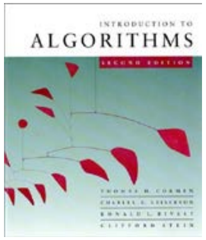
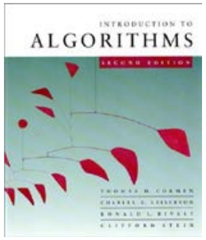


Paths in graphs

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

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

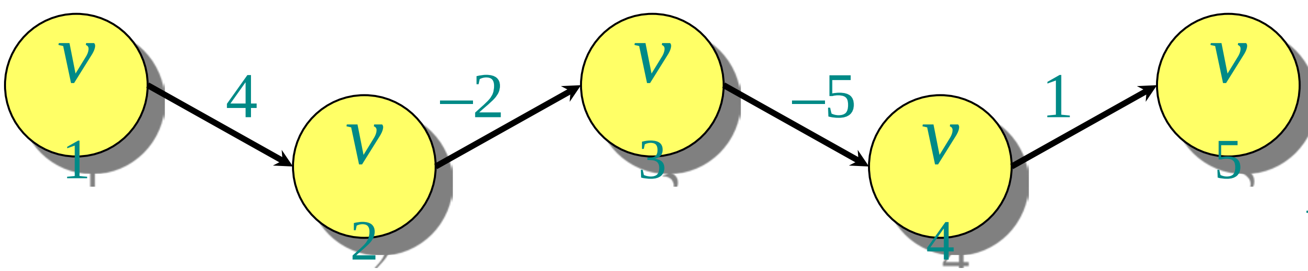


Paths in graphs

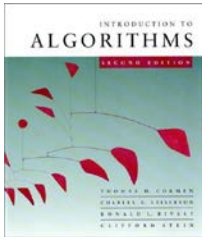
Consider a digraph $G = (V, E)$ with edge-weight function $w : E \rightarrow \mathbb{R}$. The ***weight*** of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is defined to be

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

Example:



$w(p) = –2$

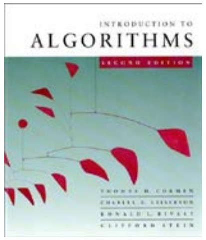


Shortest paths

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

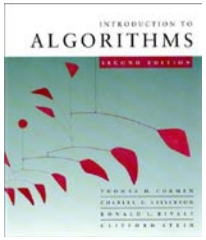
$$\delta(u, v) = \min\{w(p) : p \text{ is a path from } u \text{ to } v\}.$$

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



Well-definedness of shortest paths

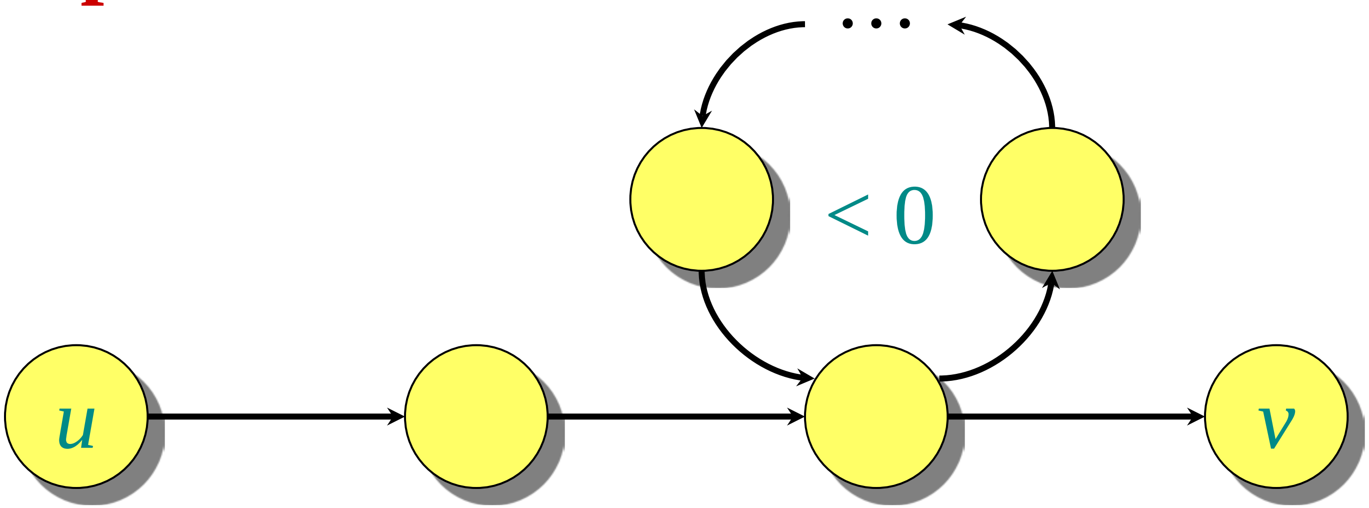
If a graph G contains a negative-weight cycle, then some shortest paths do not exist.

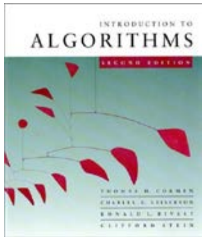


Well-definedness of shortest paths

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

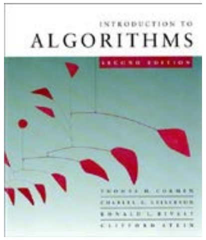
Example:





Optimal substructure

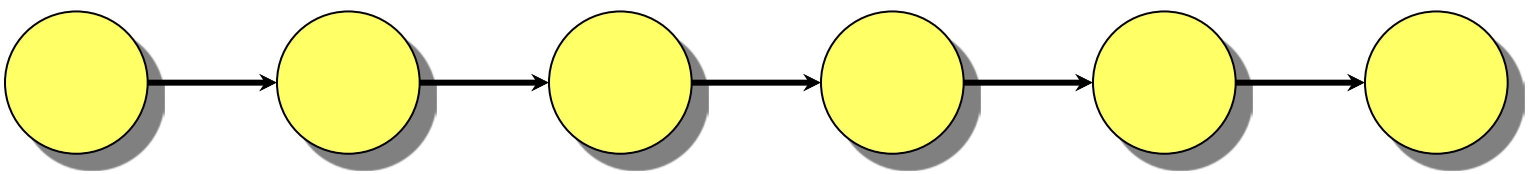
Theorem. A subpath of a shortest path is a shortest path.



Optimal substructure

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

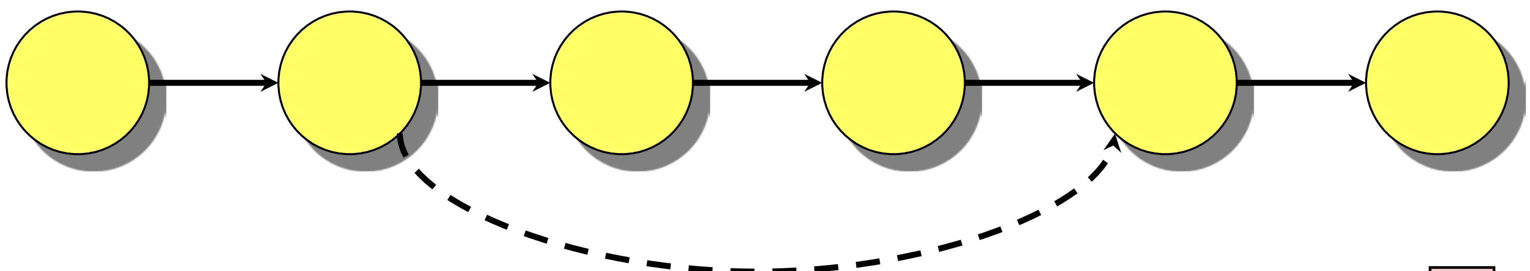
Proof. Cut and paste:



Optimal substructure

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

Proof. Cut and paste:



$\square$

Triangle inequality

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

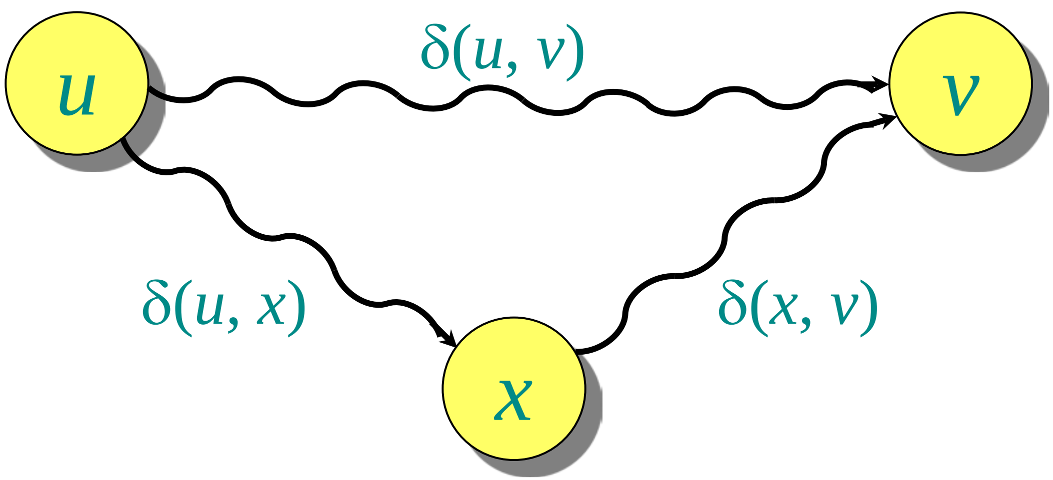
$$\delta(u, v) \le \delta(u, x) + \delta(x, v).$$

Triangle inequality

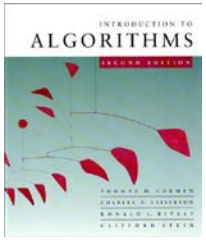
Theorem. For all $u, v, x \in V$, we have

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

Proof.



$\square$

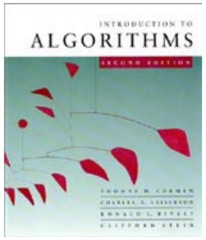


Single-source shortest paths (nonnegative edge weights)

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

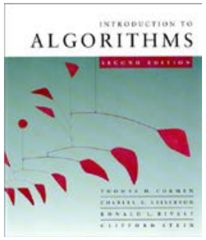
$d[s] \leftarrow 0$

for each $v \in V - \{s\}$

  do $d[v] \leftarrow \infty$

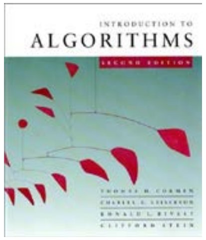
$S \leftarrow \emptyset$

$Q \leftarrow V$ ⊳ 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 $d[v] \leftarrow \infty$

$S \leftarrow \varnothing$

$Q \leftarrow V$ ⊳ Q is a priority queue maintaining $V - S$, keyed on $d[v]$

while $Q \neq \varnothing$

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

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

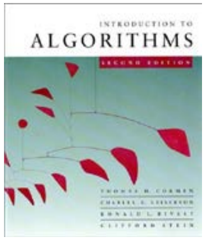
    for each $v \in Adj[u]$

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

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

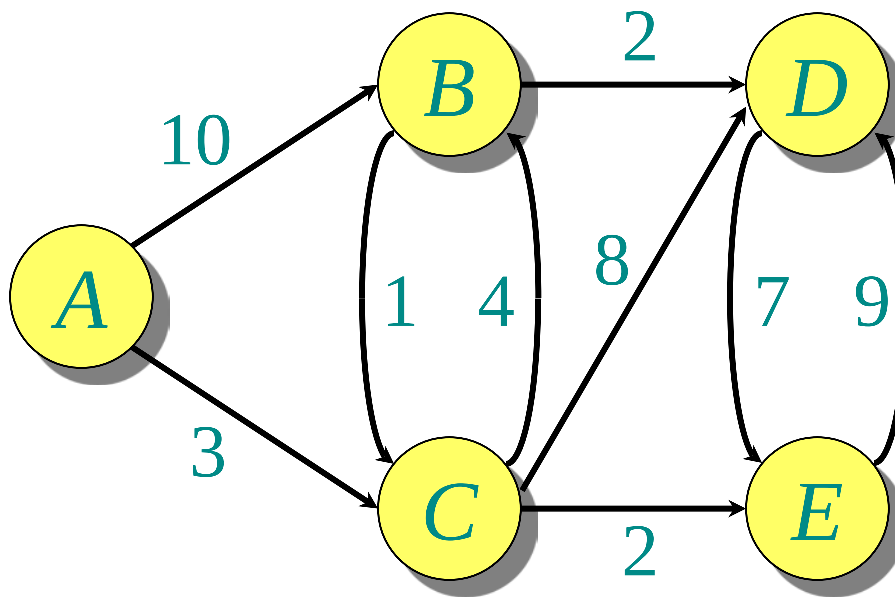
relaxation step

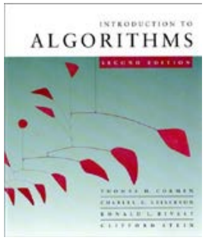
Implicit DECREASE-KEY



Example of Dijkstra's algorithm

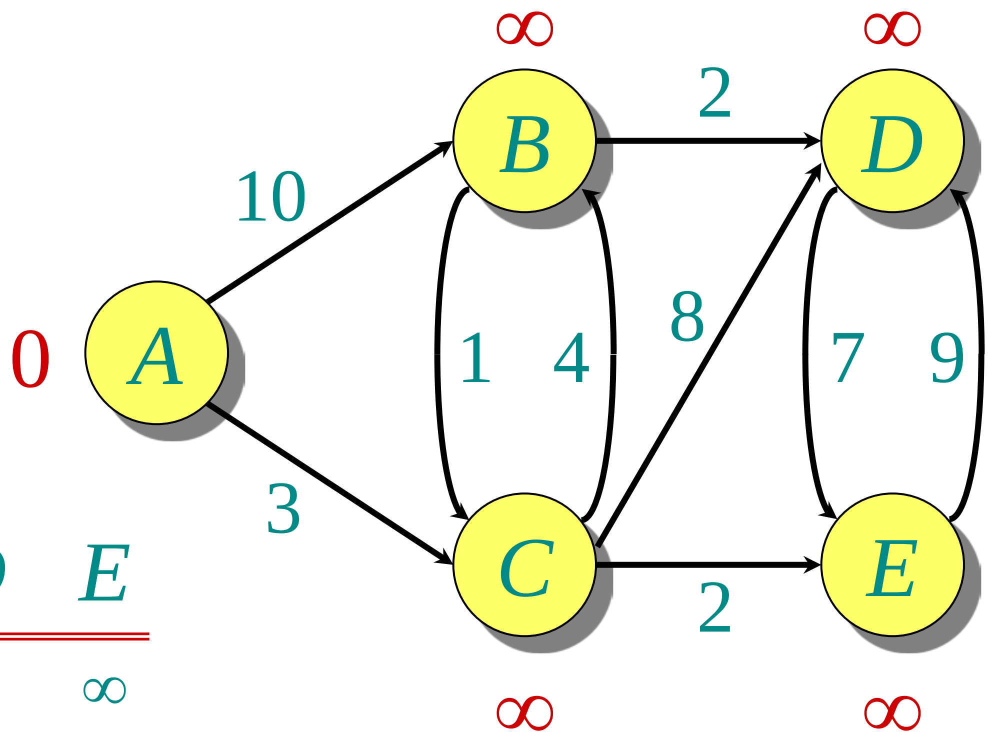
Graph with nonnegative edge weights:





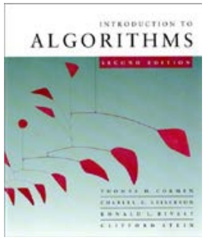
Example of Dijkstra's algorithm

Initialize:



Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞

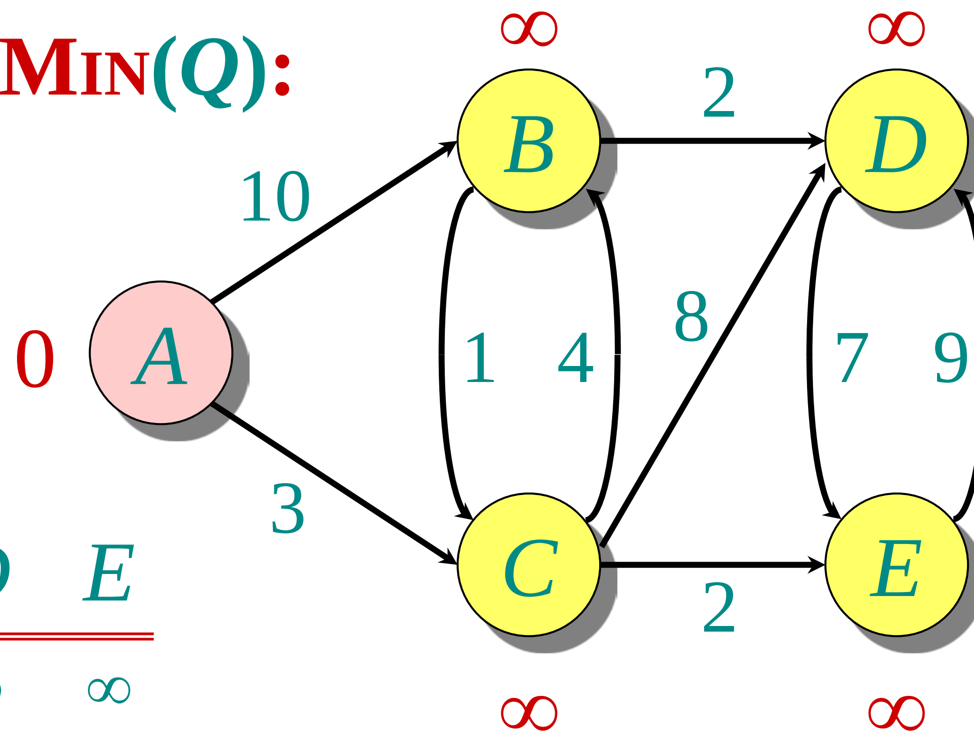
S: { }



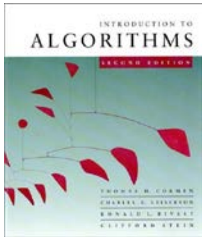
Example of Dijkstra's algorithm

"A" ← **EXTRACT-MIN(Q):**

Q:	$\color{lavender}{A}$	B	C	D	E
	0	∞	∞	∞	∞

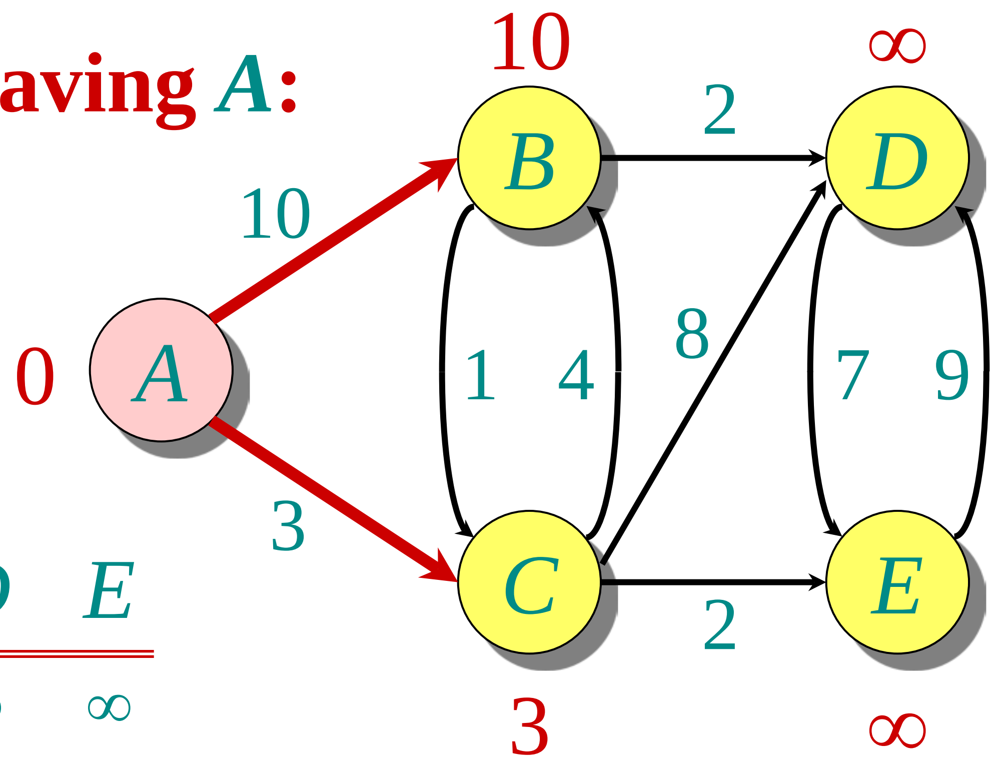


S: { A }



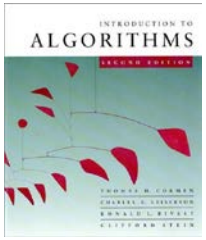
Example of Dijkstra's algorithm

Relax all edges leaving *A*:



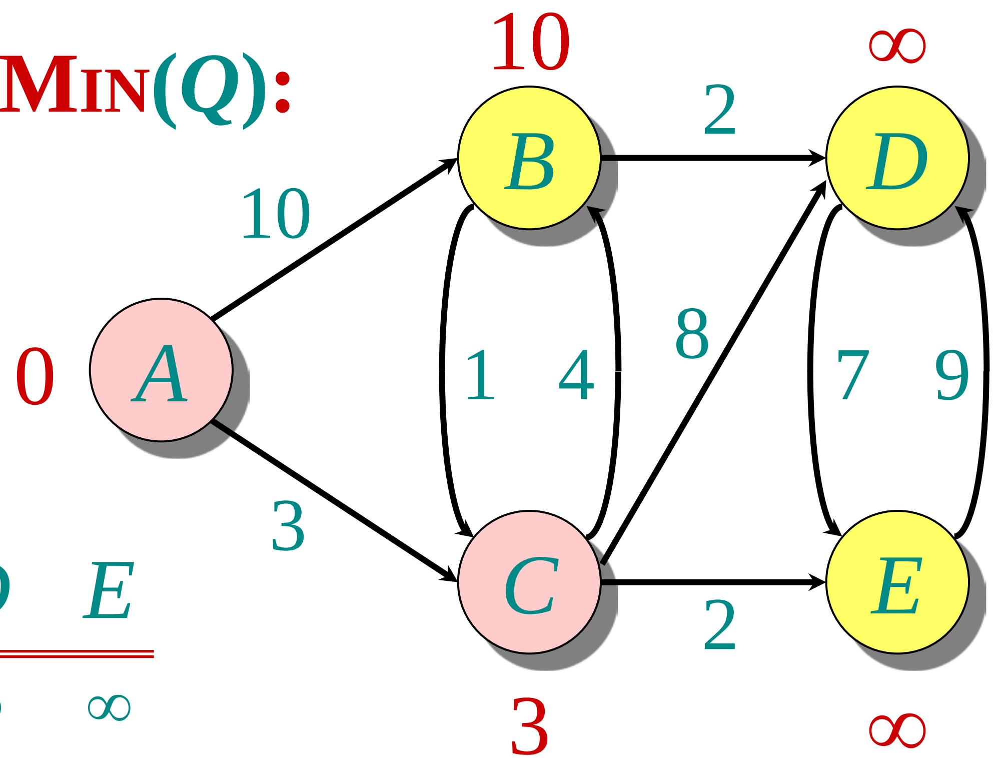
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { *A* }



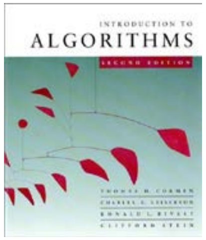
Example of Dijkstra's algorithm

"C" ← **EXTRACT-MIN(Q):**



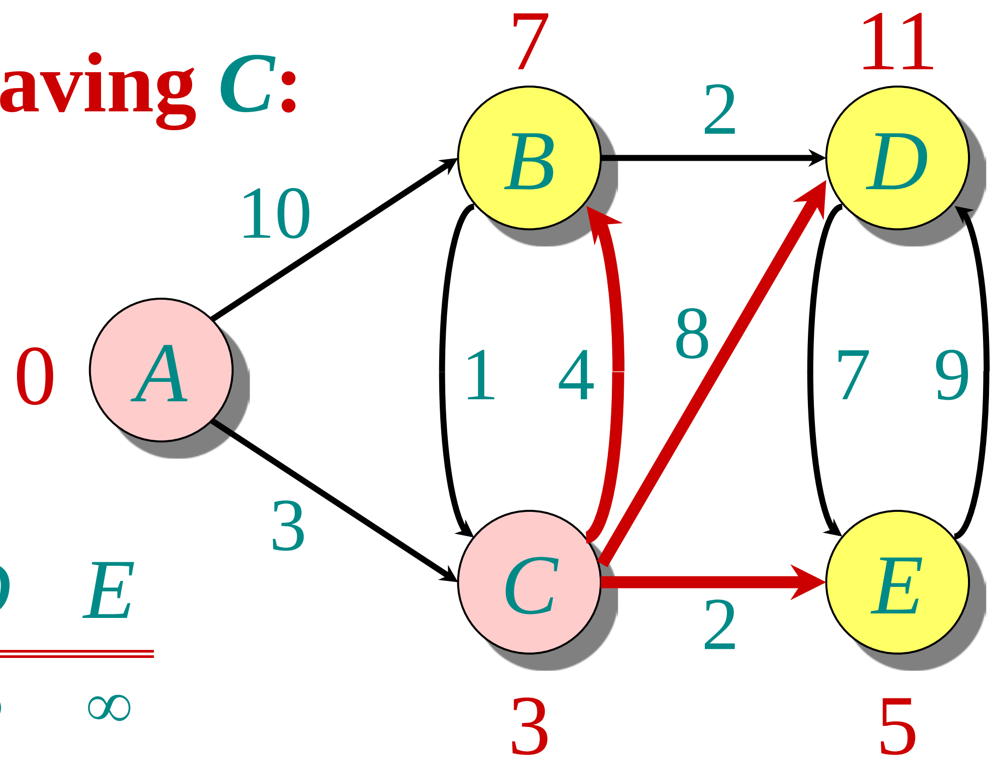
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }



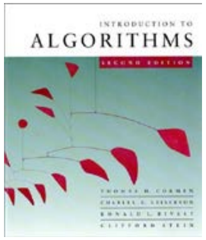
Example of Dijkstra's algorithm

Relax all edges leaving *C*:



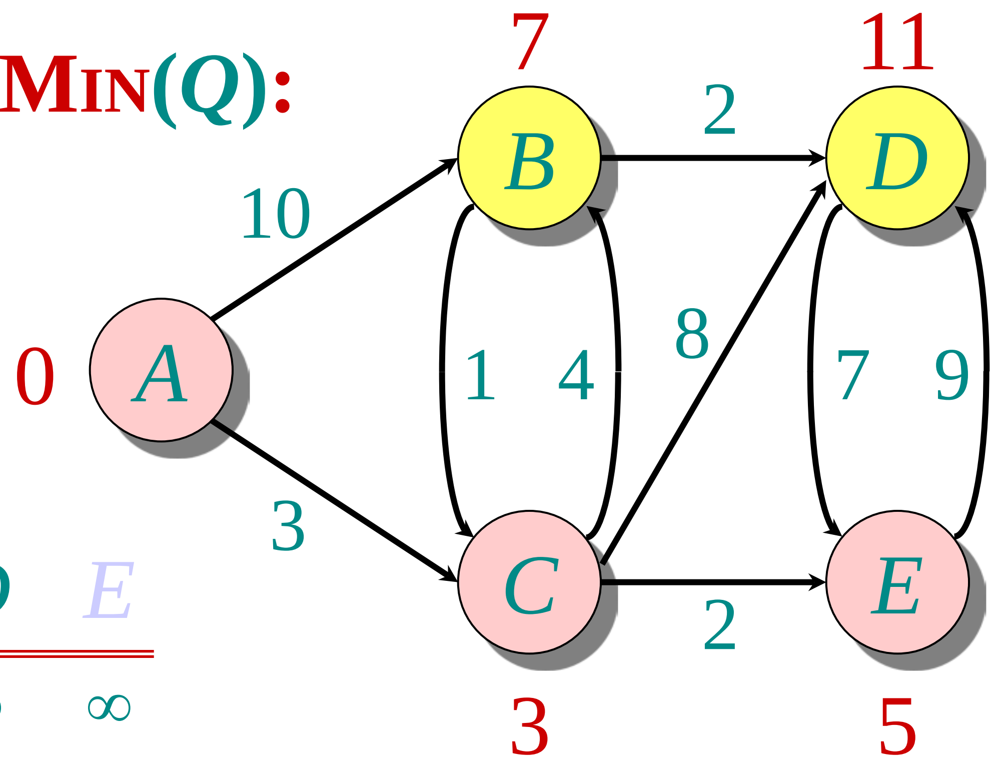
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { *A*, *C* }



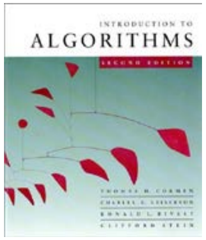
Example of Dijkstra's algorithm

"E" $\leftarrow$ EXTRACT-MIN(Q):



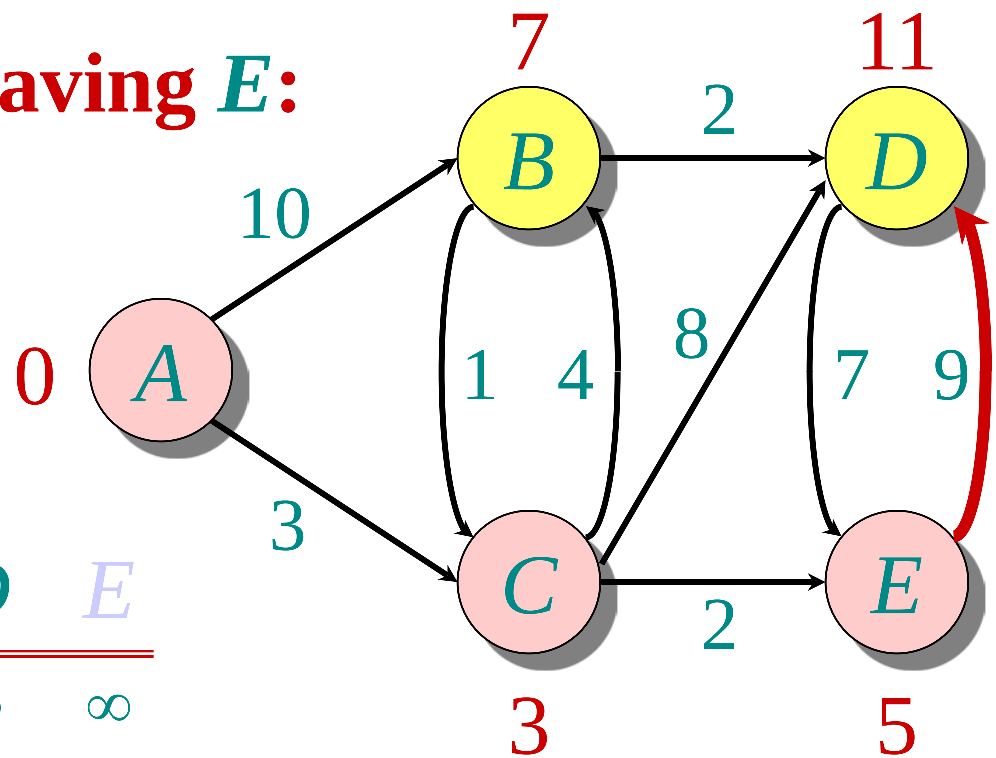
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C, E }



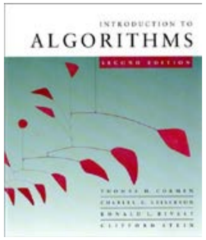
Example of Dijkstra's algorithm

Relax all edges leaving *E*:



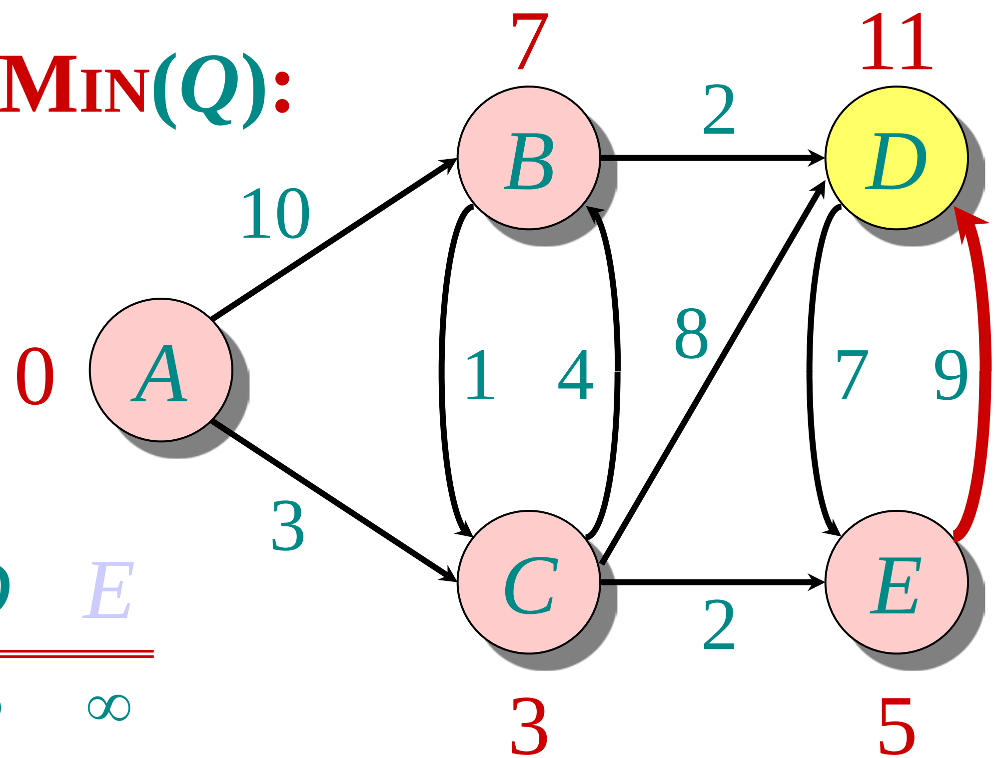
Q:	*A*	*B*	*C*	*D*	*E*
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

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



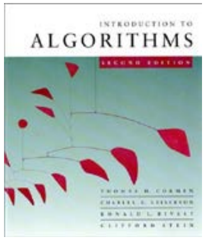
Example of Dijkstra's algorithm

"B" ← EXTRACT-MIN(Q):



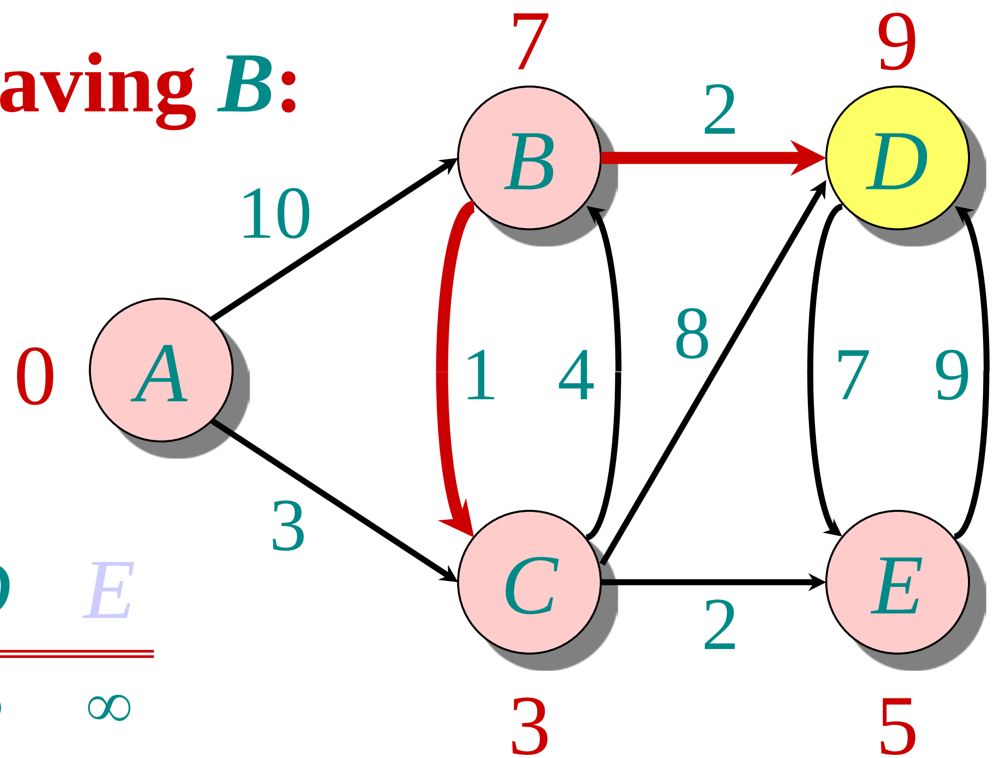
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E, B }



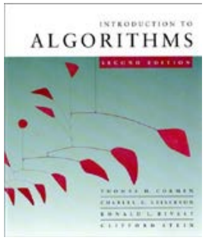
Example of Dijkstra's algorithm

Relax all edges leaving *B*:



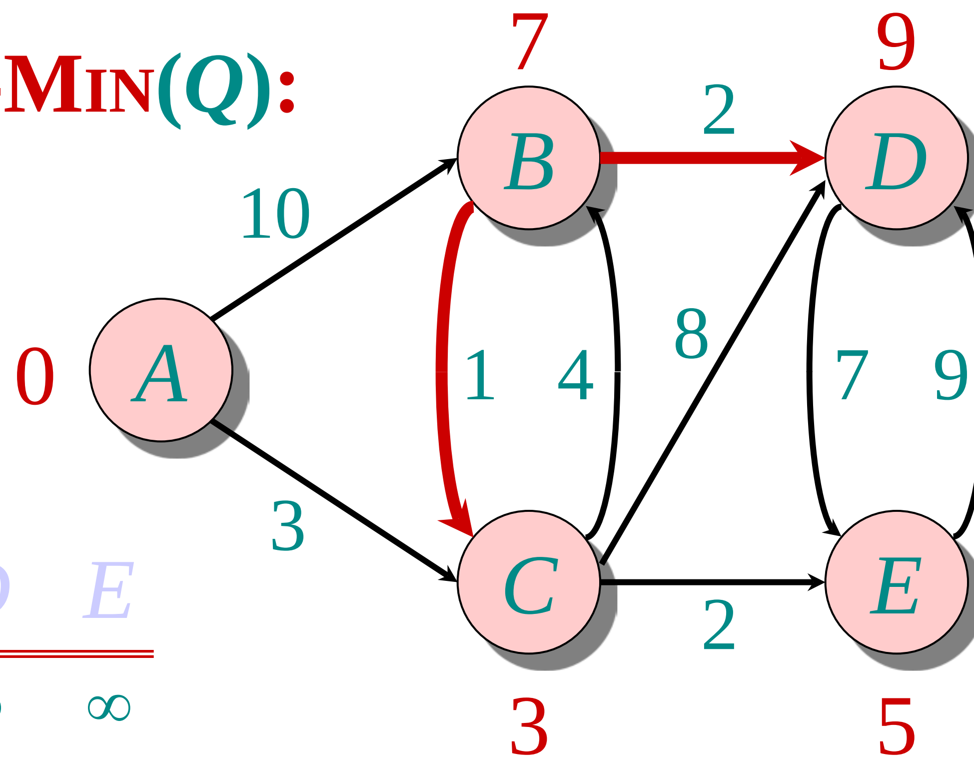
Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	
				9	

S: { A, C, E, B }



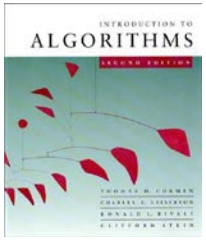
Example of Dijkstra's algorithm

"D" $\leftarrow$ 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 $Q \neq \emptyset$

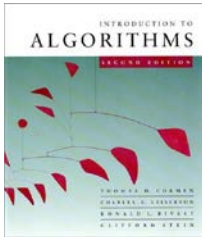
  do $u \leftarrow$ EXTRACT-MIN(Q)

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

    for each $v \in Adj[u]$

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

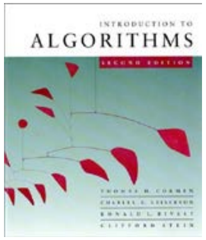
        then $d[v] \leftarrow d[u] + w(u, v)$



Analysis of Dijkstra

$|V|$ times

while $Q \neq \emptyset$
  do $u \leftarrow \textsc{Extract-Min}(Q)$
    $S \leftarrow S \cup \{u\}$
    for each $v \in Adj[u]$
      do if $d[v] > d[u] + w(u, v)$
        then $d[v] \leftarrow d[u] + w(u, v)$

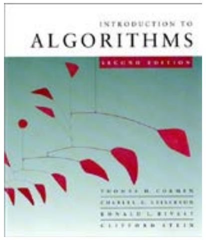


Analysis of Dijkstra

$|V|$ times

while $Q \neq \emptyset$
  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 $d[v] \leftarrow d[u] + w(u, v)$

$degree(u)$ times



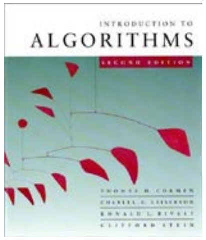
Analysis of Dijkstra

$|V|$ times

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

degree(u) times

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



Analysis of Dijkstra

$|V|$ times

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

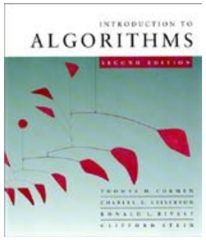
degree(u) times

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

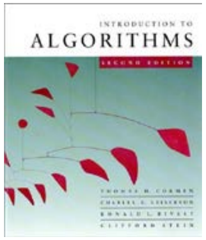
Note: Same formula as in the analysis of Prim's minimum spanning tree algorithm.



Analysis of Dijkstra (continued)

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

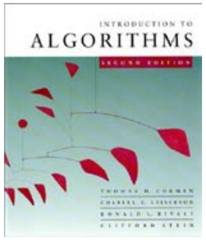
Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total



Analysis of Dijkstra (continued)

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

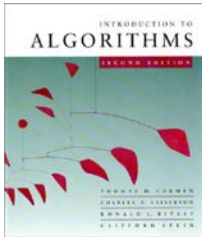
Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$



Analysis of Dijkstra (continued)

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

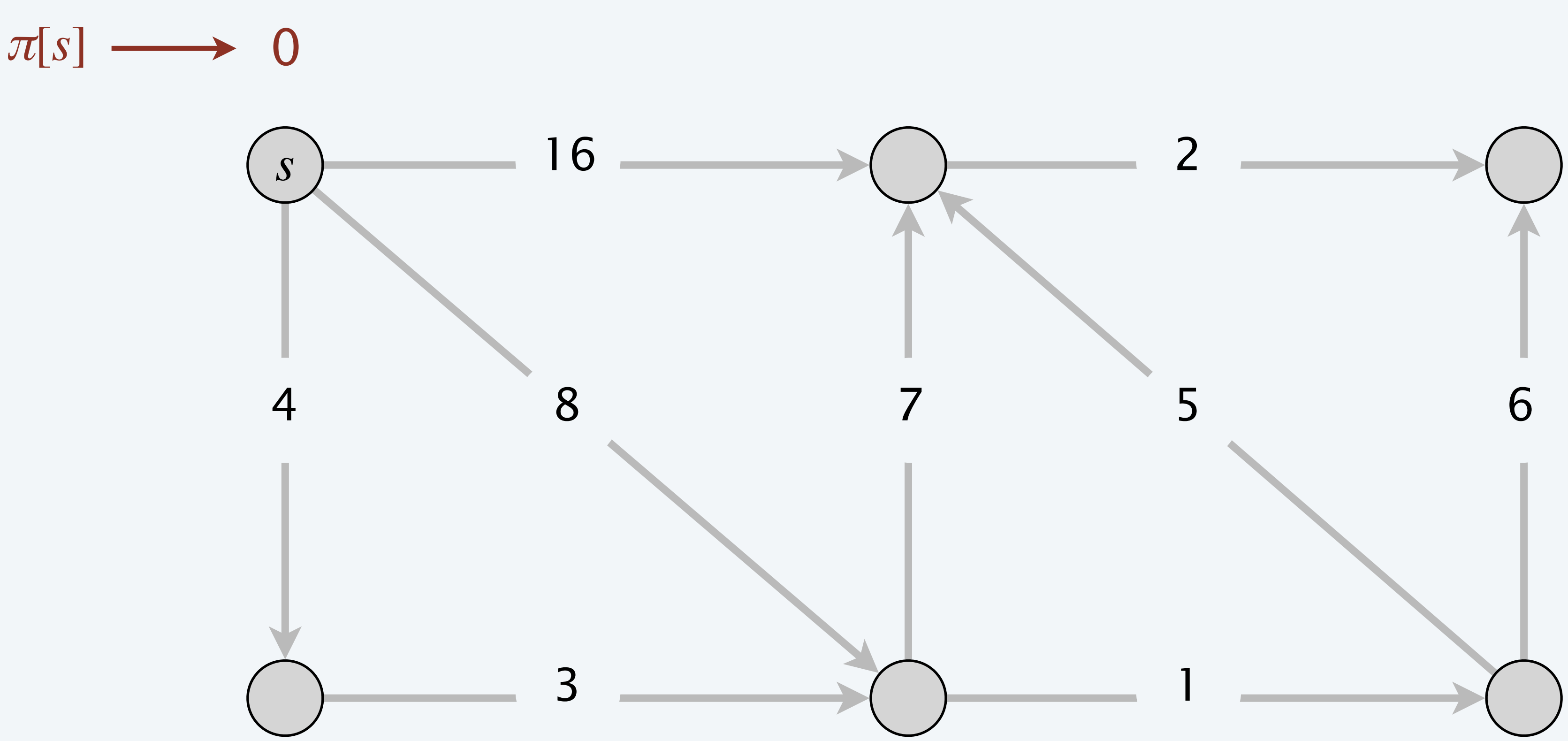


Analysis of Dijkstra (continued)

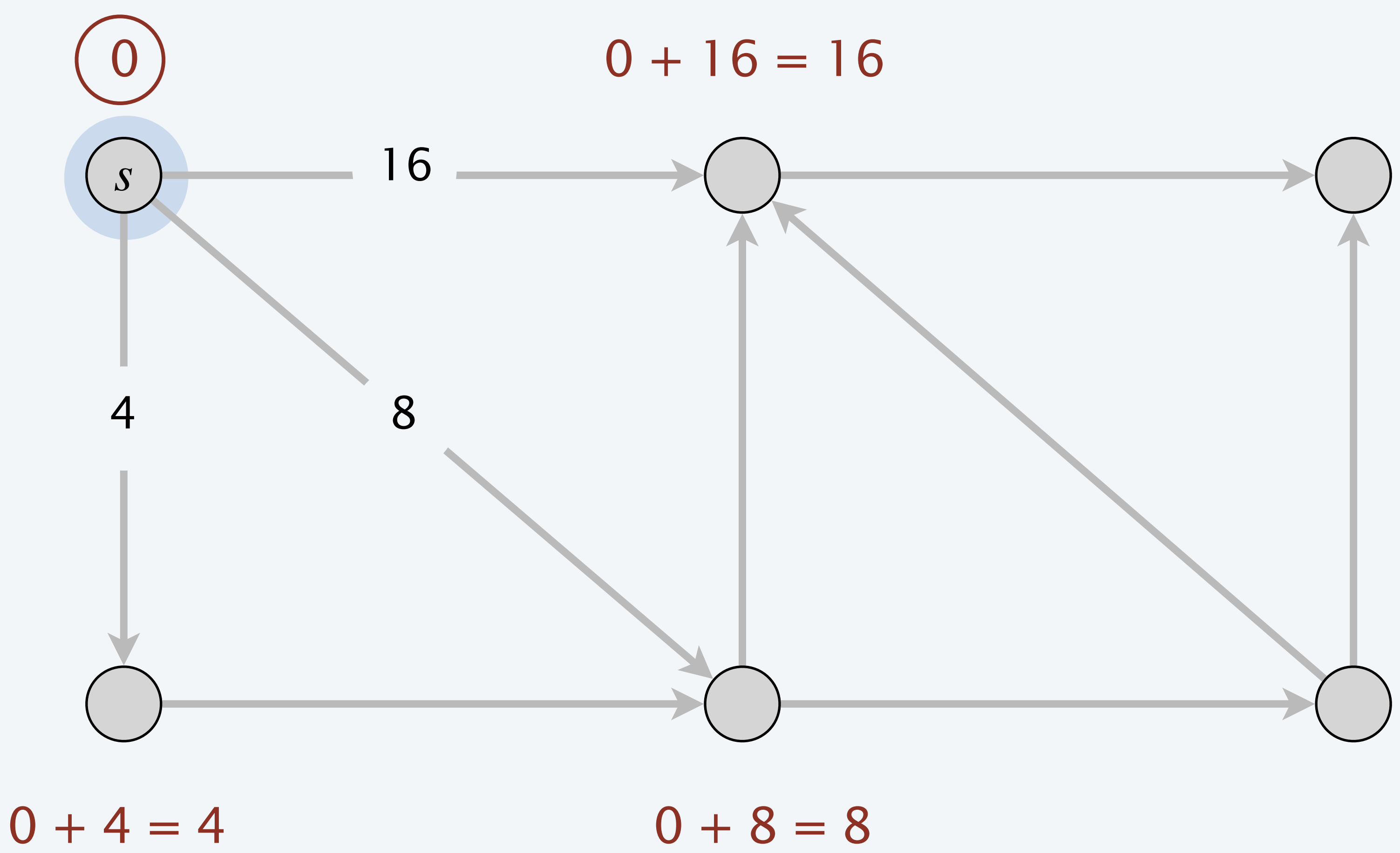
Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

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

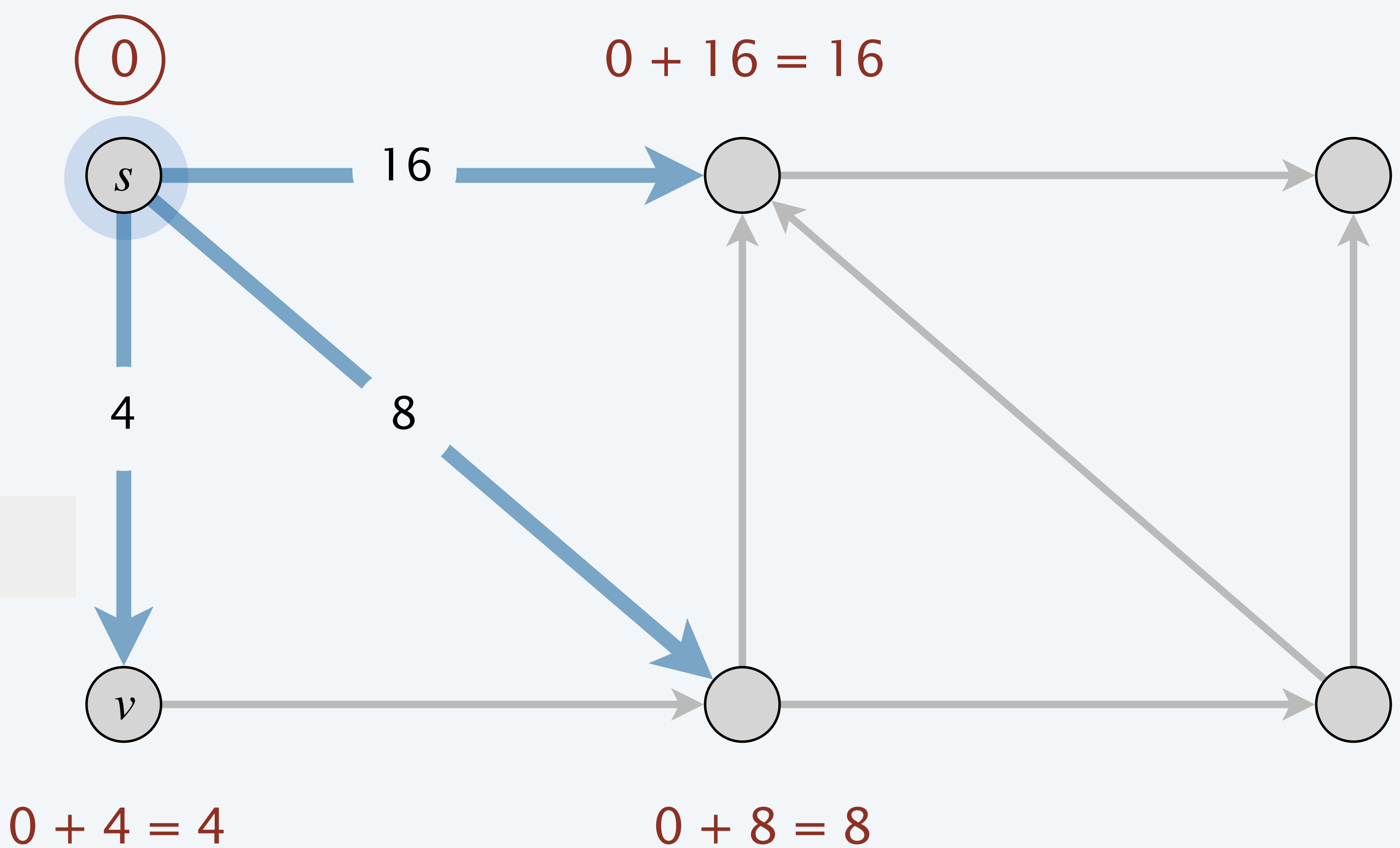
Dijkstra's algorithm demo (efficient implementation)



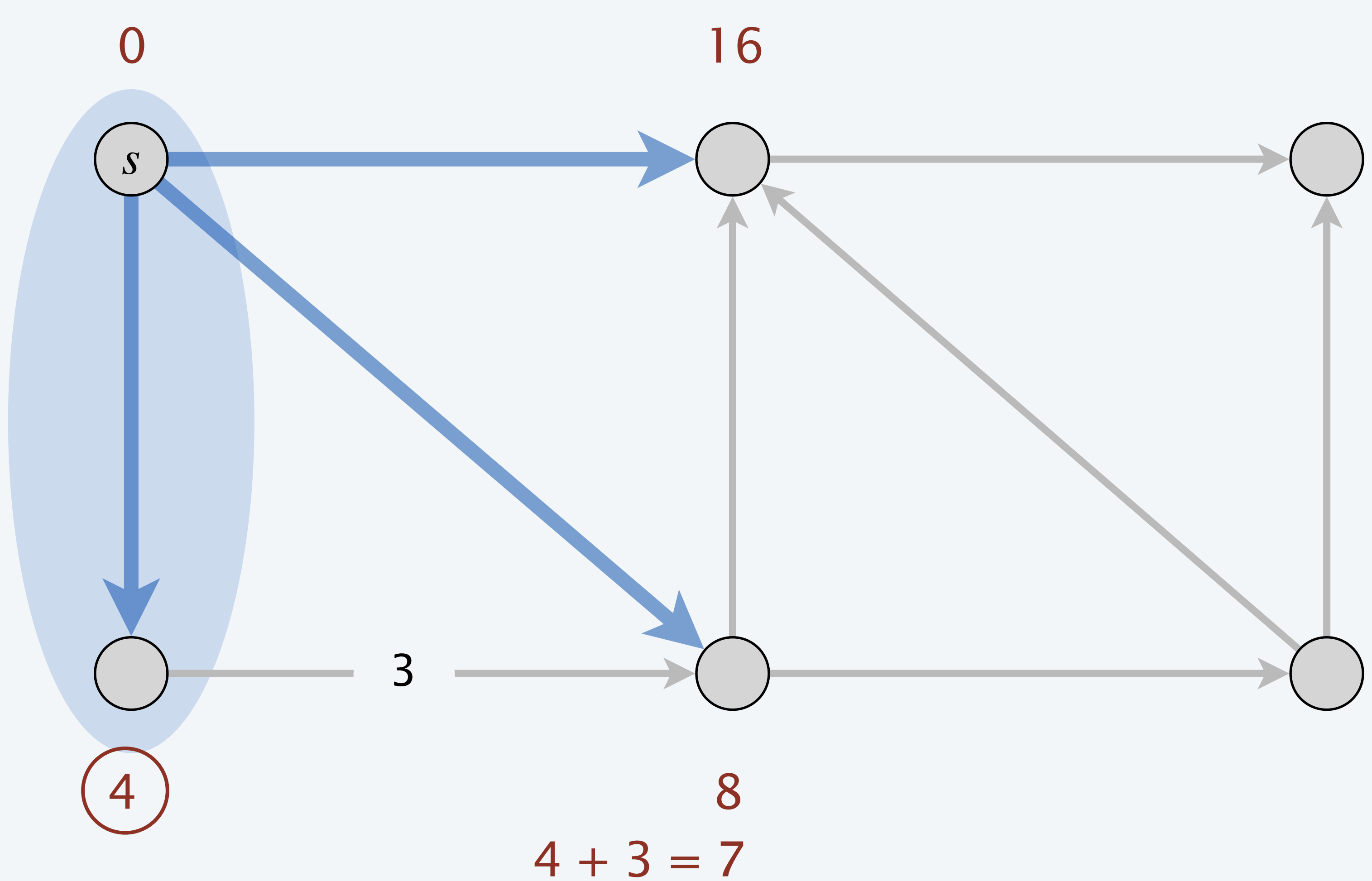
Dijkstra's algorithm demo (efficient implementation)



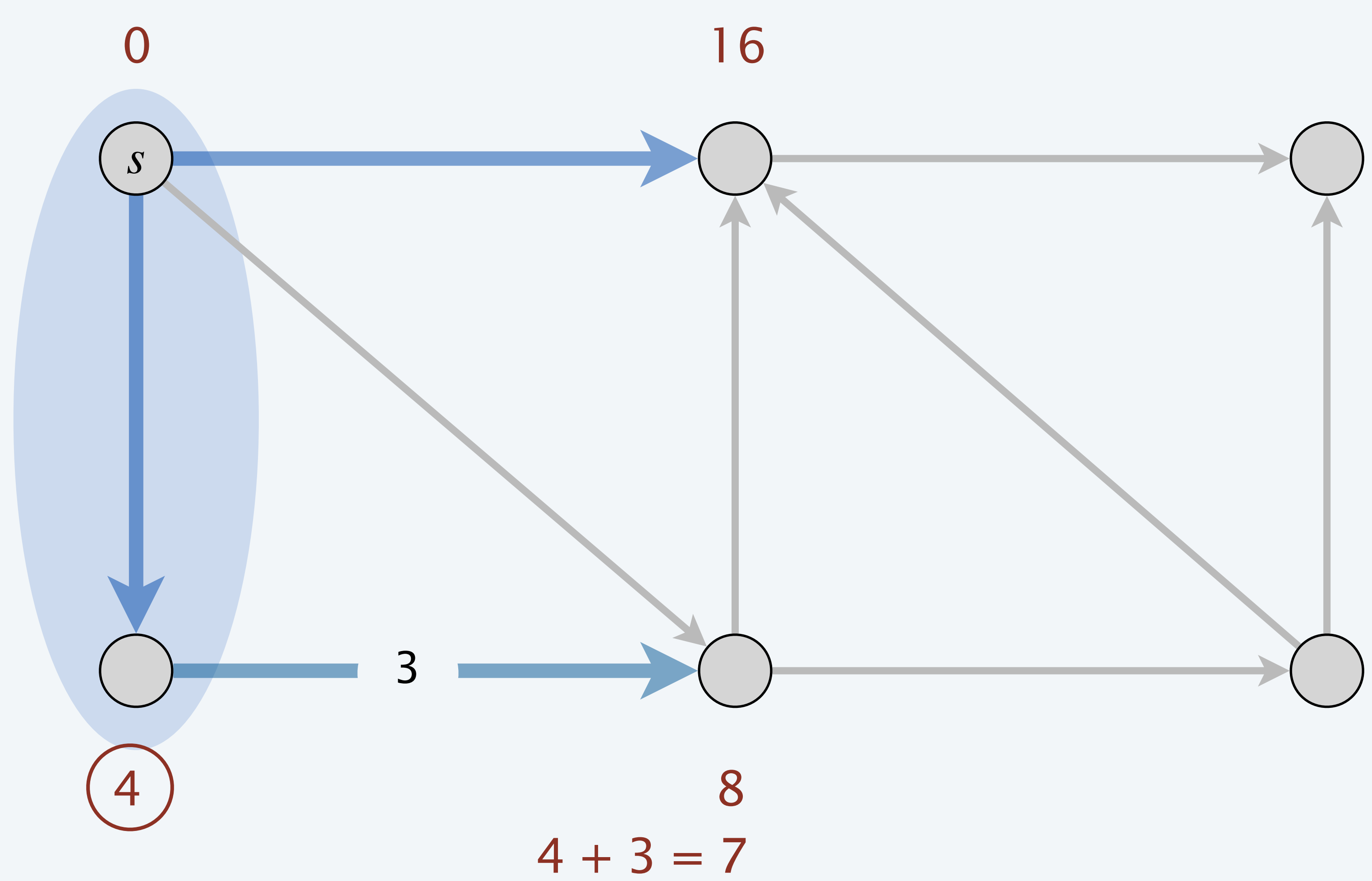
Dijkstra's algorithm demo (efficient implementation)



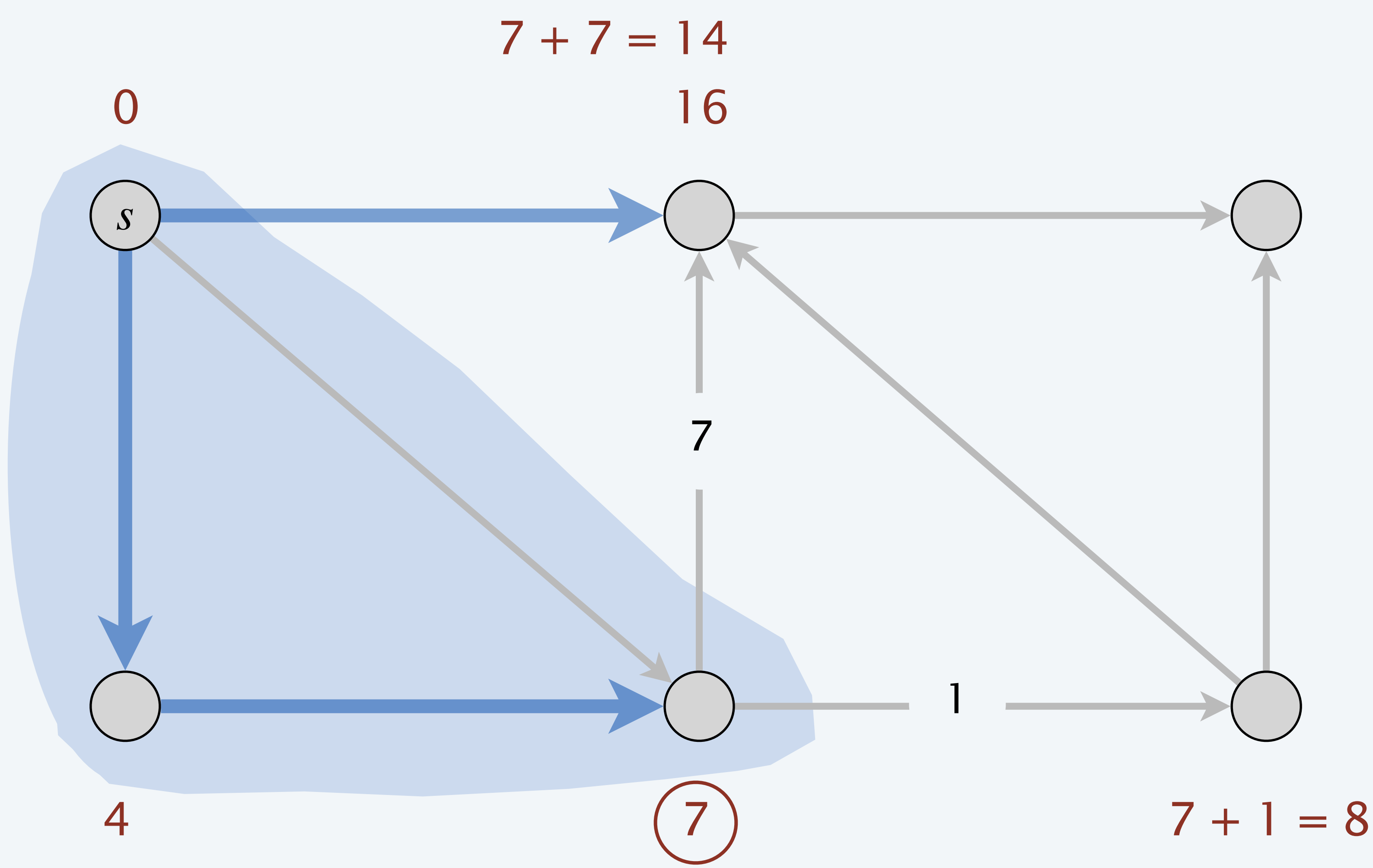
Dijkstra's algorithm demo (efficient implementation)



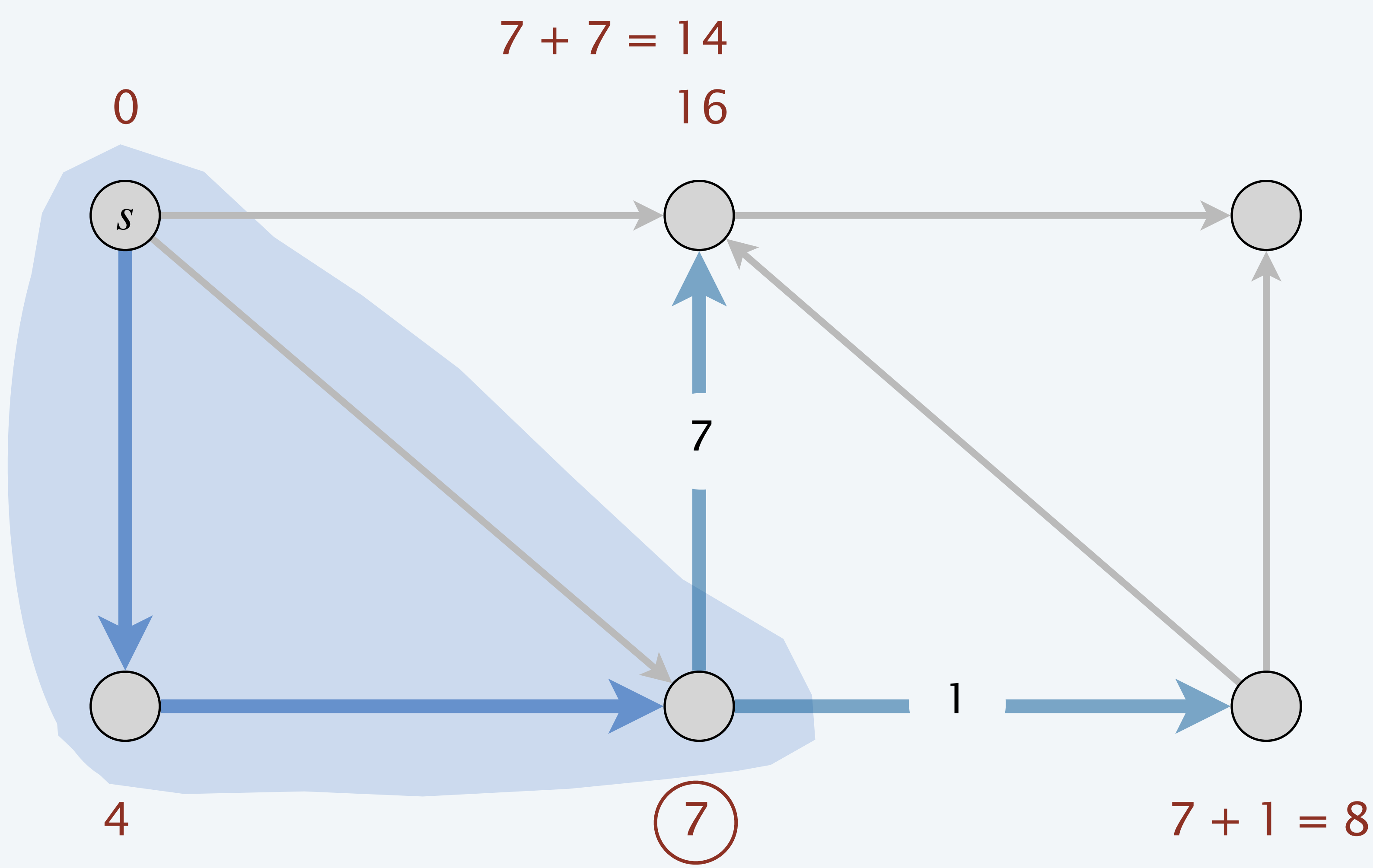
Dijkstra's algorithm demo (efficient implementation)



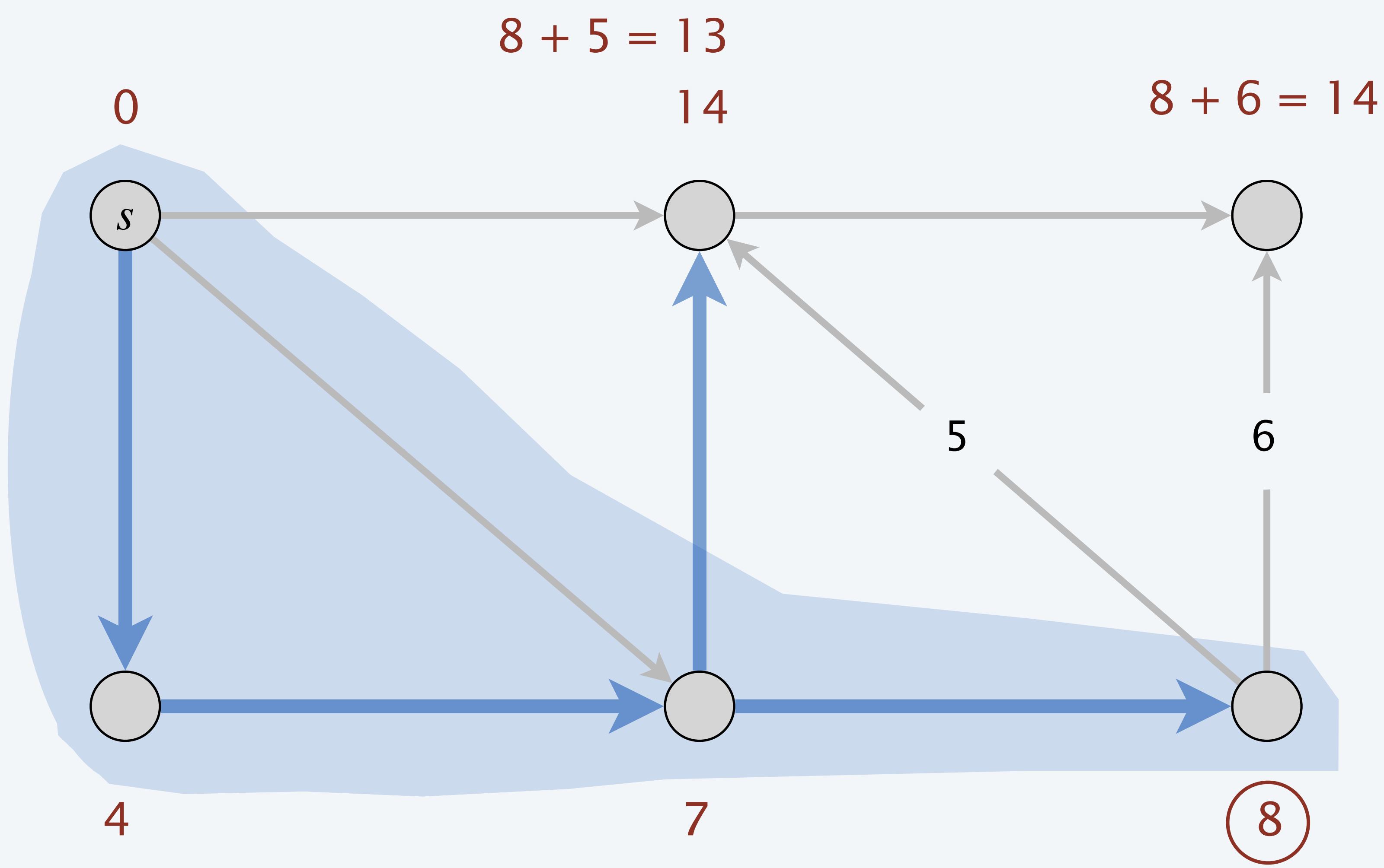
Dijkstra's algorithm demo (efficient implementation)



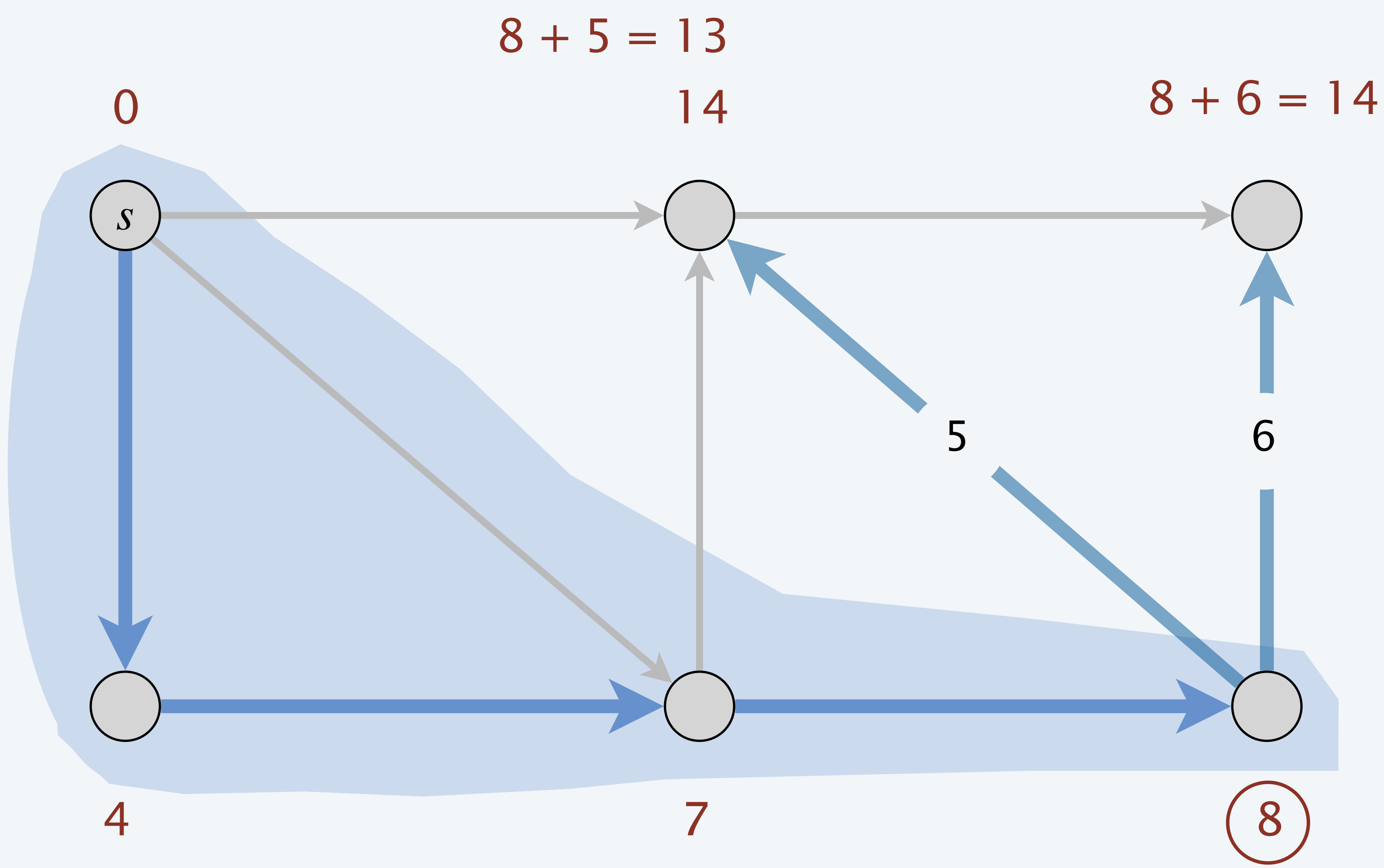
Dijkstra's algorithm demo (efficient implementation)



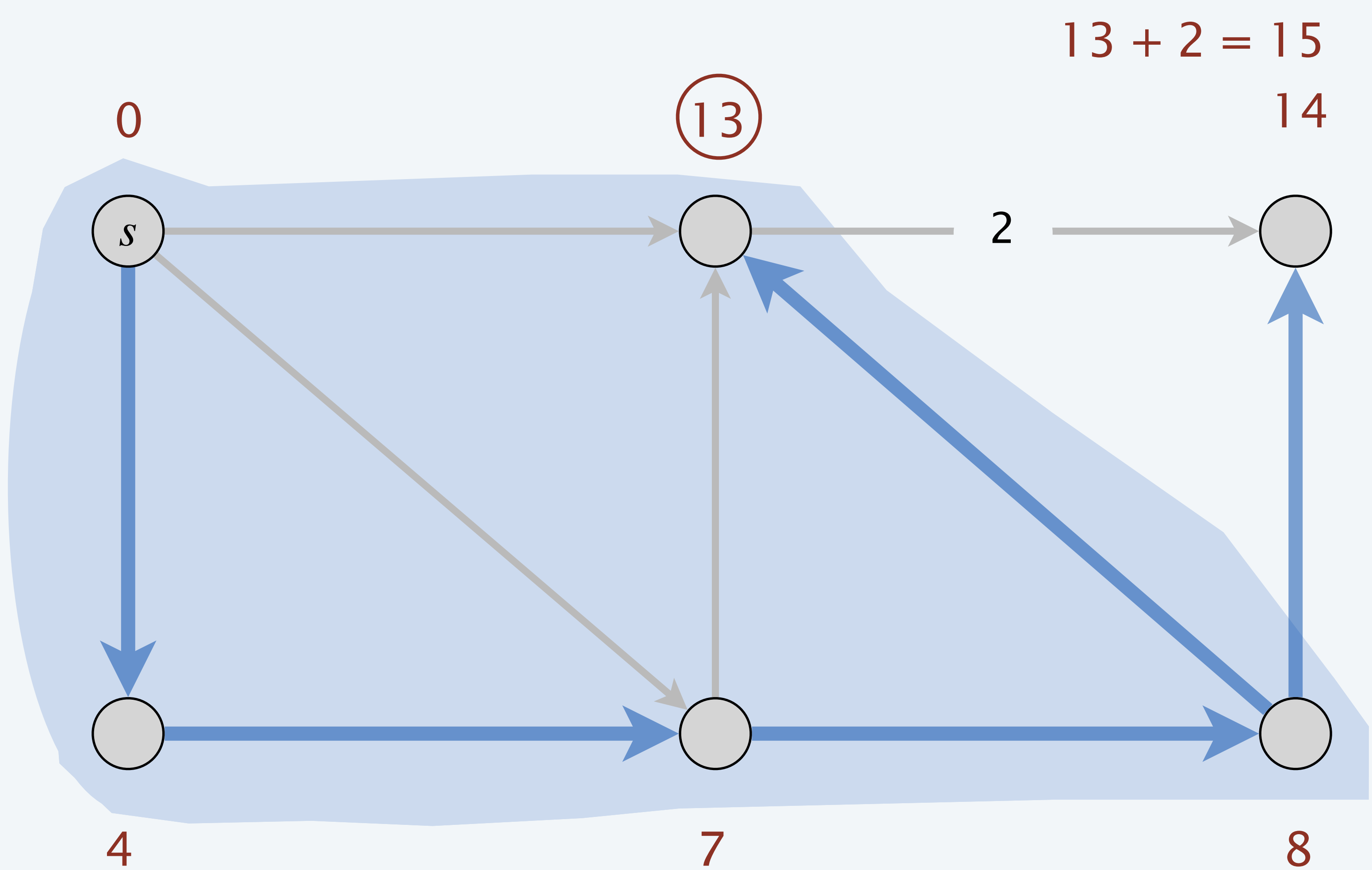
Dijkstra's algorithm demo (efficient implementation)



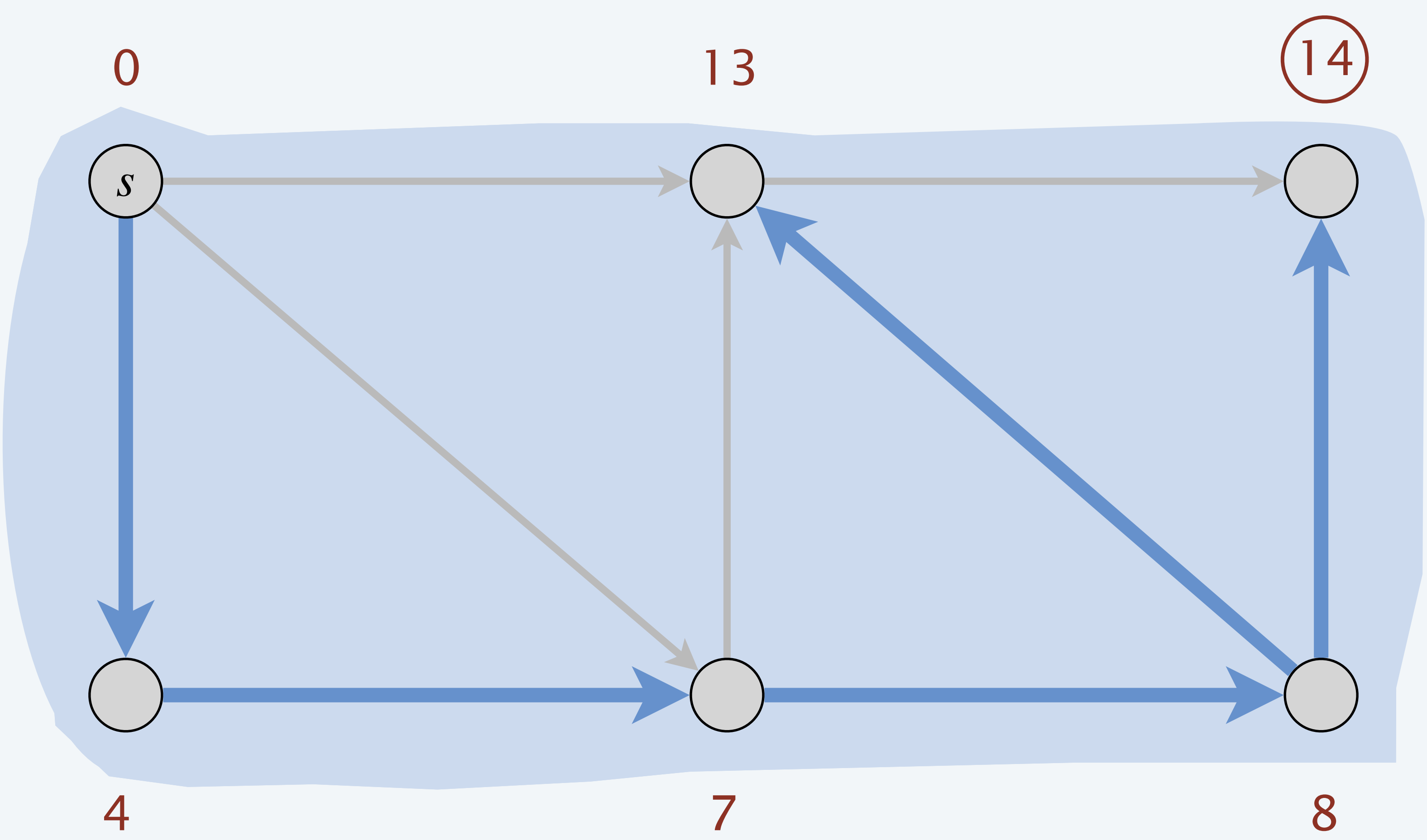
Dijkstra's algorithm demo (efficient implementation)



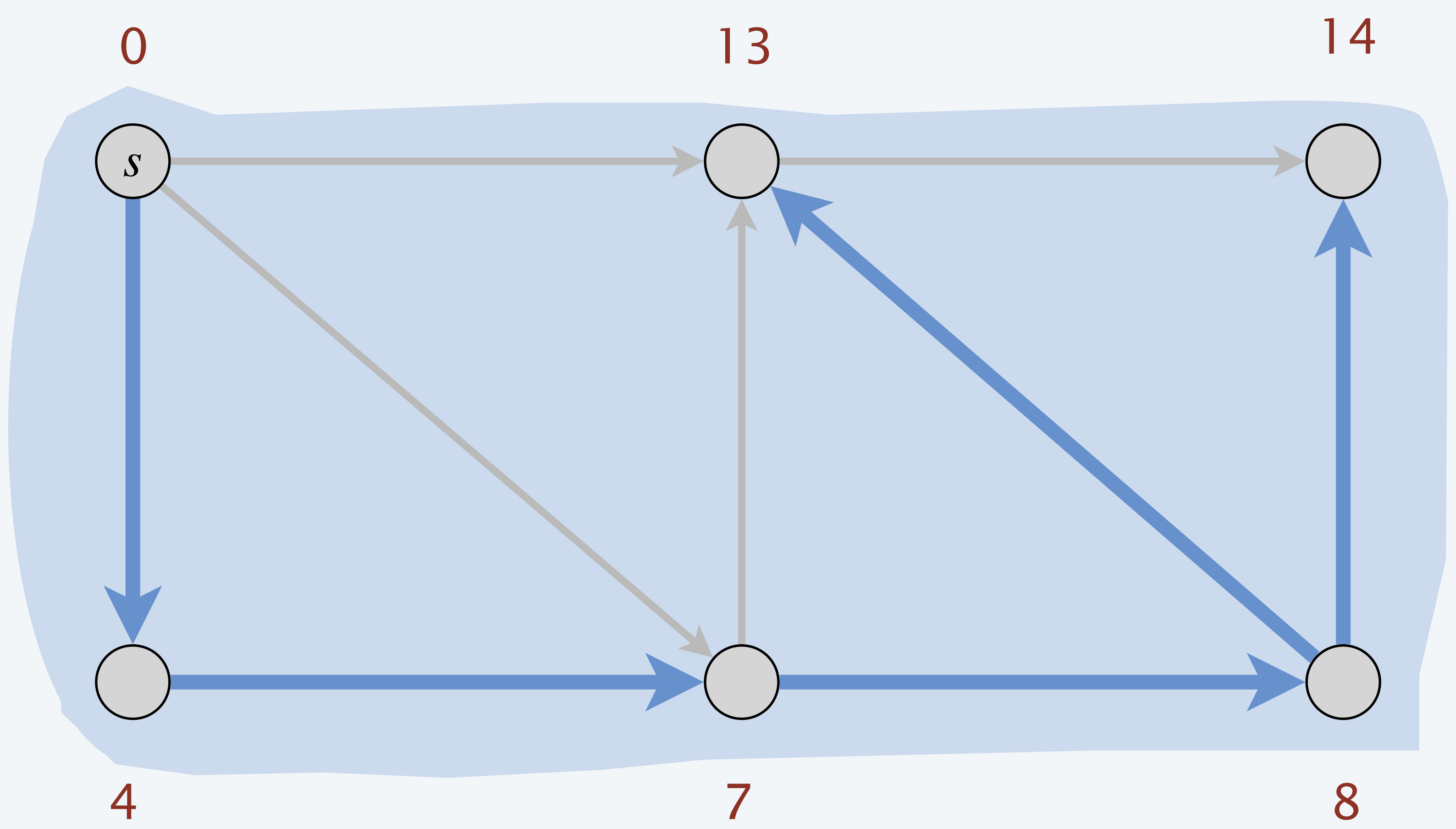
Dijkstra's algorithm demo (efficient implementation)



Dijkstra's algorithm demo (efficient implementation)



Dijkstra's algorithm demo (efficient implementation)



Dijkstra's algorithm demo (efficient implementation)