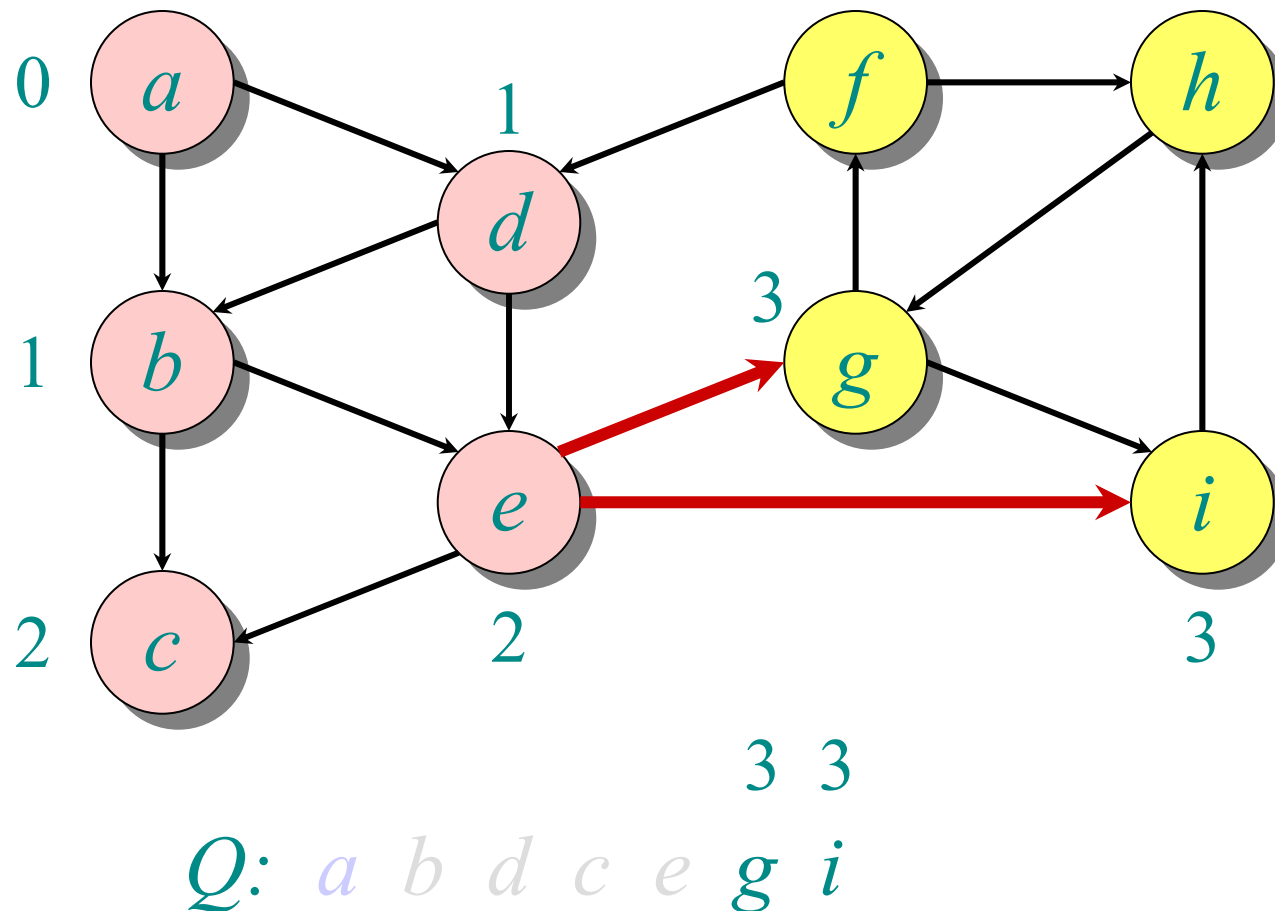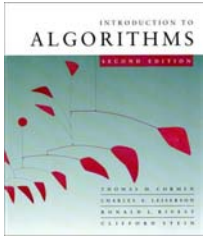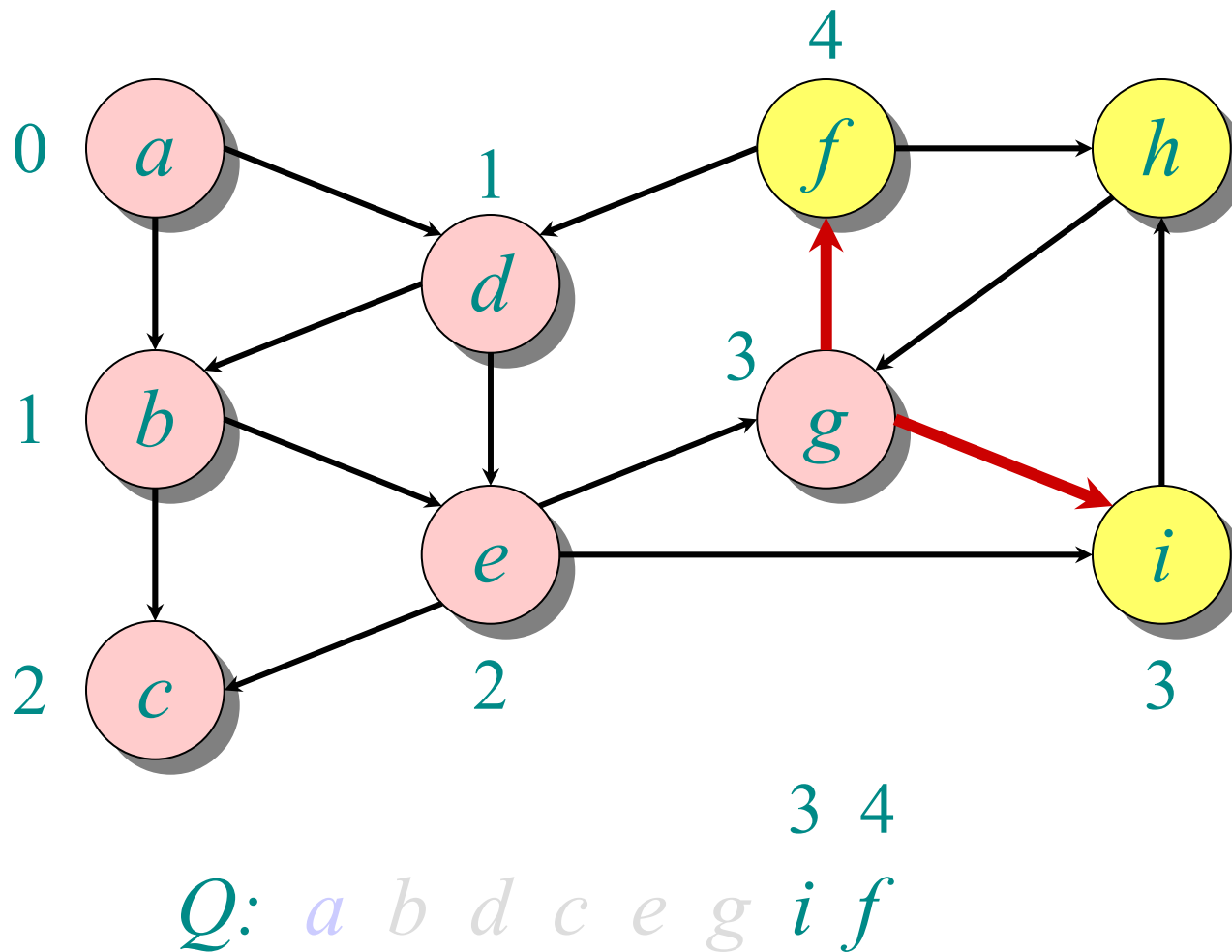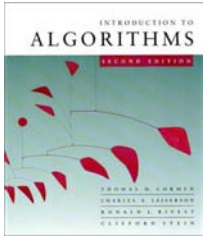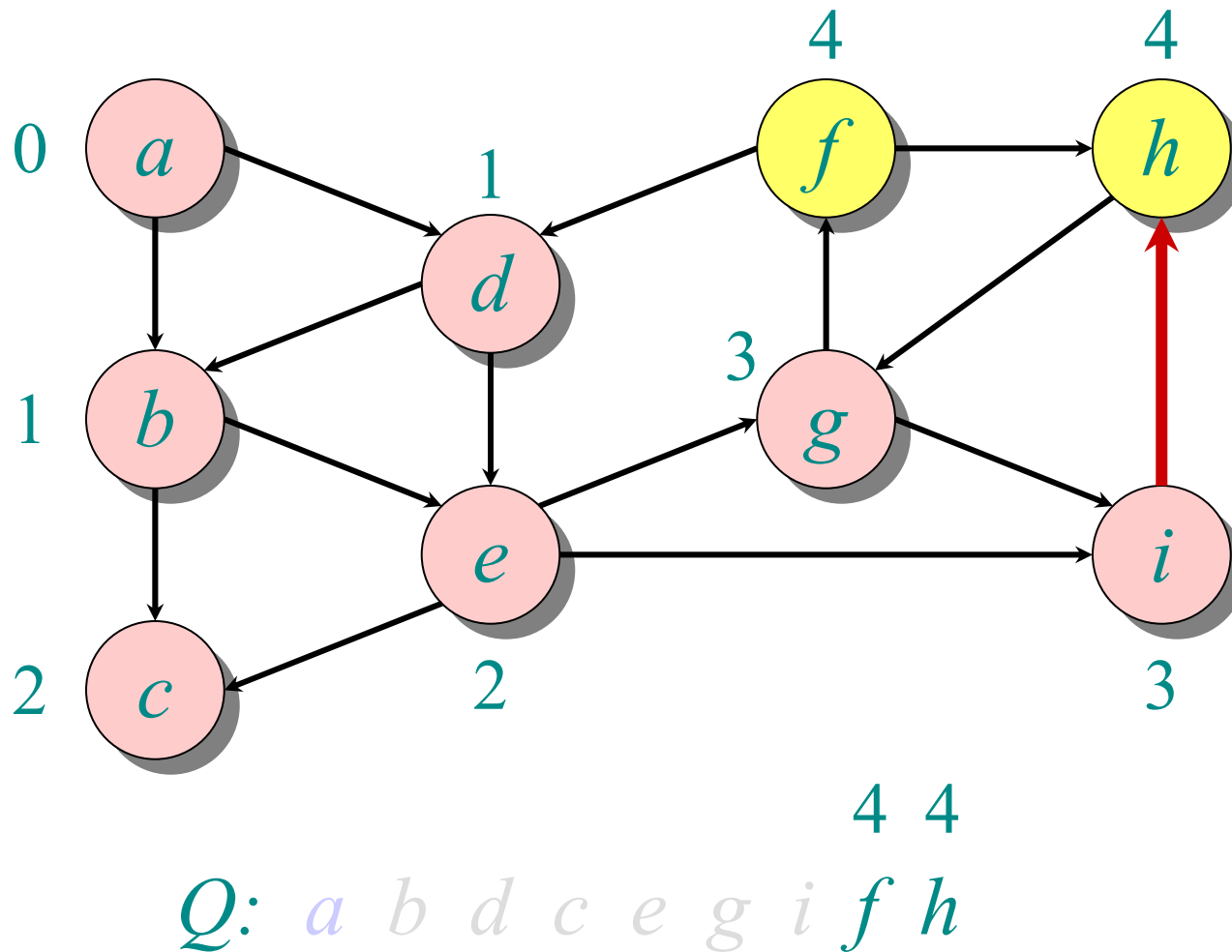
Correctness of BFS

$$\textbf{while } Q \neq \varnothing$$
$$\textbf{do } u \leftarrow \text{DEQUEUE}(Q)$$
$$\textbf{for } \text{each } v \in Adj[u]$$
$$\textbf{do if } d[v] = \infty$$
$$\textbf{then } d[v] \leftarrow d[u] + 1$$
$$\text{ENQUEUE}(Q, v)$$

Key idea:

The FIFO Q in breadth-first search mimics the priority queue Q in Dijkstra.

- **Invariant:** v comes after u in Q implies that $d[v] = d[u]$ or $d[v] = d[u] + 1$.

Introduction to Algorithms