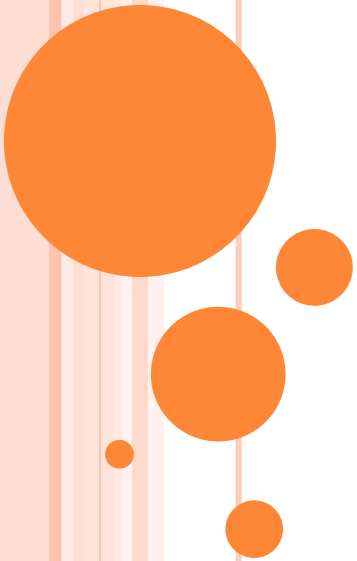
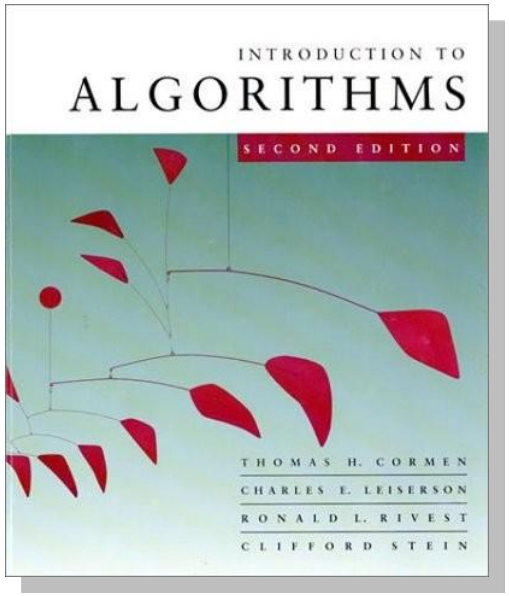


Graph Theory
Single Source Shortest Path

Design and Analysis of Algorithms

Fall 2022

INTRODUCTION TO ALGORITHMS
6.046J/18.401J



LECTURE 17

Shortest Paths I

- Properties of shortest paths
- Dijkstra's algorithm
- Correctness
- Analysis
- Breadth-first search

Prof. Erik Demaine

Paths in graphs

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

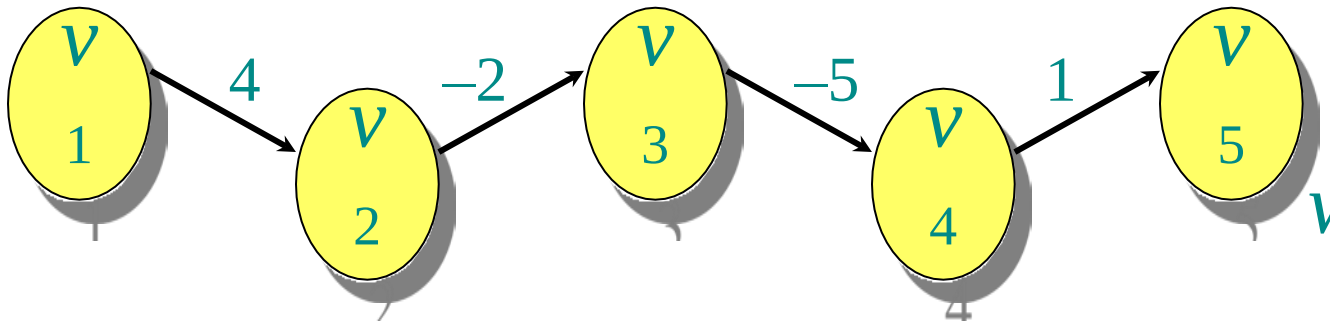
$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1}) .$$

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:



$w(p) = -2$

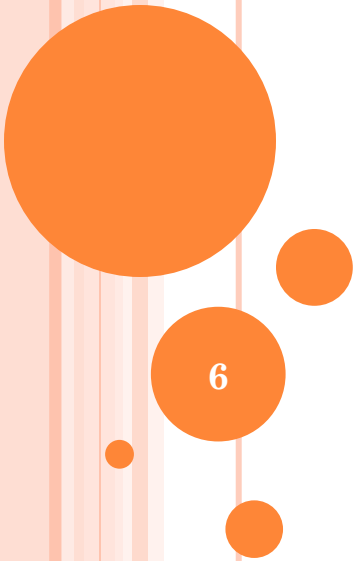
Shortest paths

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.

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

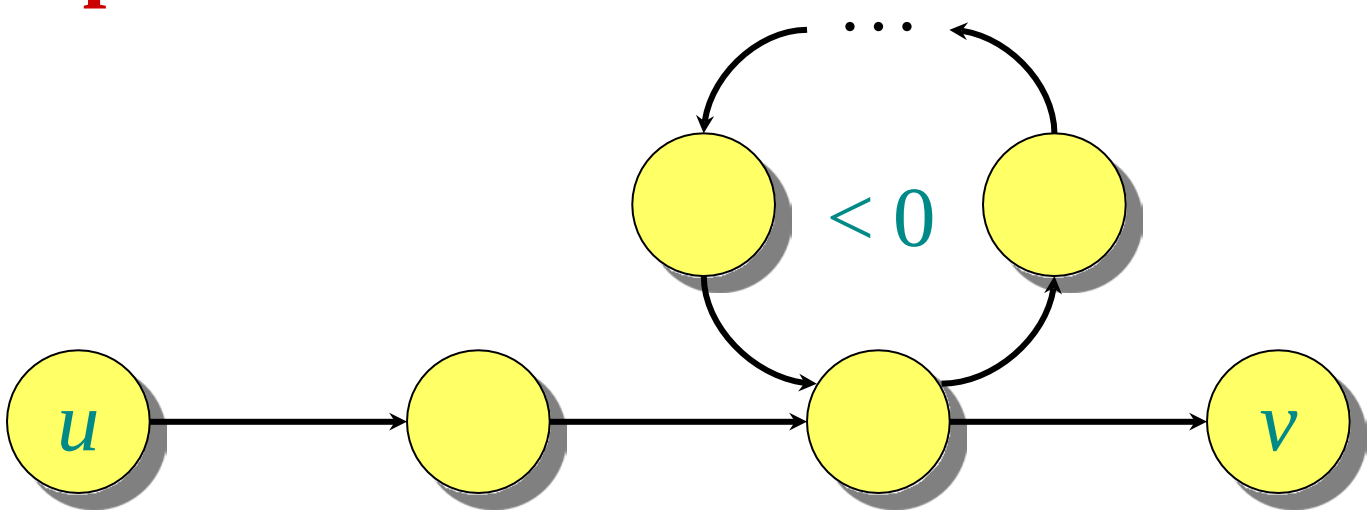


WELL-DEFINEDNESS OF SHORTEST PATHS

Well-definedness of shortest paths

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

Example:



Negative Weight-Real World Applications

The weight on edges can represent anything in the real world, for example

- Finance:
  - Arbitrage: Performing an arbitrage (the simultaneous buying and selling of securities, currency, or commodities in different markets or in derivative forms in order to take advantage of differing prices for the same asset between two or more markets
  - Money Transfer: the amount of money to be transferred from one account to another account. The amounts can be positive or negative. For example, if you want to go from a to b in your graph while losing as less money as possible (shortest path), then you can consider negative weights.
- Chemistry
  - The weights can be used to represent the heat produced during a chemical reaction.

NEGATIVE-WEIGHT CYCLE

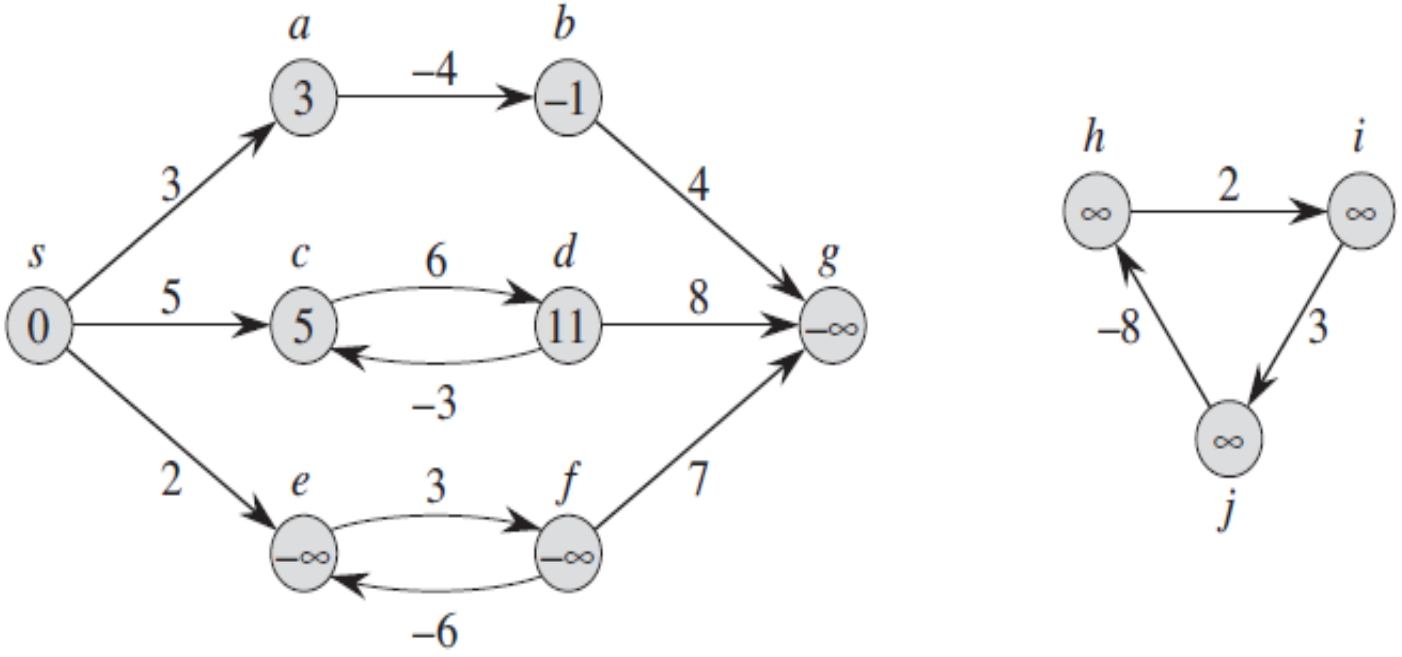
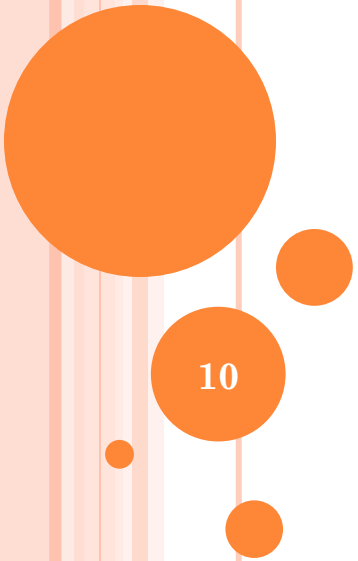


Figure 24.1 Negative edge weights in a directed graph. The shortest-path weight from source s appears within each vertex. Because vertices e and f form a negative-weight cycle reachable from s, they have shortest-path weights of $-\infty$. Because vertex g is reachable from a vertex whose shortest-path weight is $-\infty$, it, too, has a shortest-path weight of $-\infty$. Vertices such as h, i, and j are not reachable from s, and so their shortest-path weights are ∞, even though they lie on a negative-weight cycle.

Optimal substructure

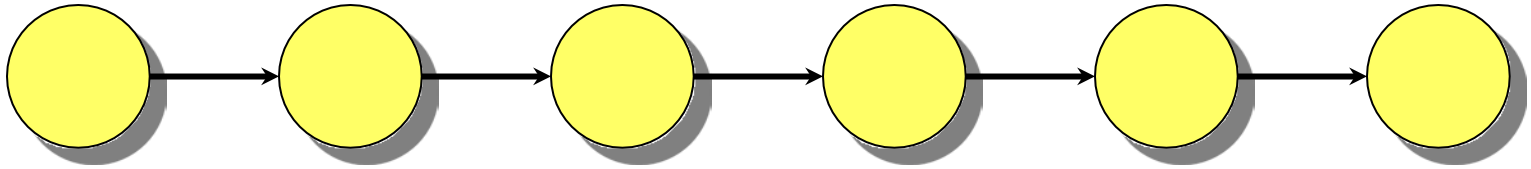
Theorem. A subpath of a shortest path is a shortest path.



Optimal substructure

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

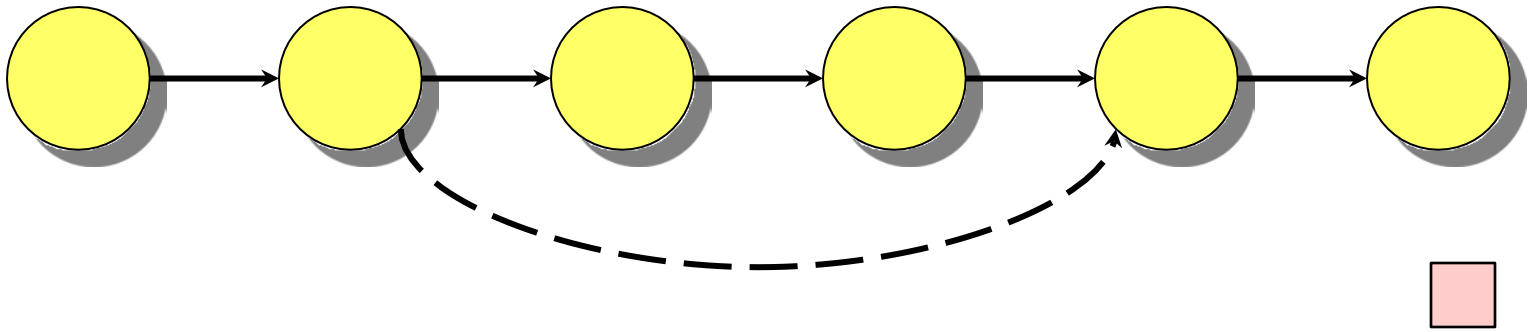
Proof. Cut and paste:



Optimal substructure

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

Proof. Cut and paste:



Triangle inequality

Theorem. For all $u, v, x \in V$, we have

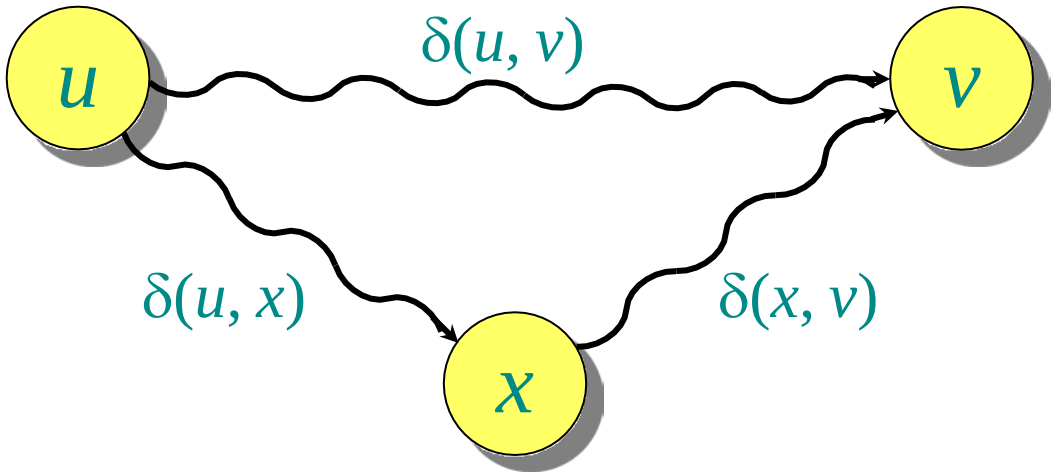
$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

Triangle inequality

Theorem. For all $u, v, x \in V$, we have

$$\delta(u, v) \leq \delta(u, x) + \delta(x, v).$$

Proof.



Single-source shortest paths (nonnegative edge weights)

Problem. Assume that $w(u, v) \geq 0$ for all $(u, v) \in E$. (Hence, all shortest-path weights must exist.) From a given source vertex $s \in V$, find the shortest-path weights $\delta(s, v)$ for all $v \in V$.

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 minimum.
3. Update the distance estimates of vertices adjacent to v.

Dijkstra's algorithm

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

DIJKSTRA'S ALGORITHM

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

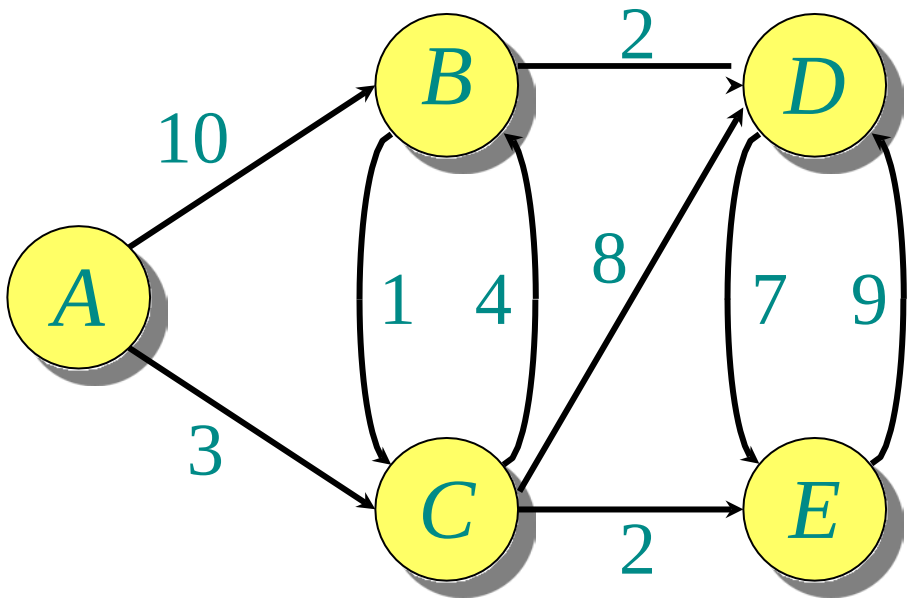
$d[s] \leftarrow 0$
for each $v \in V - \{s\}$
  do $d[v] \leftarrow \infty$
$S \leftarrow \emptyset$
$Q \leftarrow V$ ▷ Q is a priority queue maintaining $V - S$, keyed on $d[v]$
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    $S \leftarrow S \cup \{u\}$
    for each $v \in Adj[u]$
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

relaxation step

Implicit DECREASE-KEY

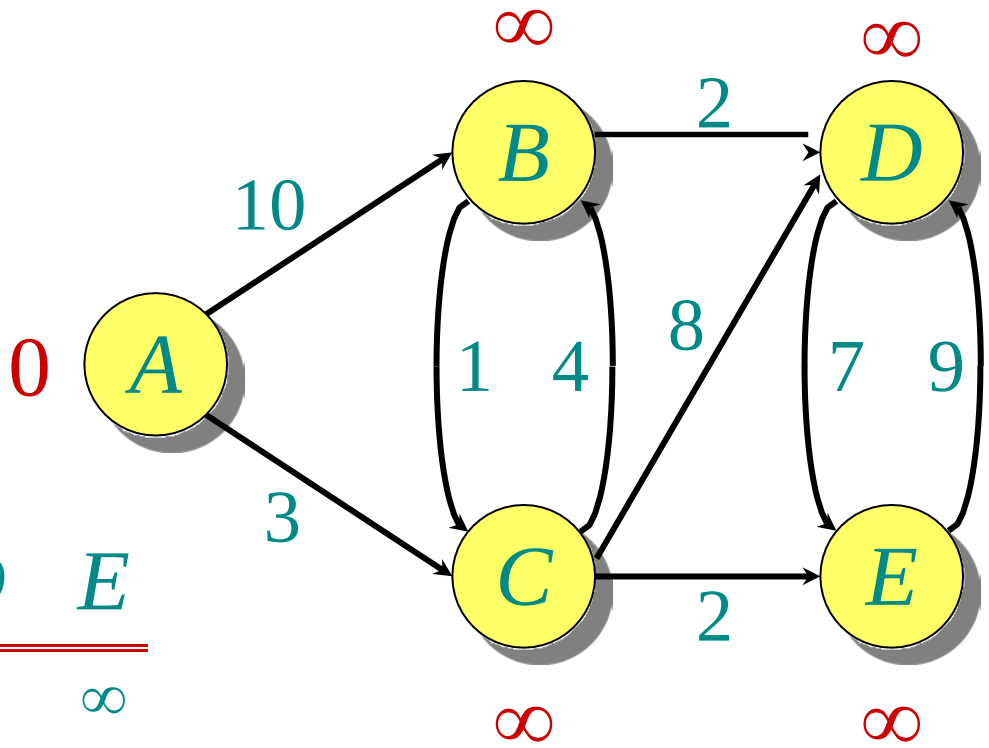
Example of Dijkstra's algorithm

Graph with nonnegative edge weights:



Example of Dijkstra's algorithm

Initialize:

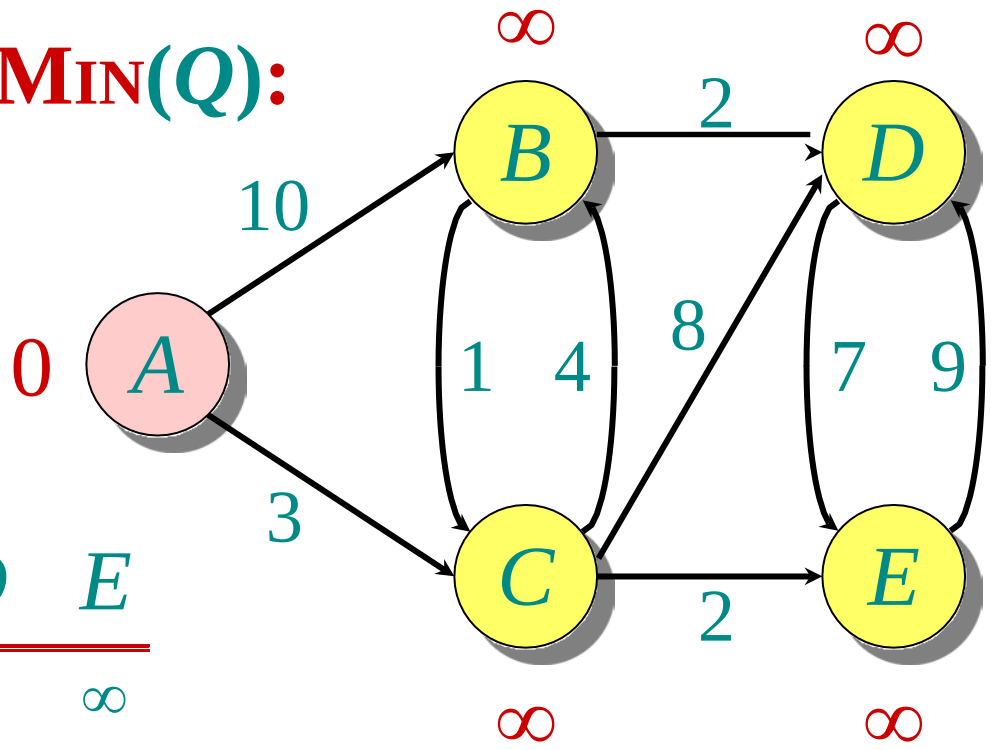


Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}

EXAMPLE OF DIJKSTRA'S ALGORITHM

"*A*" ← EXTRACT-MIN(*Q*):



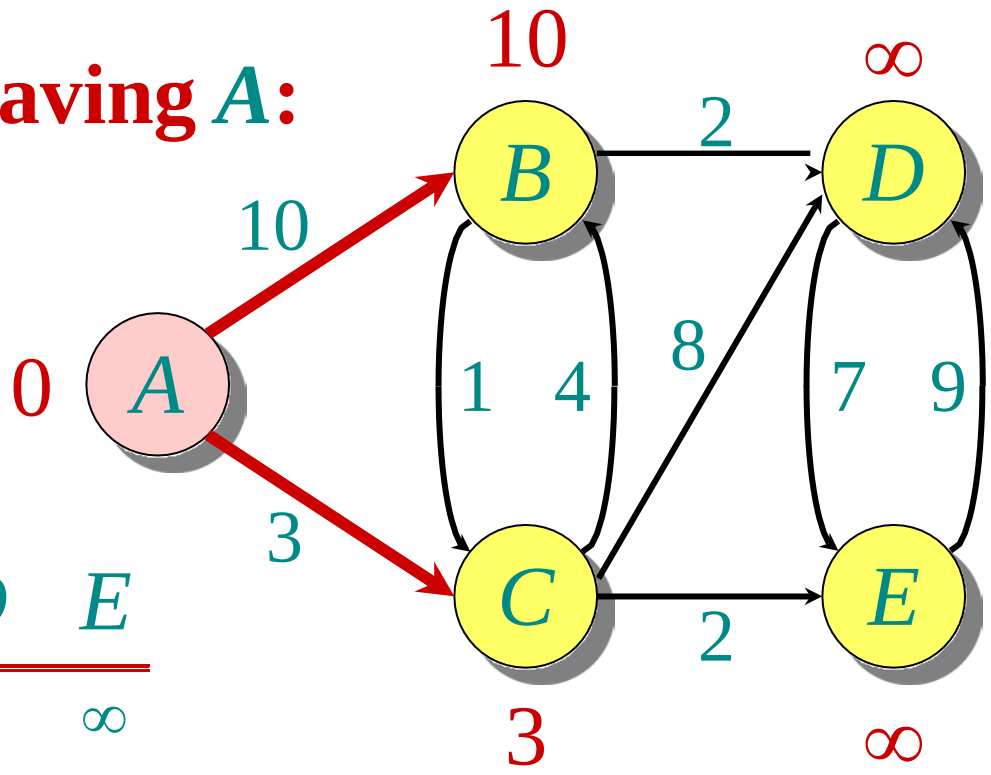
Q: *A* *B* *C* *D* *E*

A	*B*	*C*	*D*	*E*
0	∞	∞	∞	∞

S: { *A* }

EXAMPLE OF DIJKSTRA'S ALGORITHM

Relax all edges leaving *A*:

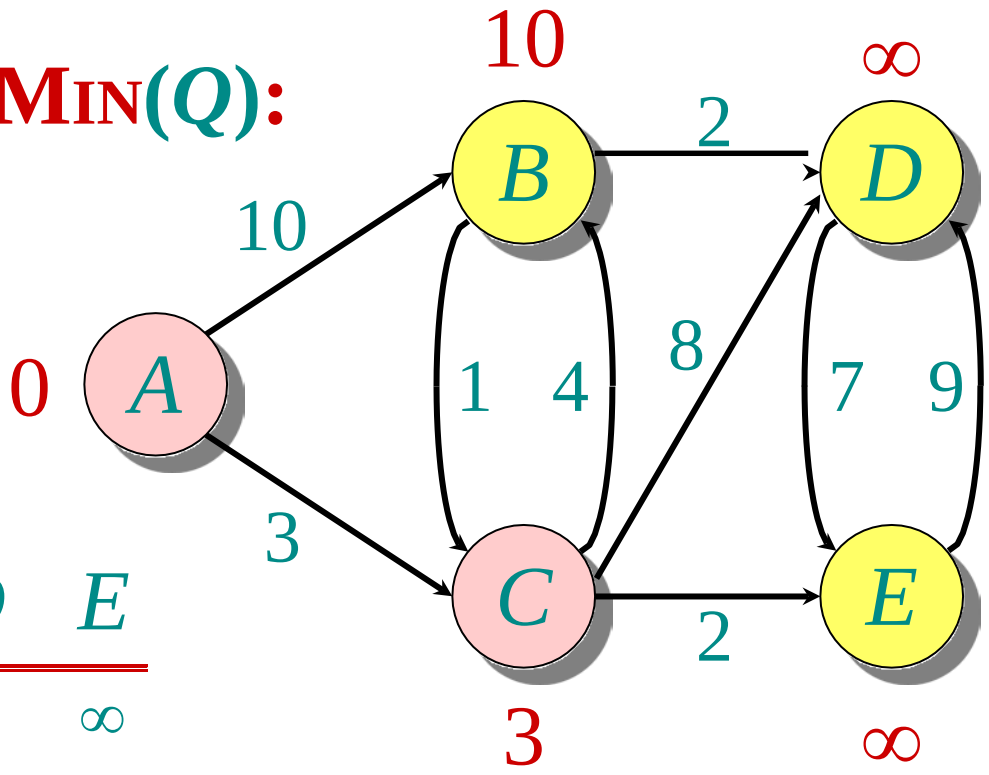


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { *A* }

EXAMPLE OF DIJKSTRA'S ALGORITHM

"C" ← EXTRACT-MIN(*Q*):

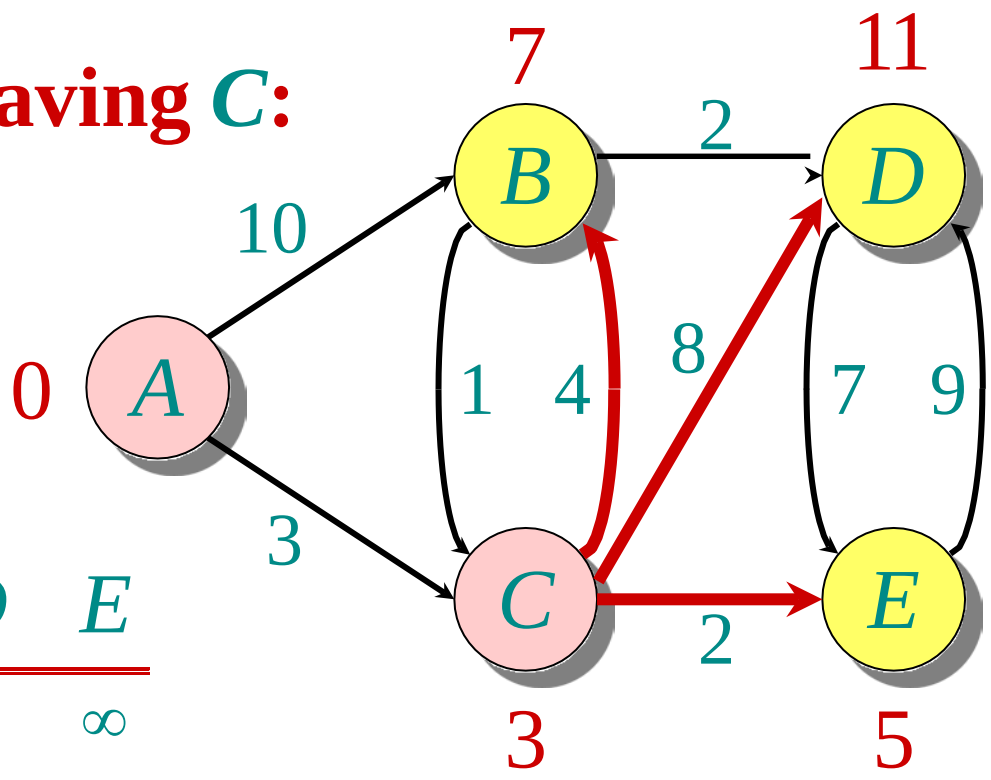


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { *A*, *C* }

Example of Dijkstra's algorithm

Relax all edges leaving *C*:

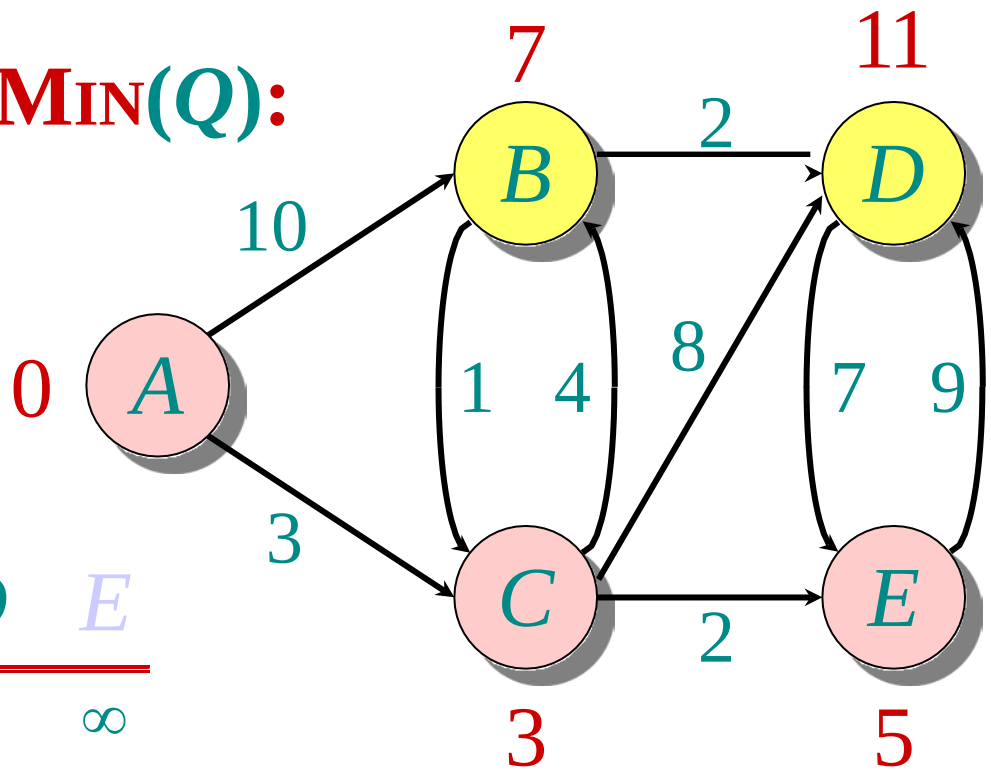


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { *A*, *C* }

Example of Dijkstra's algorithm

"*E*" ← EXTRACT-MIN(*Q*):

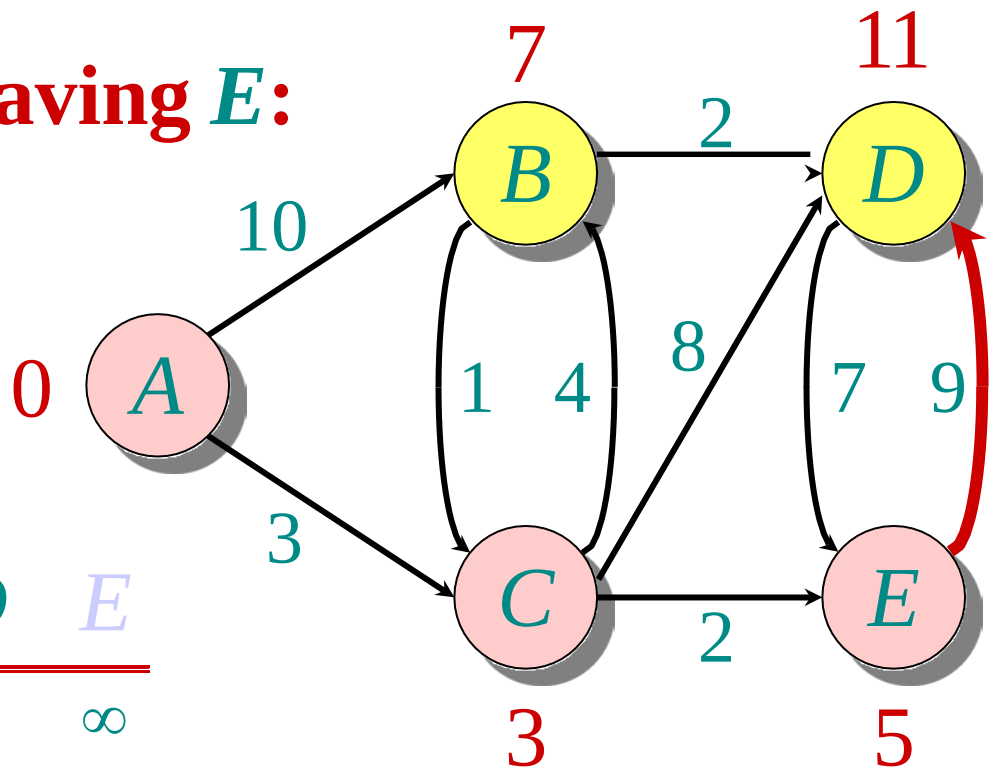


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

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

Example of Dijkstra's algorithm

Relax all edges leaving *E*:

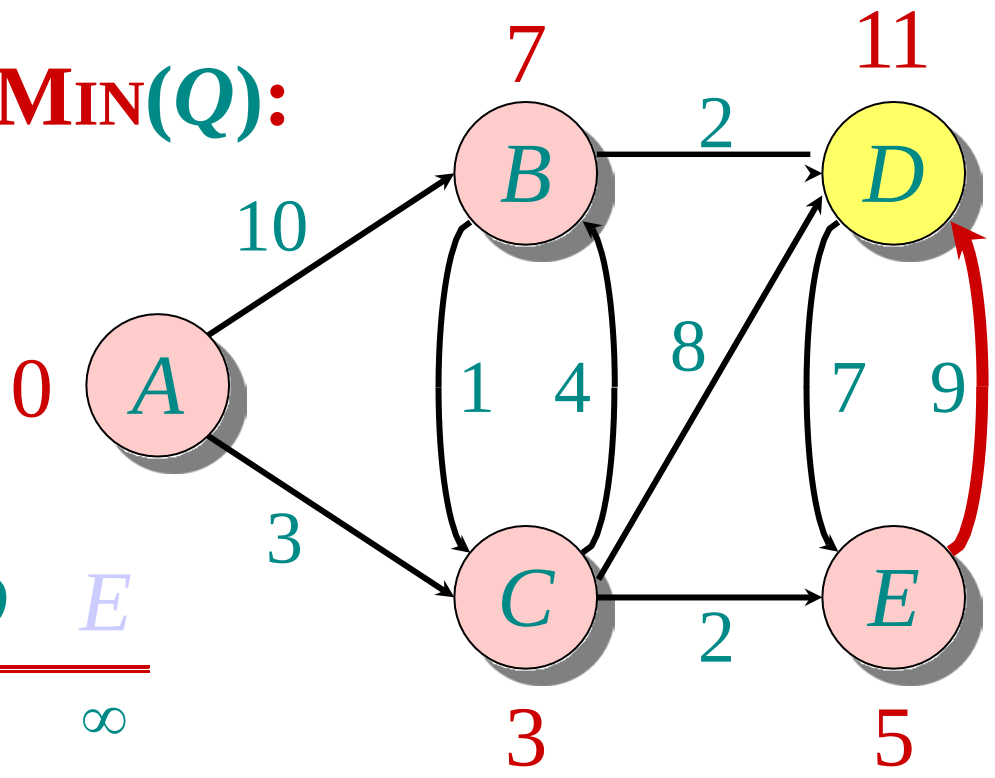


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

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

EXAMPLE OF DIJKSTRA'S ALGORITHM

"B" ← EXTRACT-MIN(*Q*):

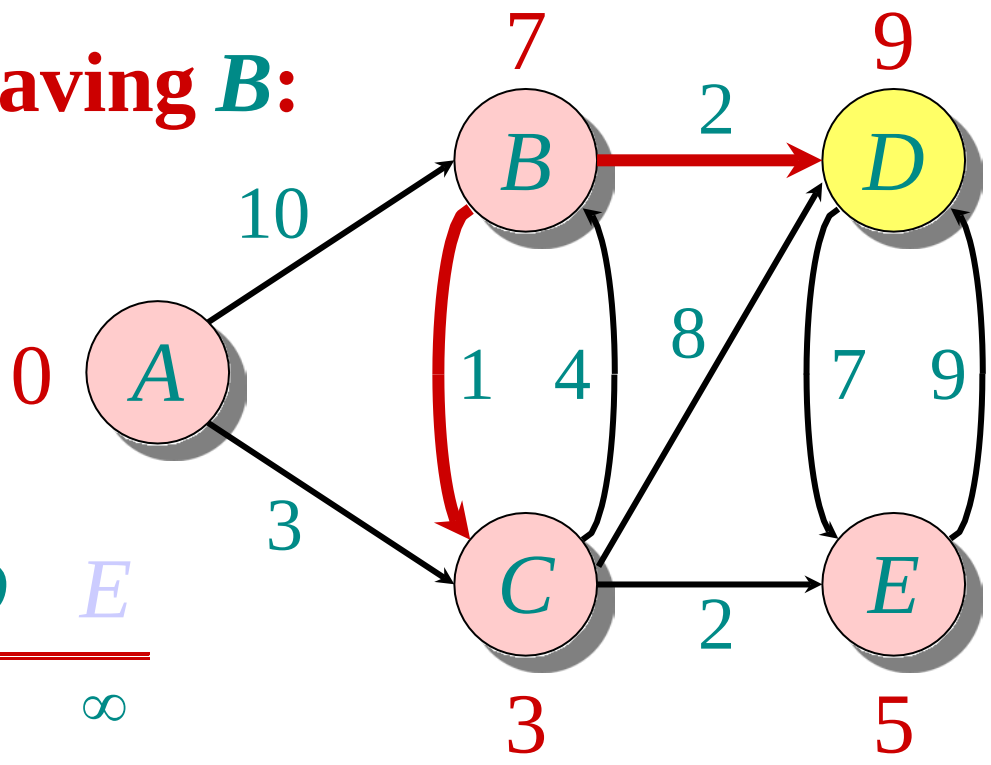


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

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

Example of Dijkstra's algorithm

Relax all edges leaving *B*:

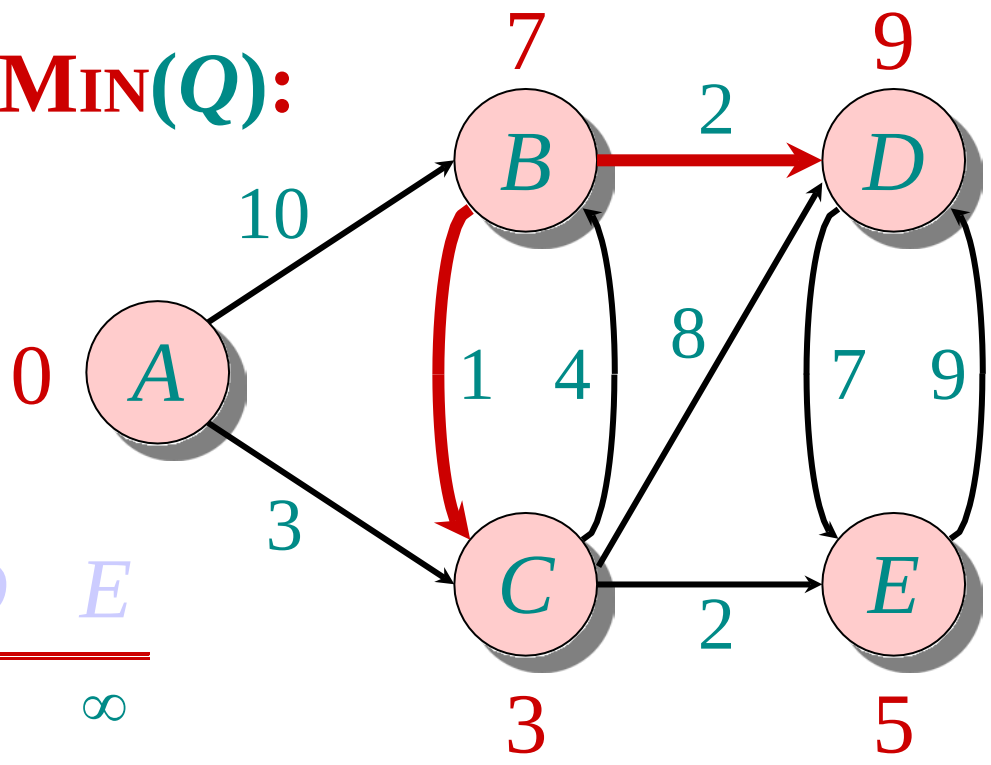


Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

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

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* }

Analysis of Dijkstra

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

Analysis of Dijkstra

$|V|$ times

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

Analysis of Dijkstra

$|V|$ times

while $Q \neq \varnothing$

do $u \leftarrow$ EXTRACT-MIN(Q)

$S \leftarrow S \cup \{u\}$

$degree(u)$ times

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:

while $Q \neq \varnothing$

do $u \leftarrow$ EXTRACT-MIN(Q)

$S \leftarrow S \cup \{u\}$

$degree(u)$ times:

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.

Analysis of Dijkstra

$|V|$ times

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

$degree(u)$ times

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.

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

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	$O(V)$	$O(1)$	$O(V^2)$

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	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

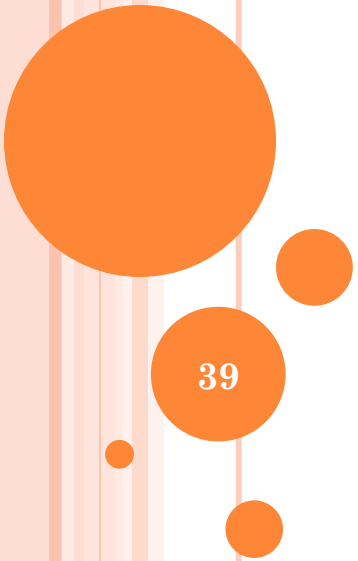
Analysis of Dijkstra (continued)

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

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

Unweighted graphs

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



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.

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 ≠ ∅
    do u ← DEQUEUE(Q)
       for each v ∈ Adj[u]
           do if d[v] = ∞
                  then d[v] ← d[u] + 1
                       ENQUEUE(Q, v)
```

UNWEIGHTED GRAPHS

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

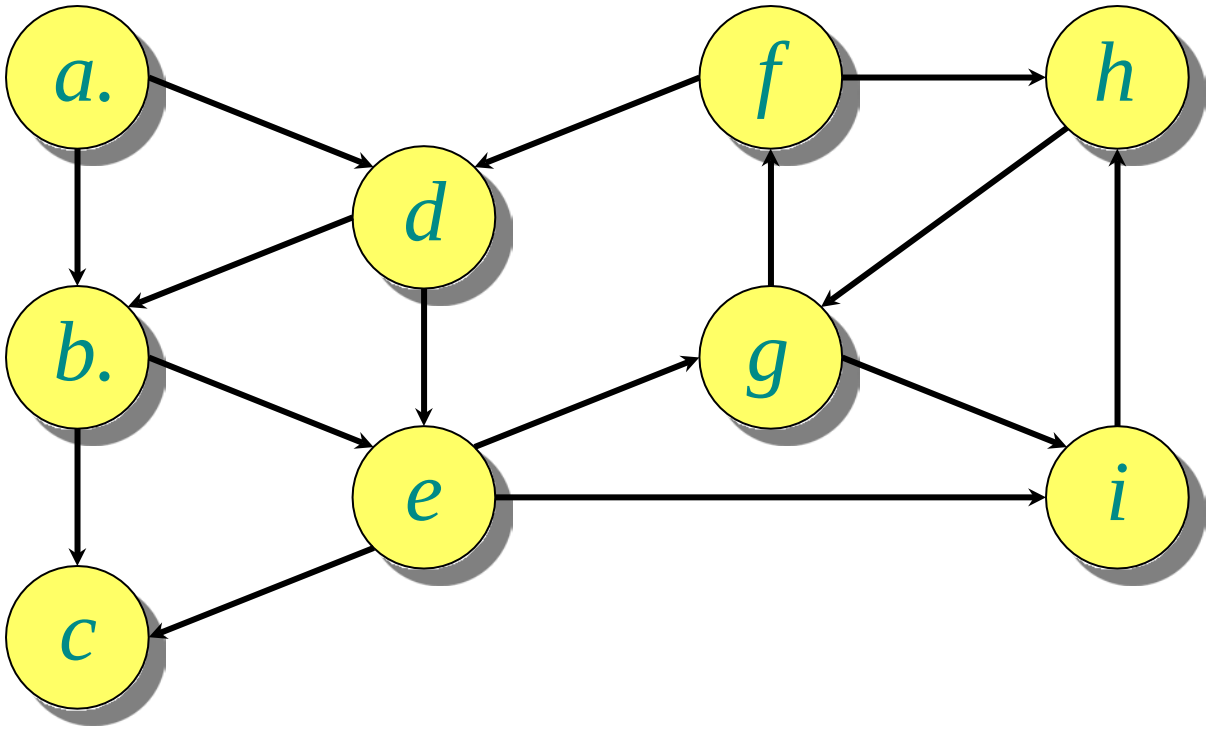
- Use a simple FIFO queue instead of a priority queue.

Breadth-first search

```
while Q ≠ ∅
    do u ← DEQUEUE(Q)
       for each v ∈ Adj[u]
           do if d[v] = ∞
                  then d[v] ← d[u] + 1
                       ENQUEUE(Q, v)
```

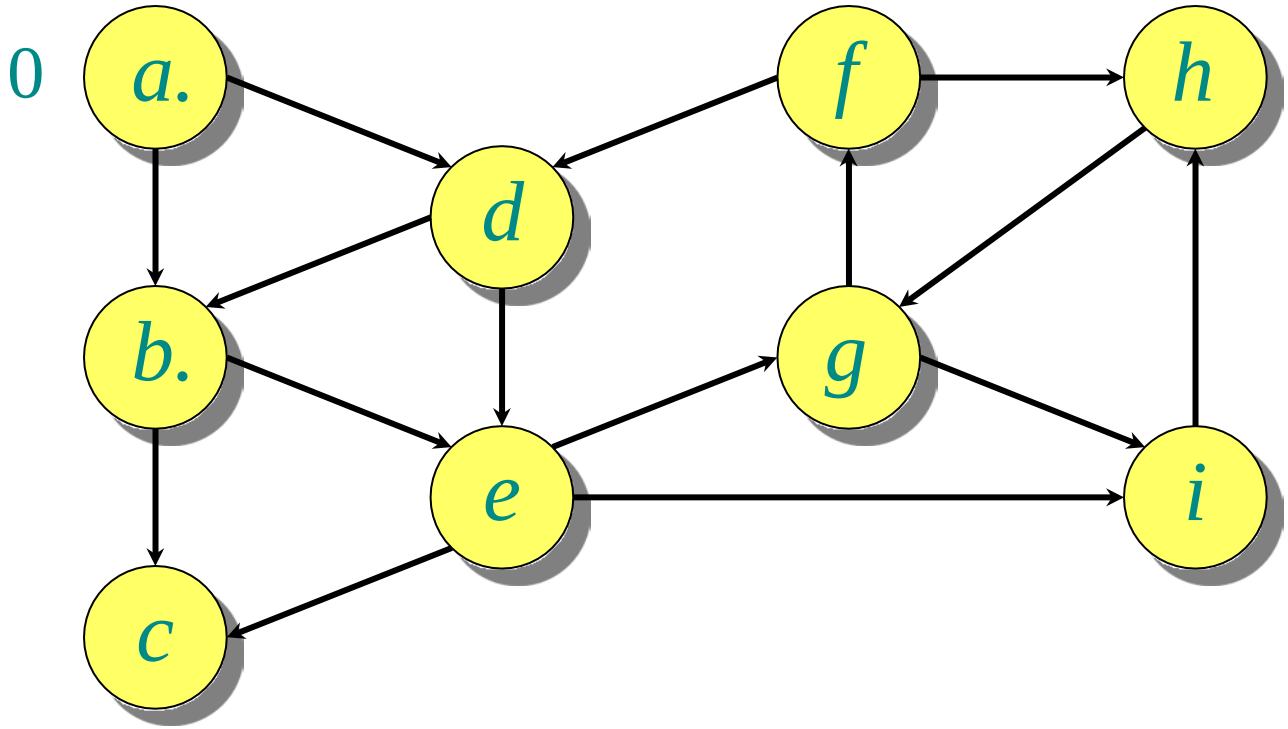
Analysis: Time = $O(V + E)$.

Example of breadth-first search



Q:

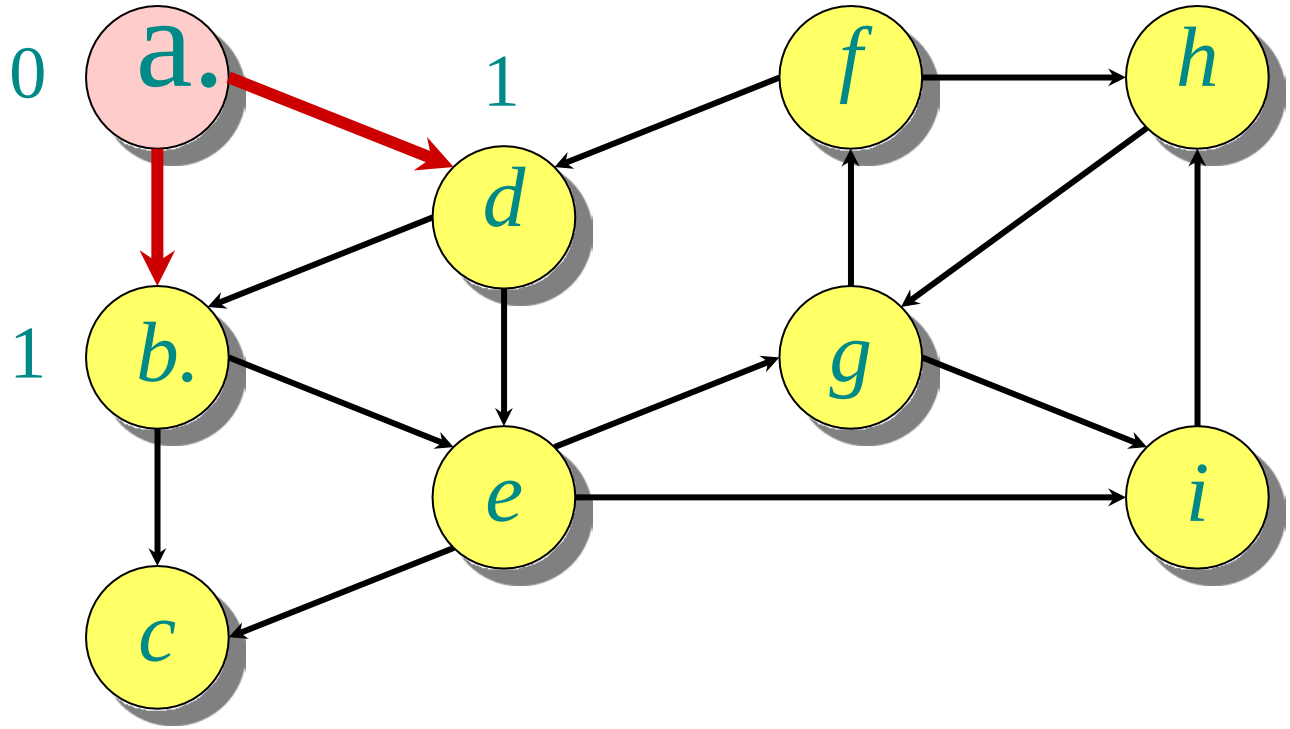
Example of breadth-first search



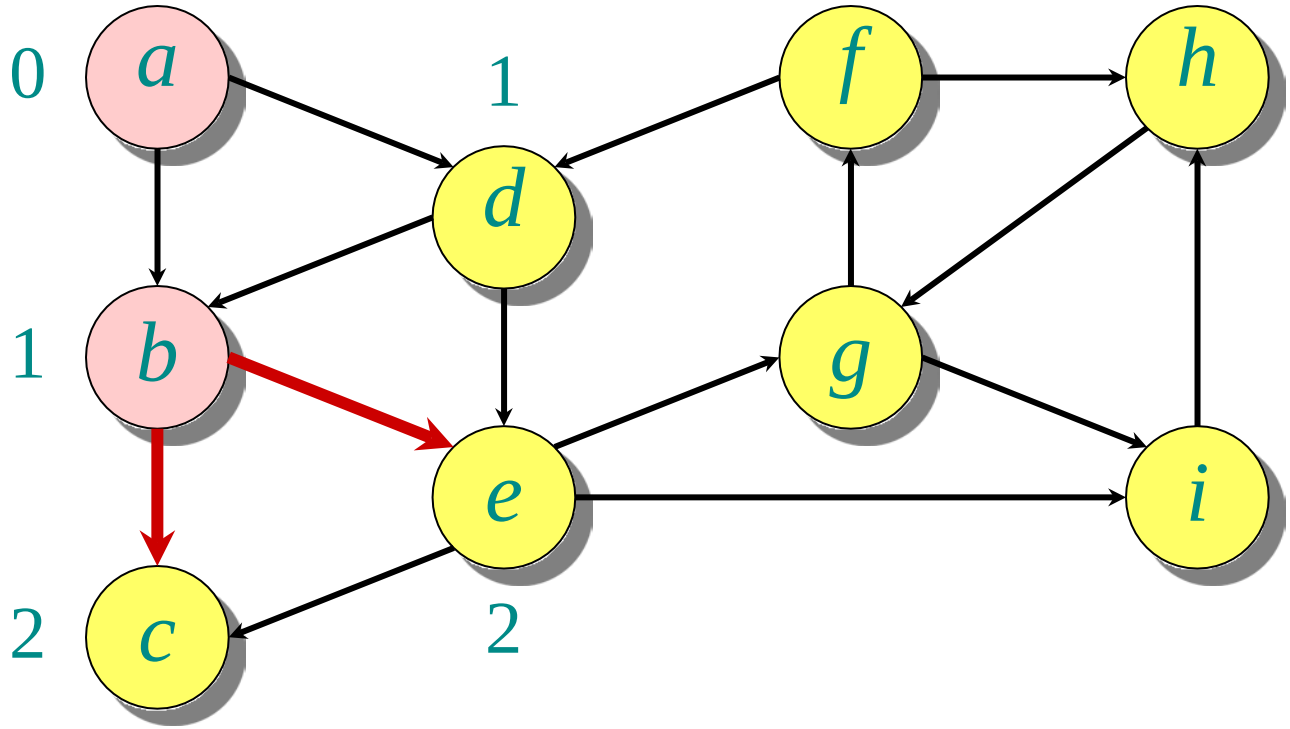
0

Q: *a*

Example of breadth-first search



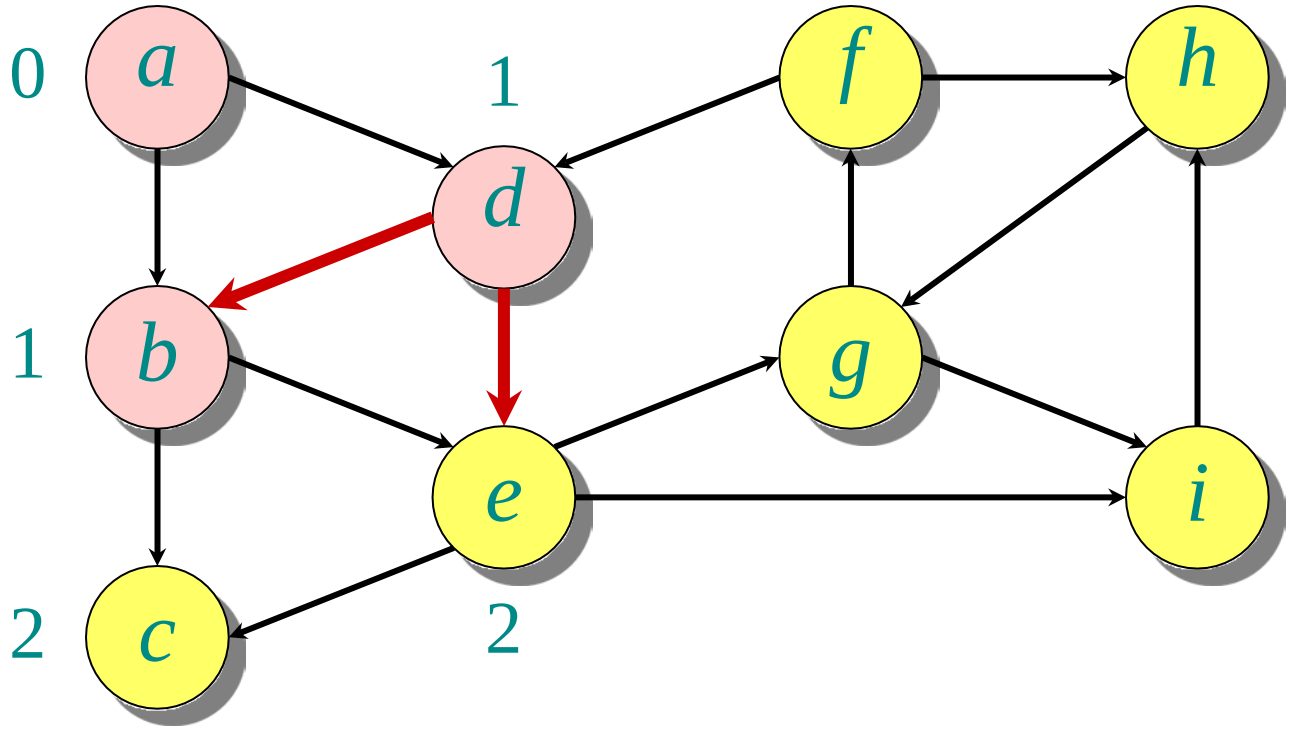
Example of breadth-first search



1 2 2

Q: *a* *b* *d* *c* *e*

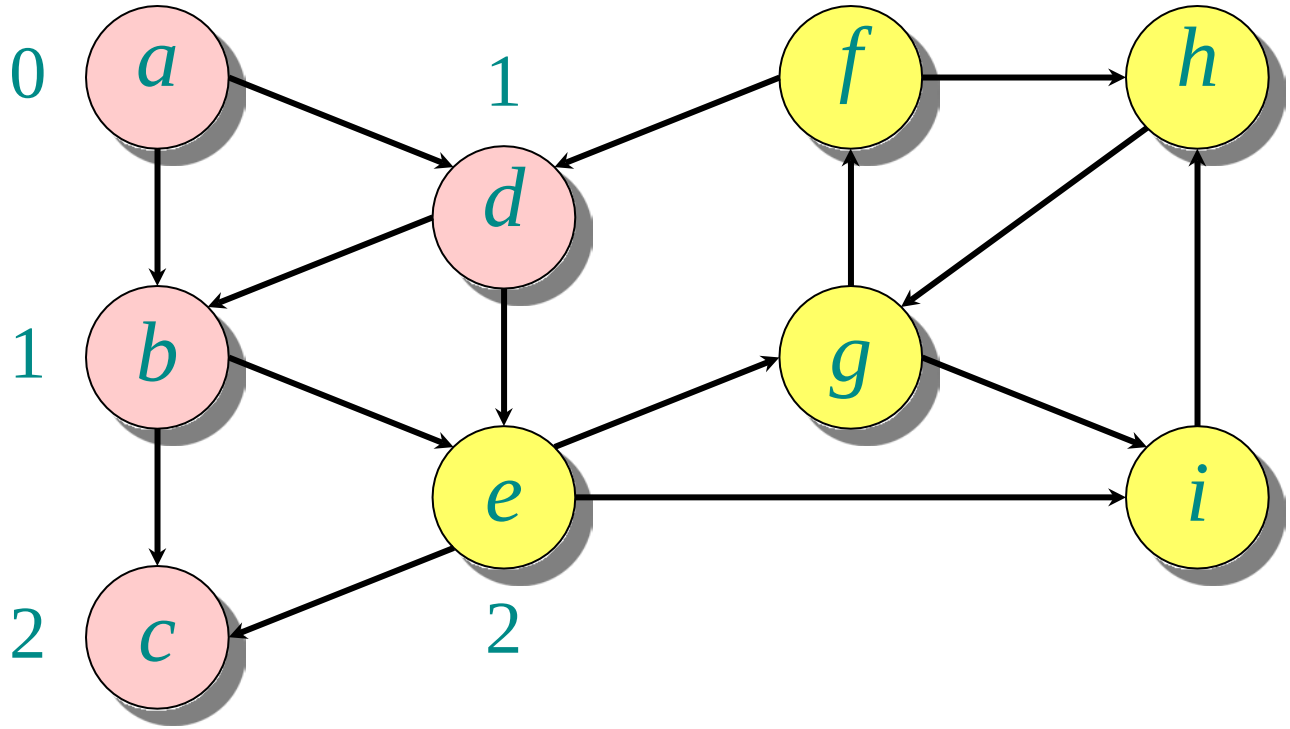
Example of breadth-first search



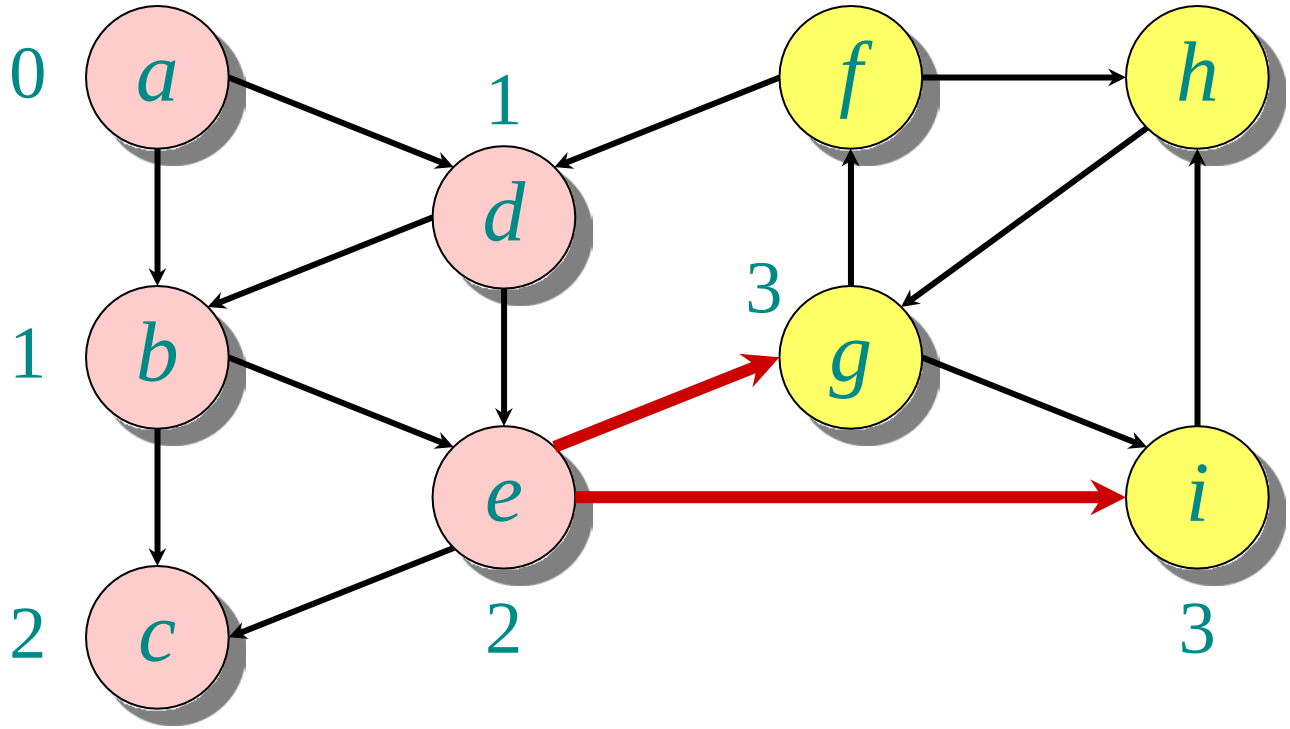
2 2

Q: a b d c e

Example of breadth-first search



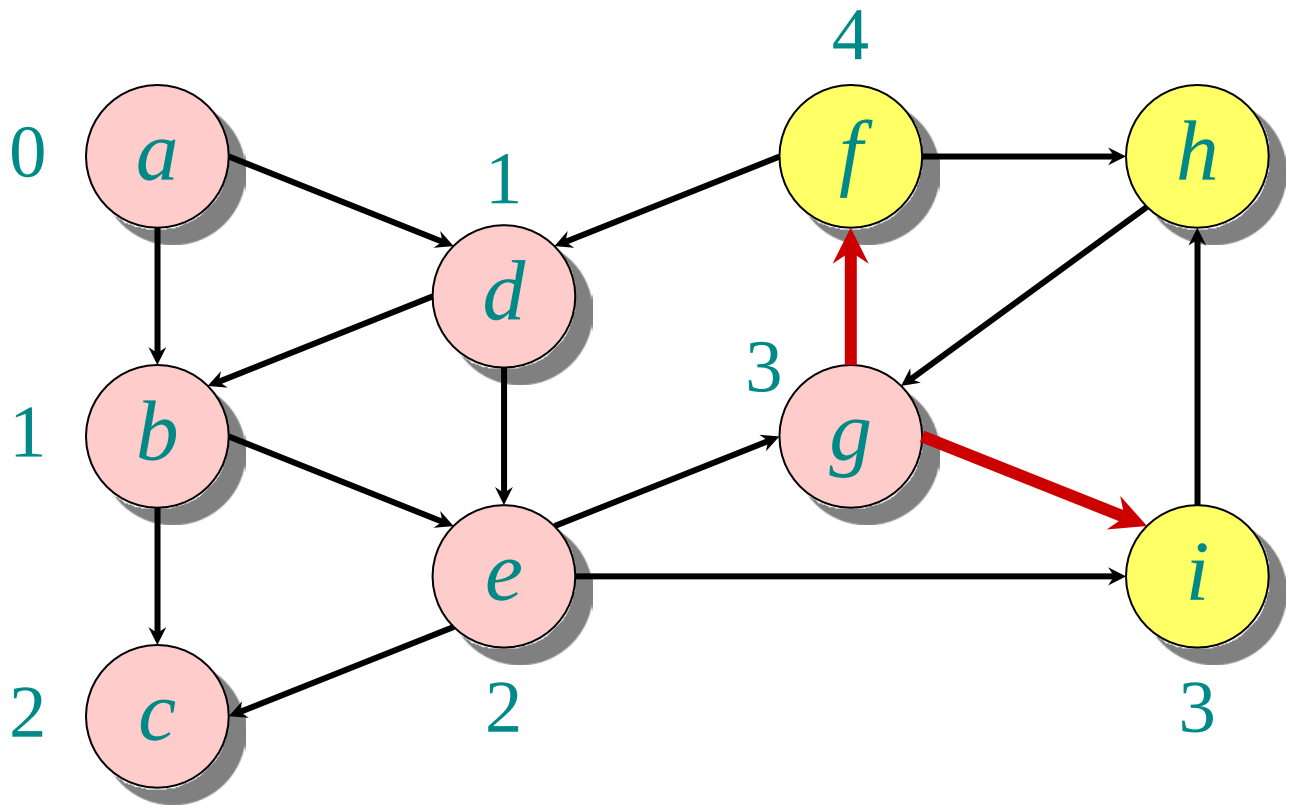
Example of breadth-first search



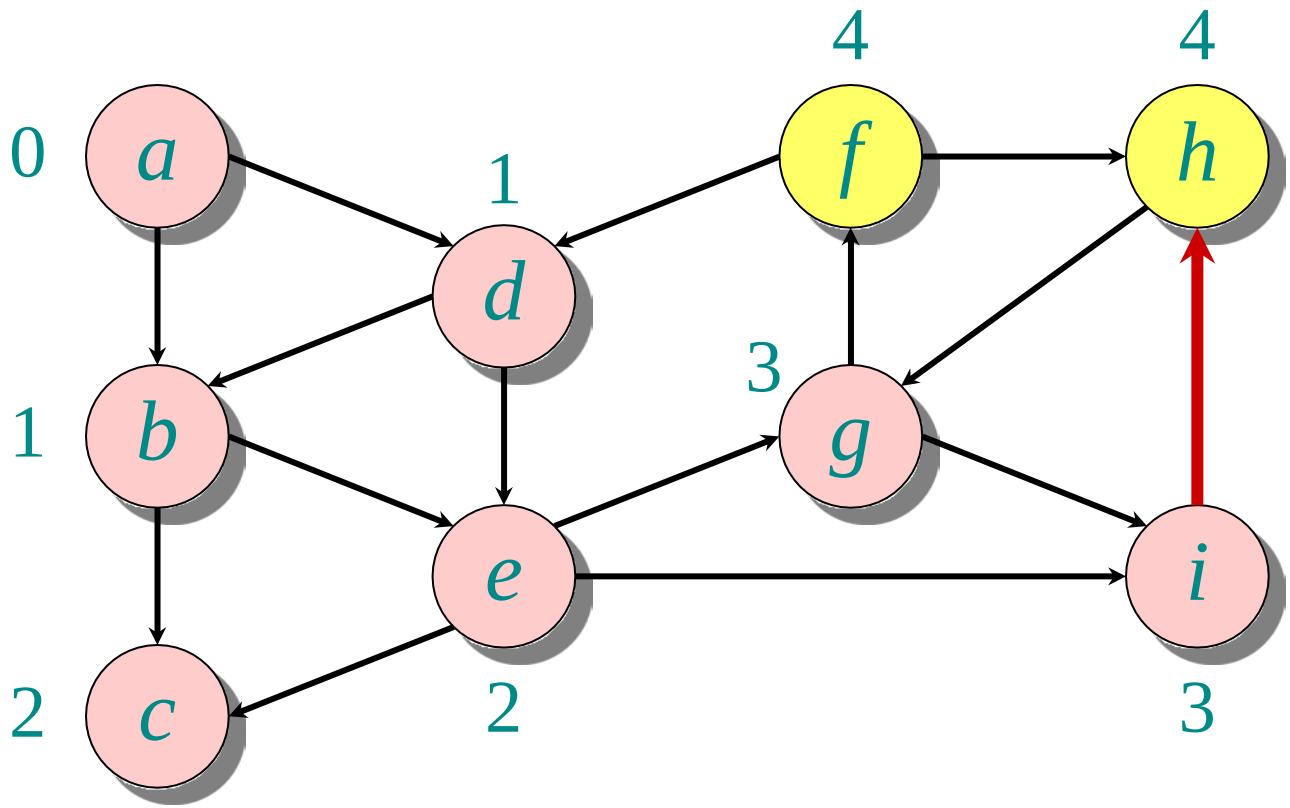
3 3

Q: a b d c e g i

Example of breadth-first search



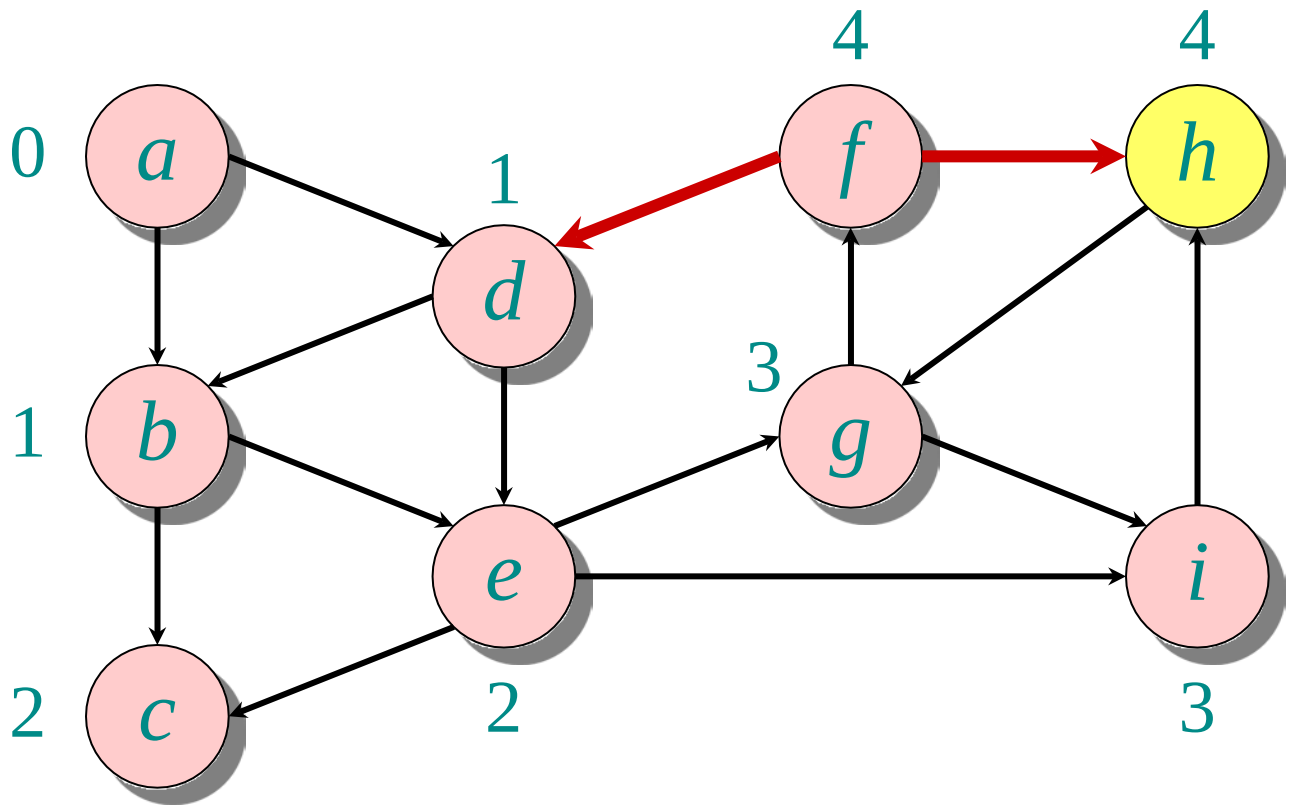
Example of breadth-first search



4 4

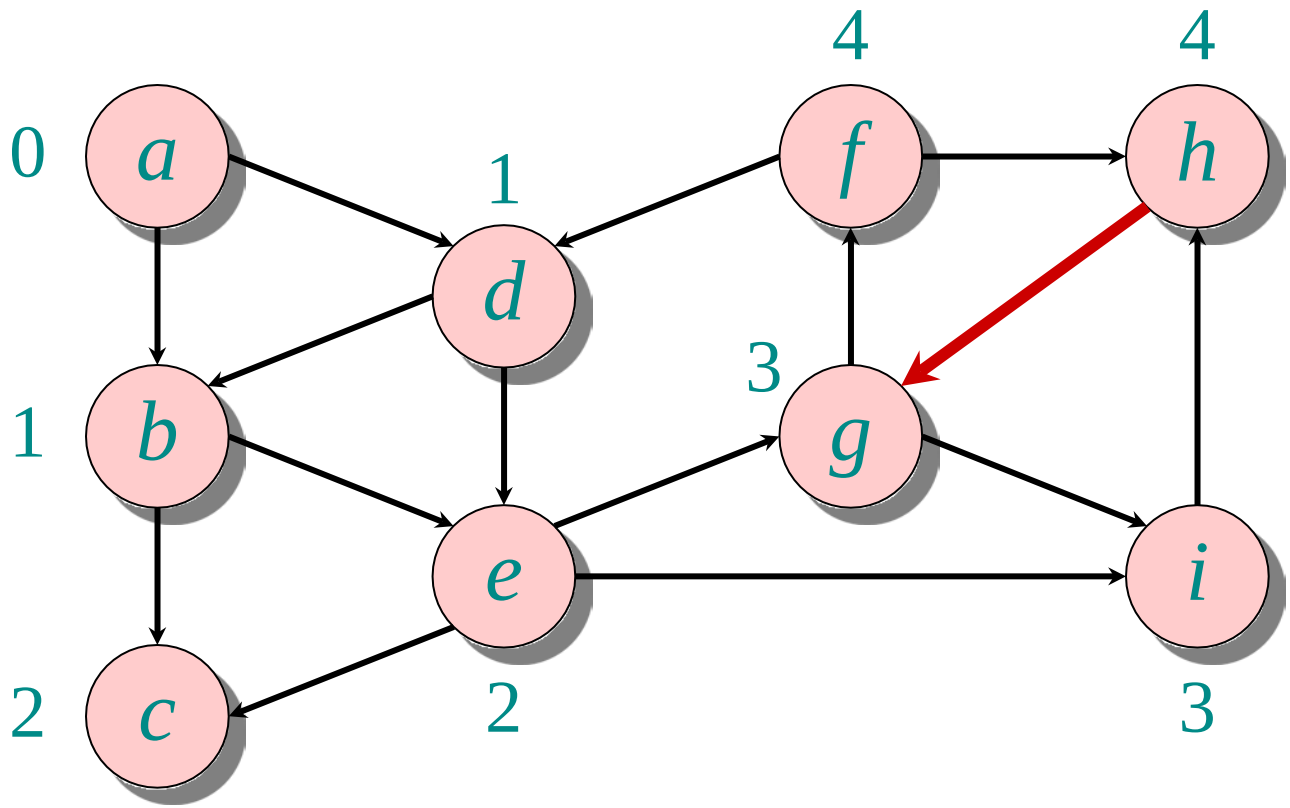
Q: a b d c e g i f h

Example of breadth-first search



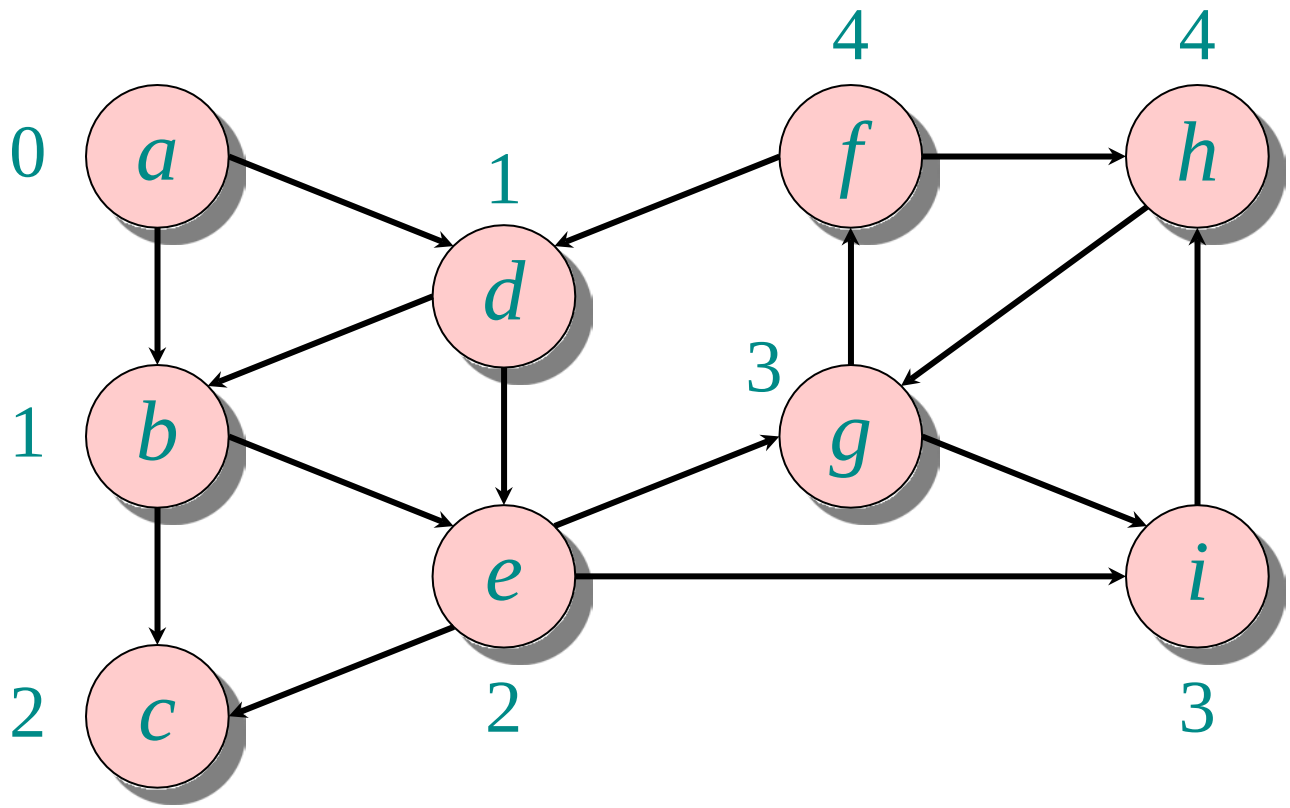
Q: a b d c e g i f h

Example of breadth-first search



Q: a b d c e g i f h

Example of breadth-first search



Q: a b d c e g i f h

Reference

- **Introduction to Algorithms**
  - Single Source Shortest Path
  - Chapter # 24
  - Thomas H. Cormen