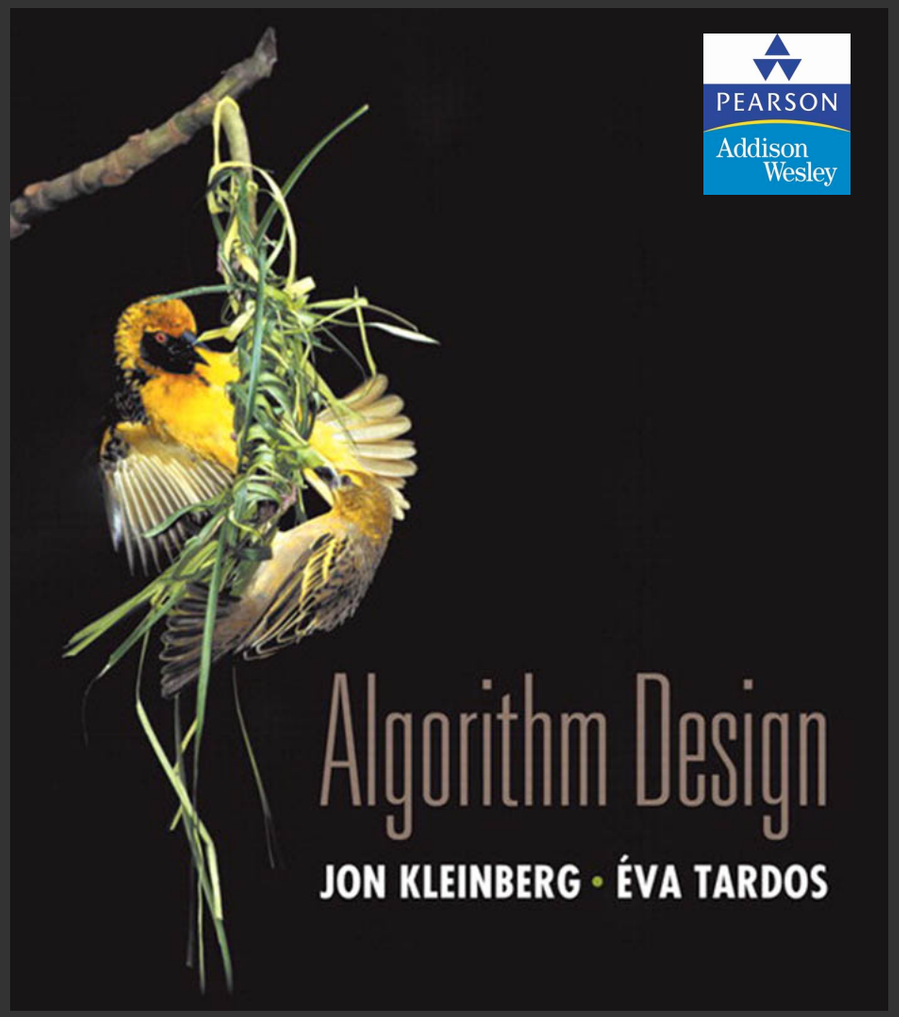


4. Greedy Algorithms III

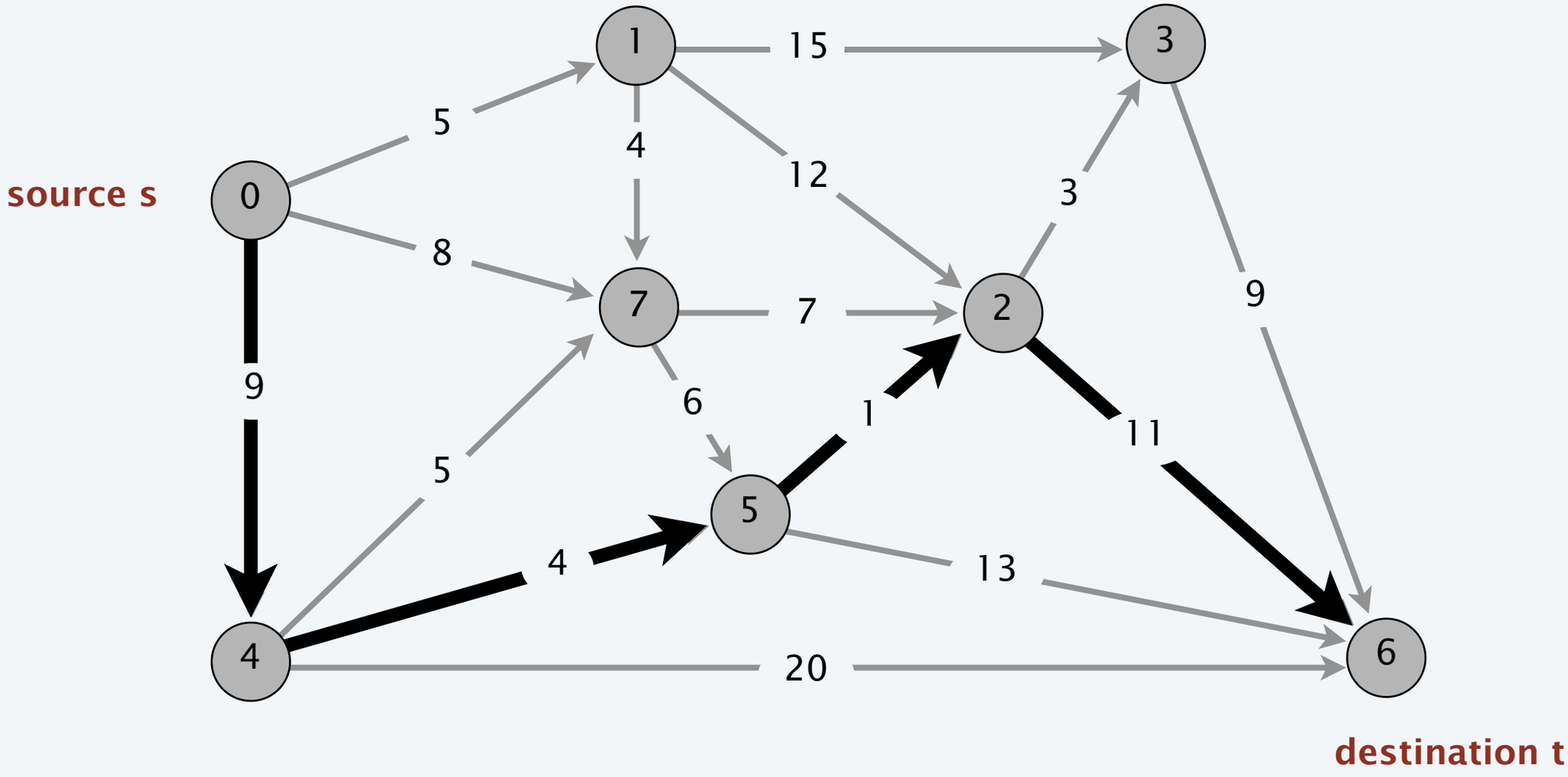
- *Dijkstra's algorithm*

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Single-pair shortest path problem

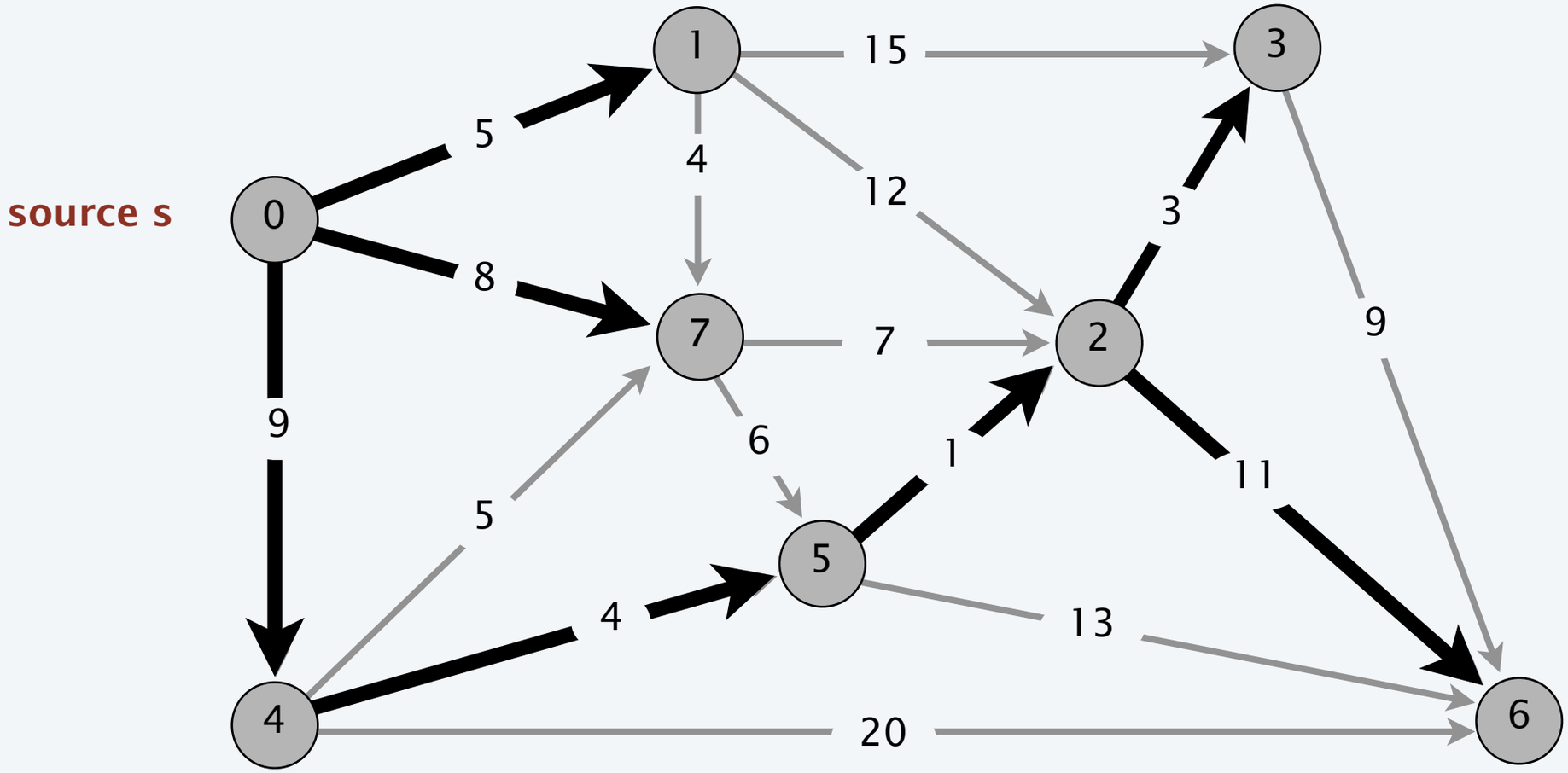
Problem. Given a digraph $G = (V, E)$, edge lengths $\ell_e \geq 0$, source $s \in V$, and destination $t \in V$, find a shortest directed path from s to t.



length of path = 9 + 4 + 1 + 11 = 25

Single-source shortest paths problem

Problem. Given a digraph $G = (V, E)$, edge lengths $\ell_e \geq 0$, source $s \in V$, find a shortest directed path from s to every node.



shortest-paths tree

Shortest paths: quiz 1

Suppose that you change the length of every edge of G as follows. For which is every shortest path in G a shortest path in G′?

A. Add 17.

B. Multiply by 17.

C. Either A or B.

D. Neither A nor B.

Shortest paths: quiz 2

Which variant in car GPS?

A. Single source: from one node s to every other node.

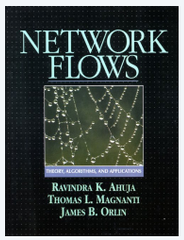
B. Single sink: from every node to one node t.

C. Source–sink: from one node s to another node t.

D. All pairs: between all pairs of nodes.

Shortest path applications

- PERT/CPM.
- Map routing.
- Seam carving.
- Robot navigation.
- Texture mapping.
- Typesetting in LaTeX.
- Urban traffic planning.
- Telemarketer operator scheduling.
- Routing of telecommunications messages.
- Network routing protocols (OSPF, BGP, RIP).
- Optimal truck routing through given traffic congestion pattern.



Network Flows: Theory, Algorithms, and Applications, by Ahuja, Magnanti, and Orlin, Prentice Hall, 1993.

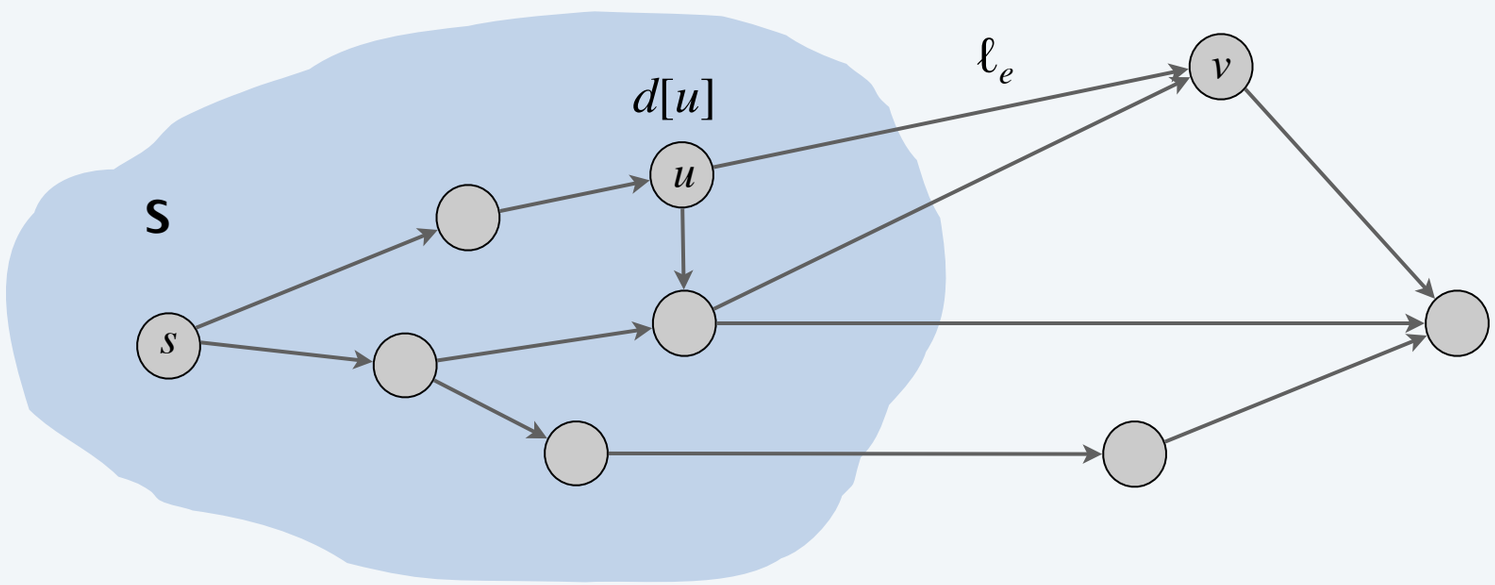
Dijkstra's algorithm (for single-source shortest paths problem)

Greedy approach. Maintain a set of explored nodes S for which algorithm has determined $d[u]$ = length of a shortest $s \rightsquigarrow u$ path.

- Initialize $S \leftarrow \{ s \}$, $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$



Dijkstra's algorithm (for single-source shortest paths problem)

Greedy approach. Maintain a set of explored nodes S for which algorithm has determined $d[u]$ = length of a shortest $s \rightsquigarrow u$ path.

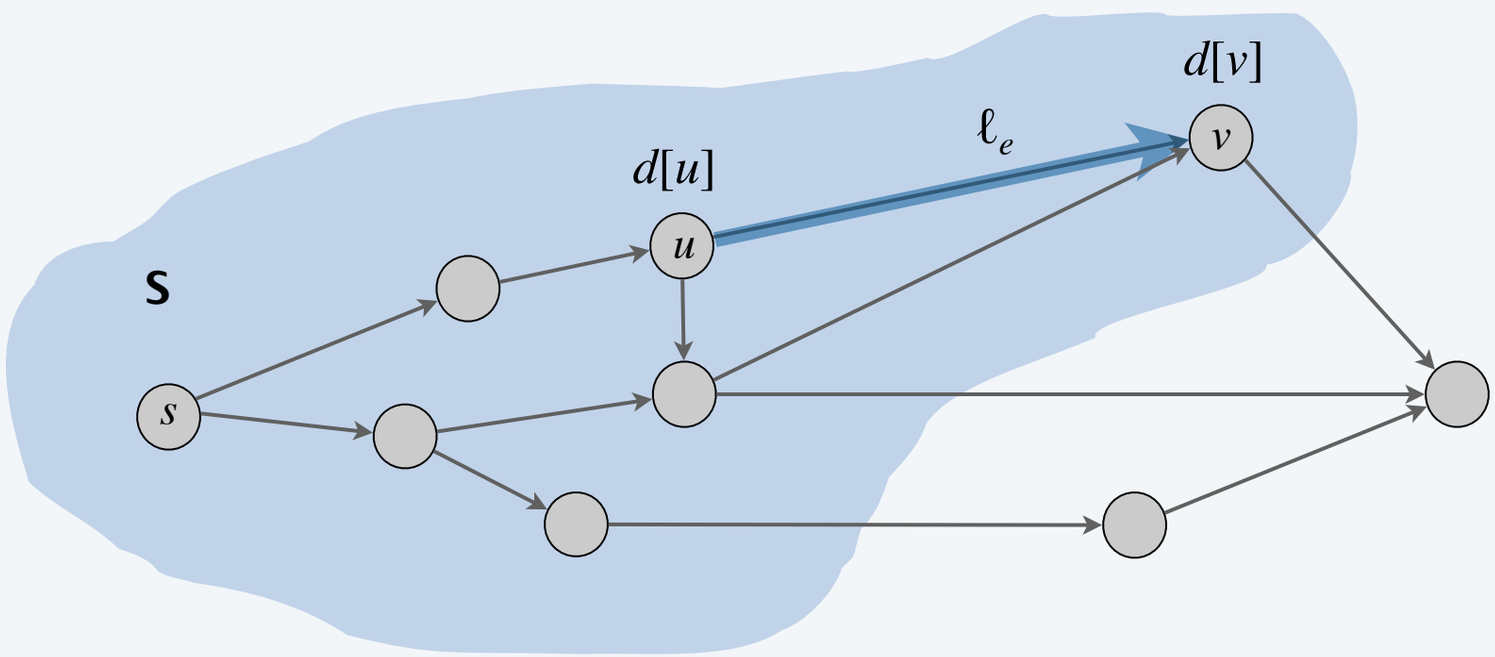
- Initialize $S \leftarrow \{ s \}$, $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

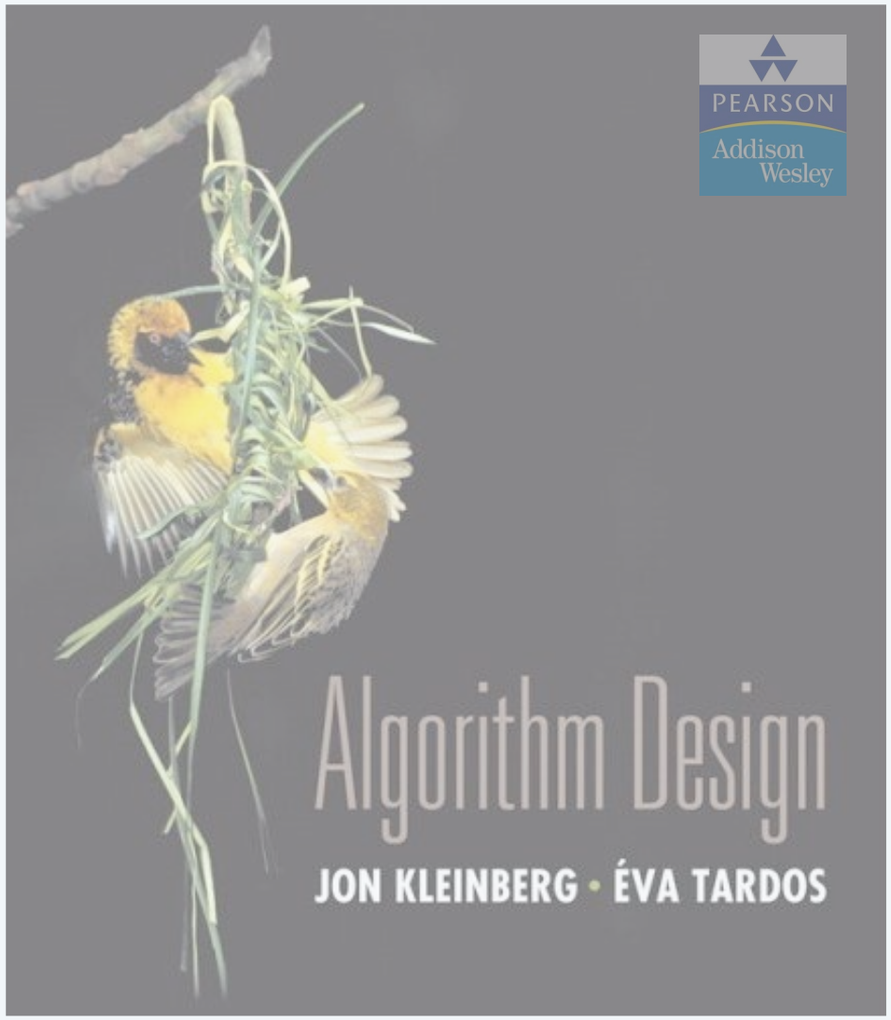
$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S, and set $d[v] \leftarrow \pi(v)$.

- To recover path, set $pred[v] \leftarrow e$ that achieves min.





4. Greedy Algorithms III

- *Dijkstra's algorithm demo*

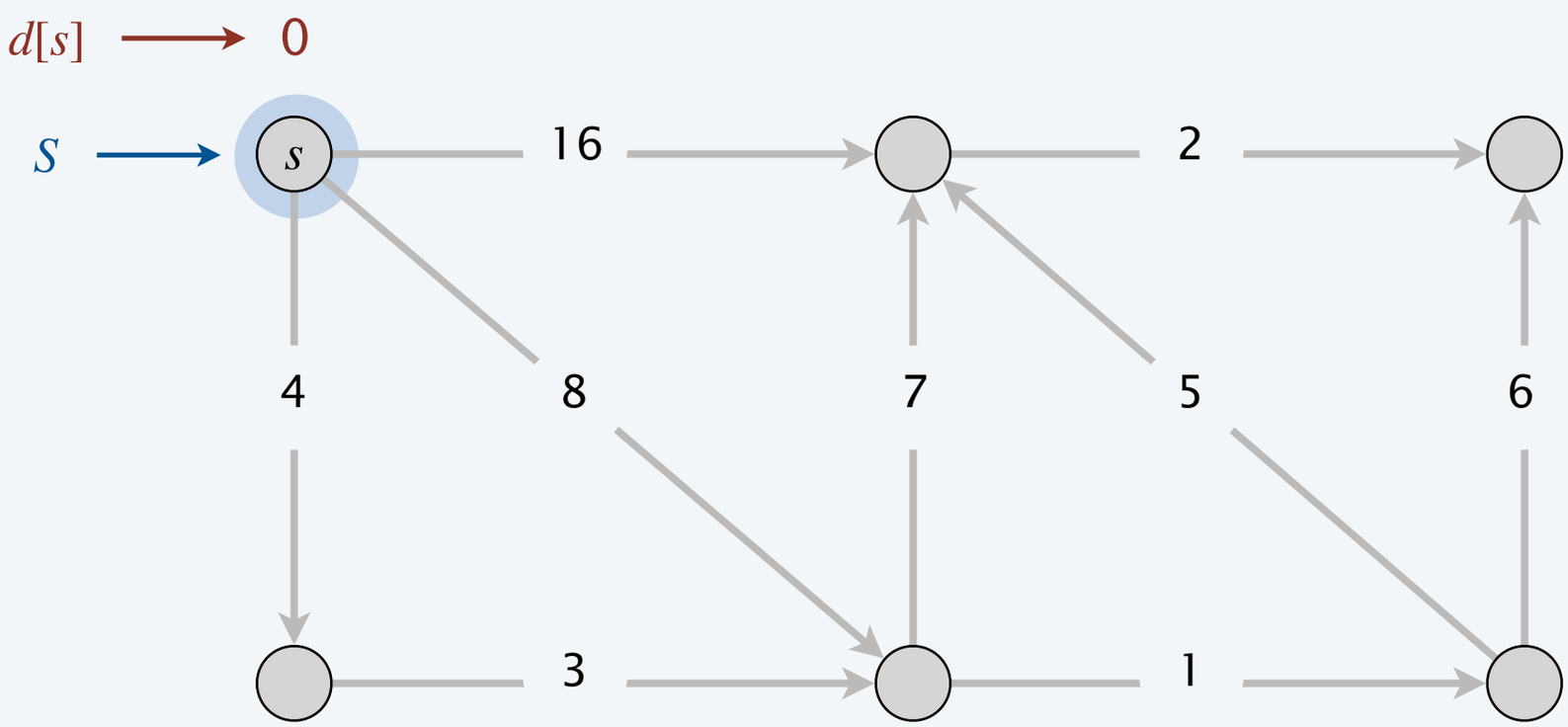
Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.



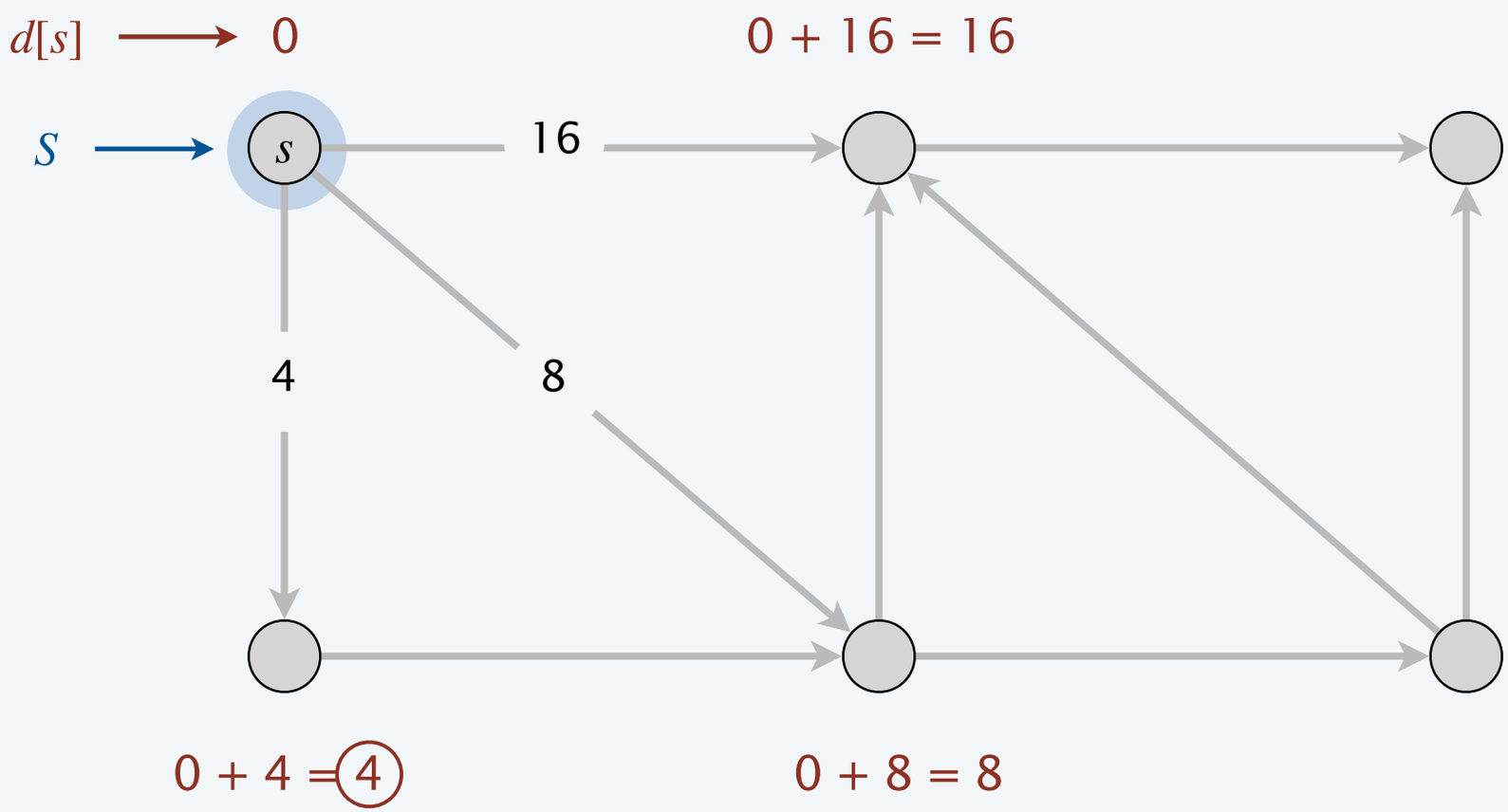
Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.



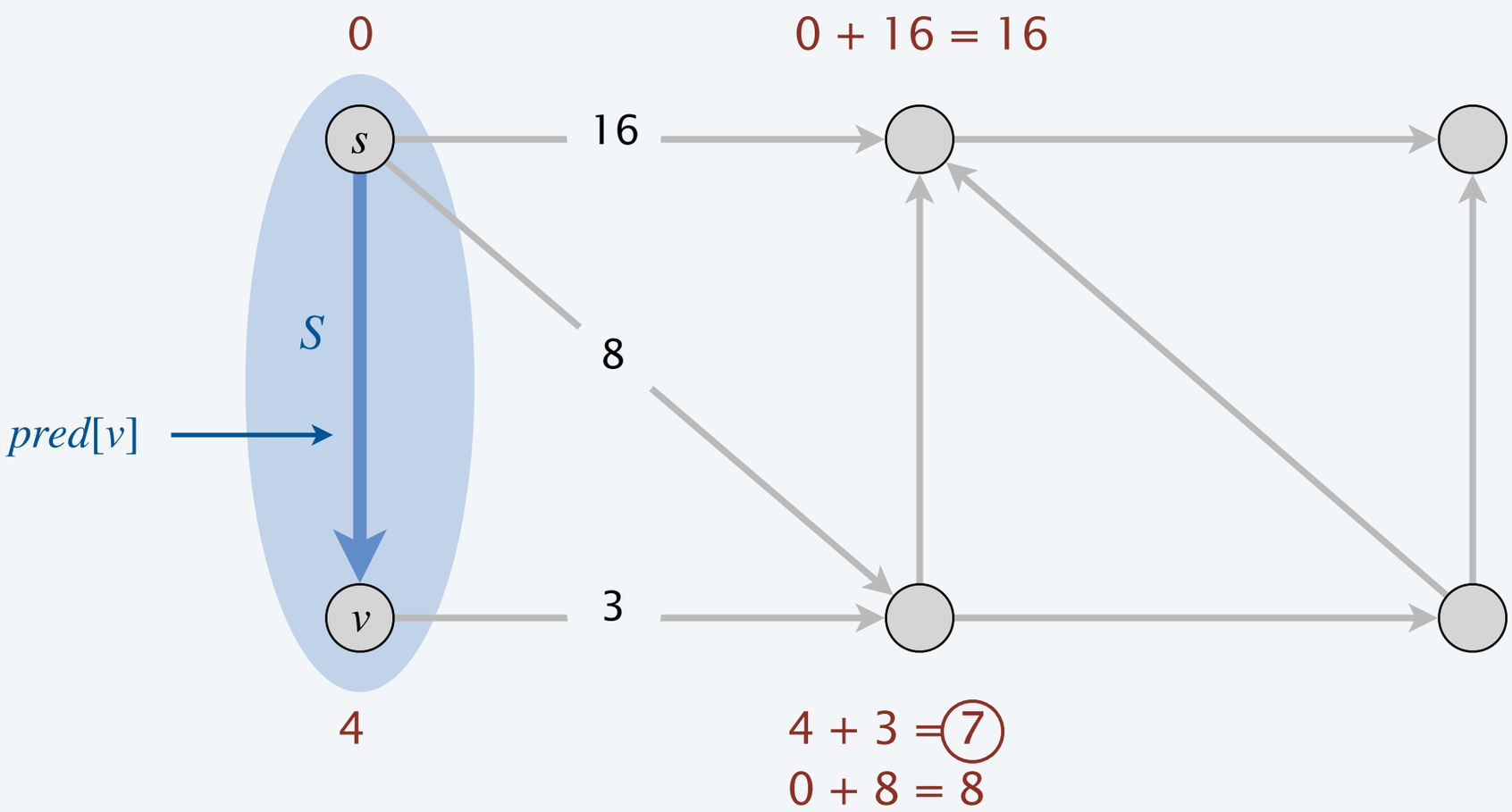
Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} \; d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.



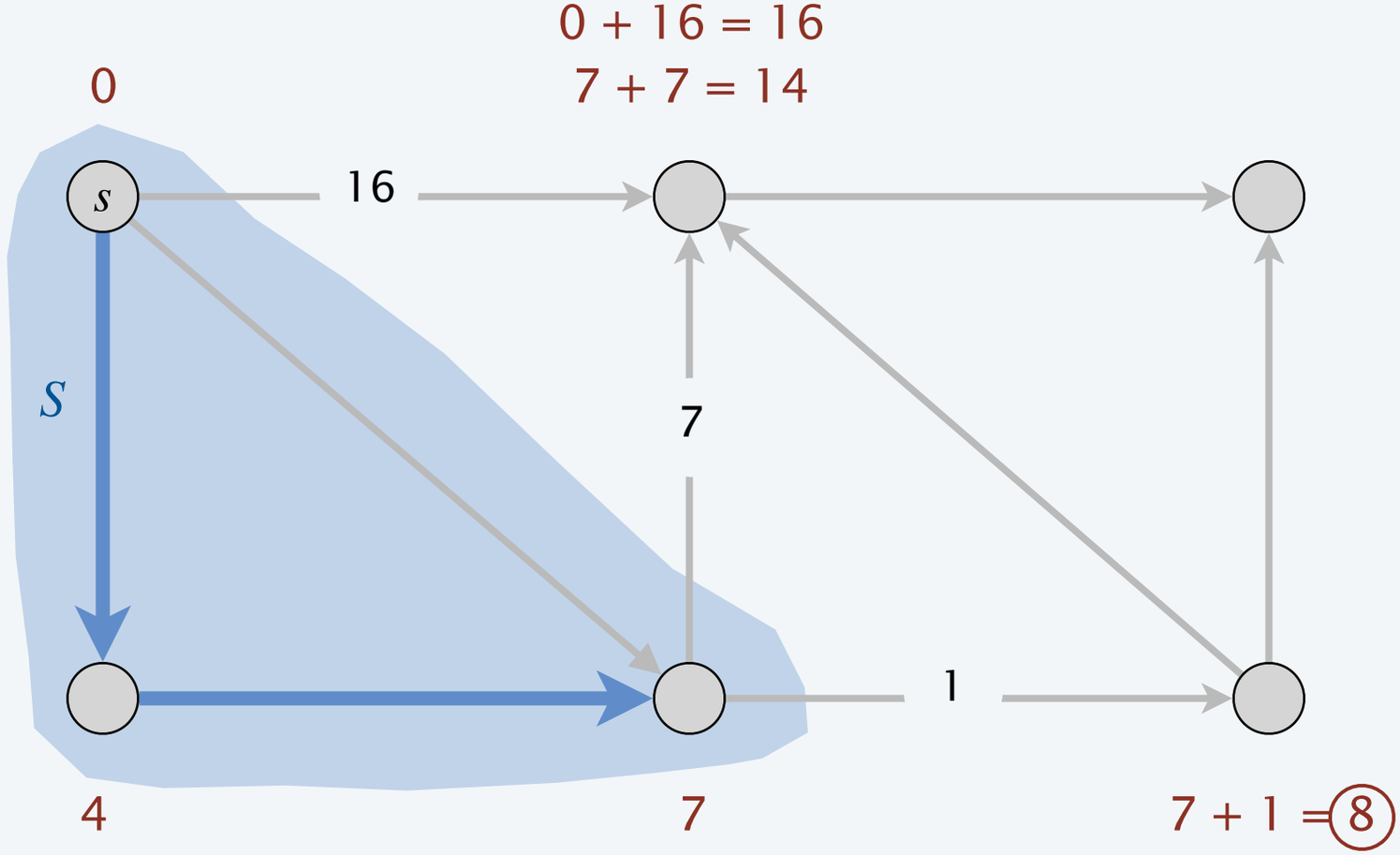
Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.



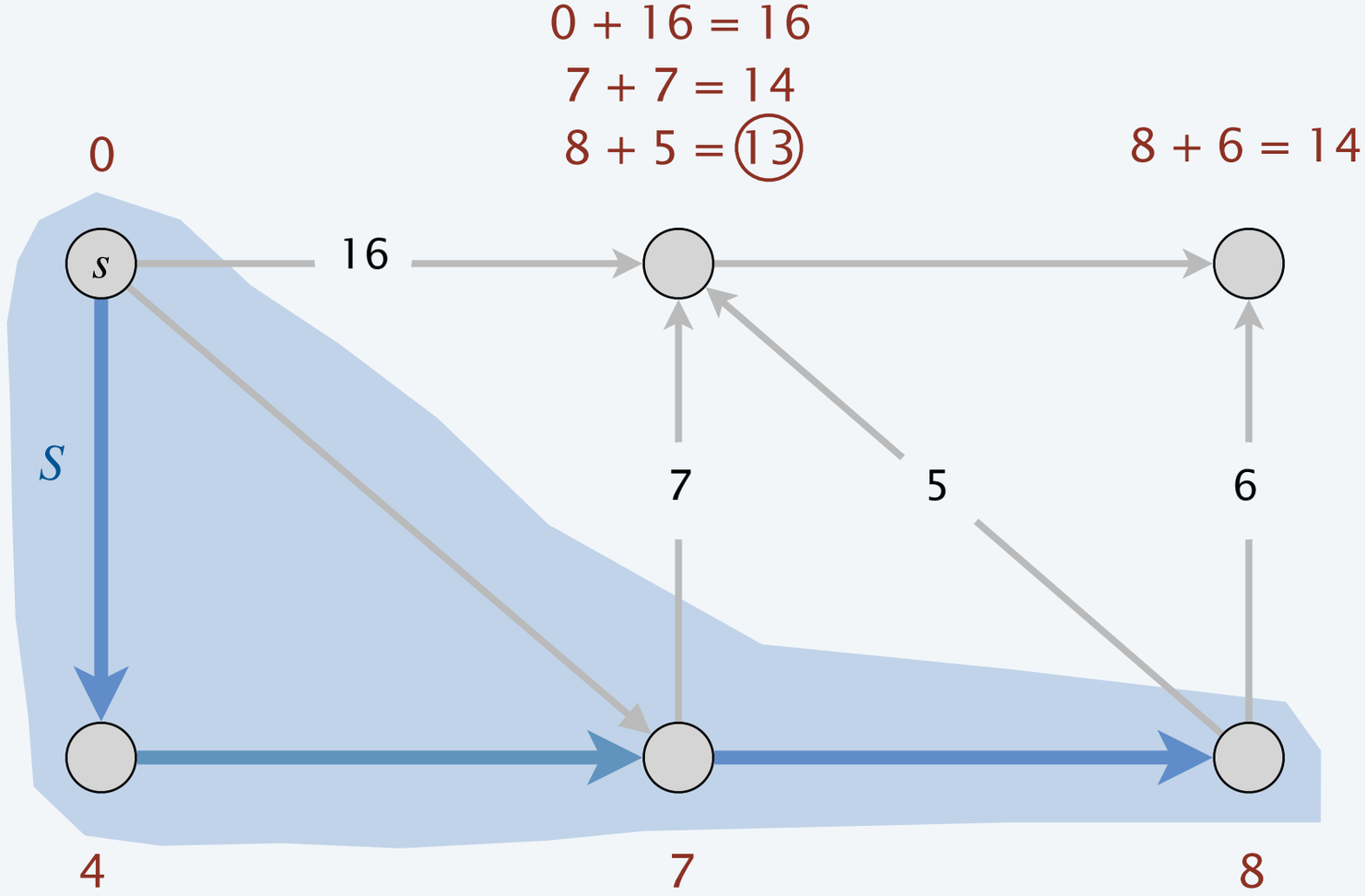
Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.



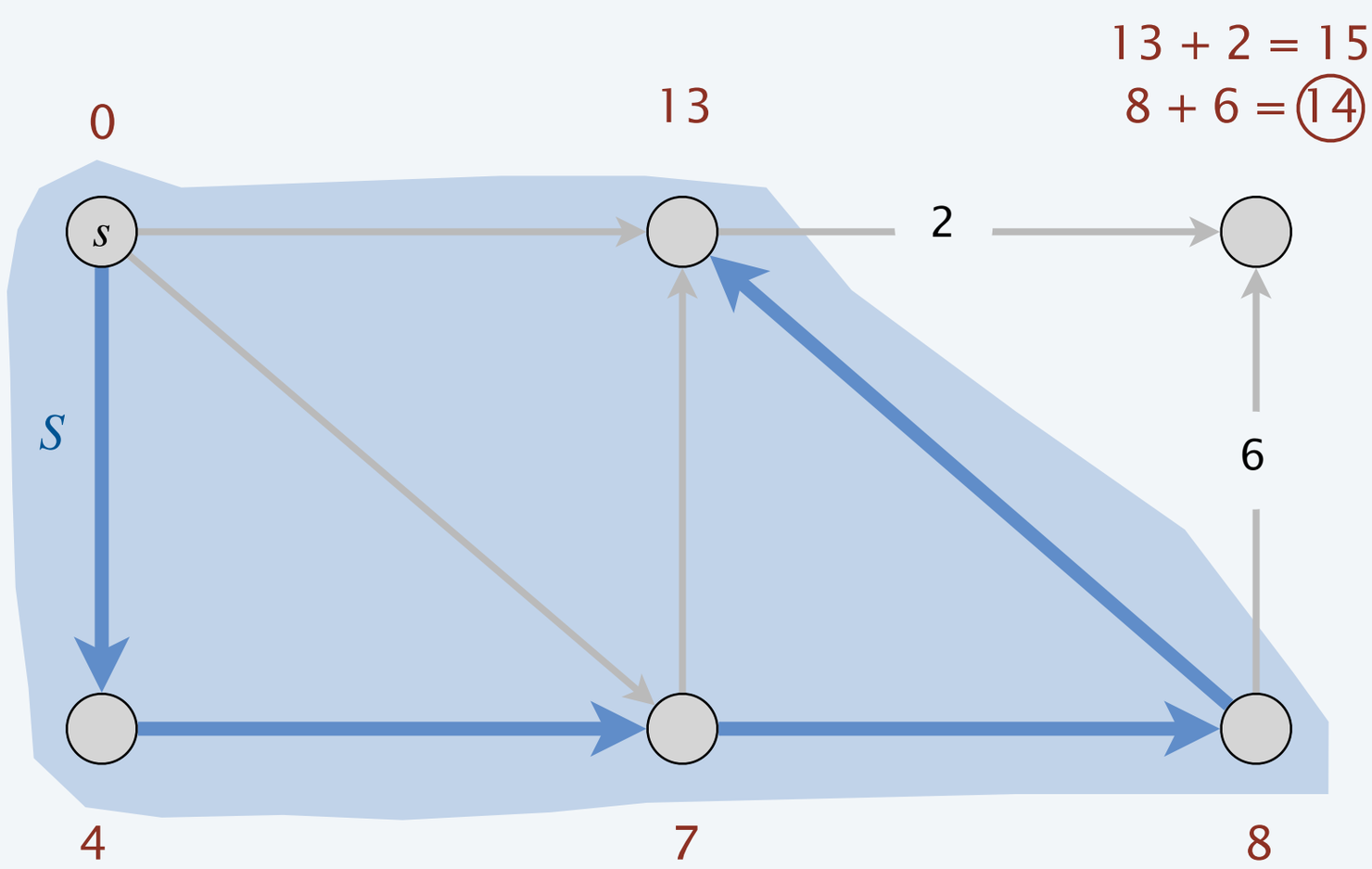
Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.

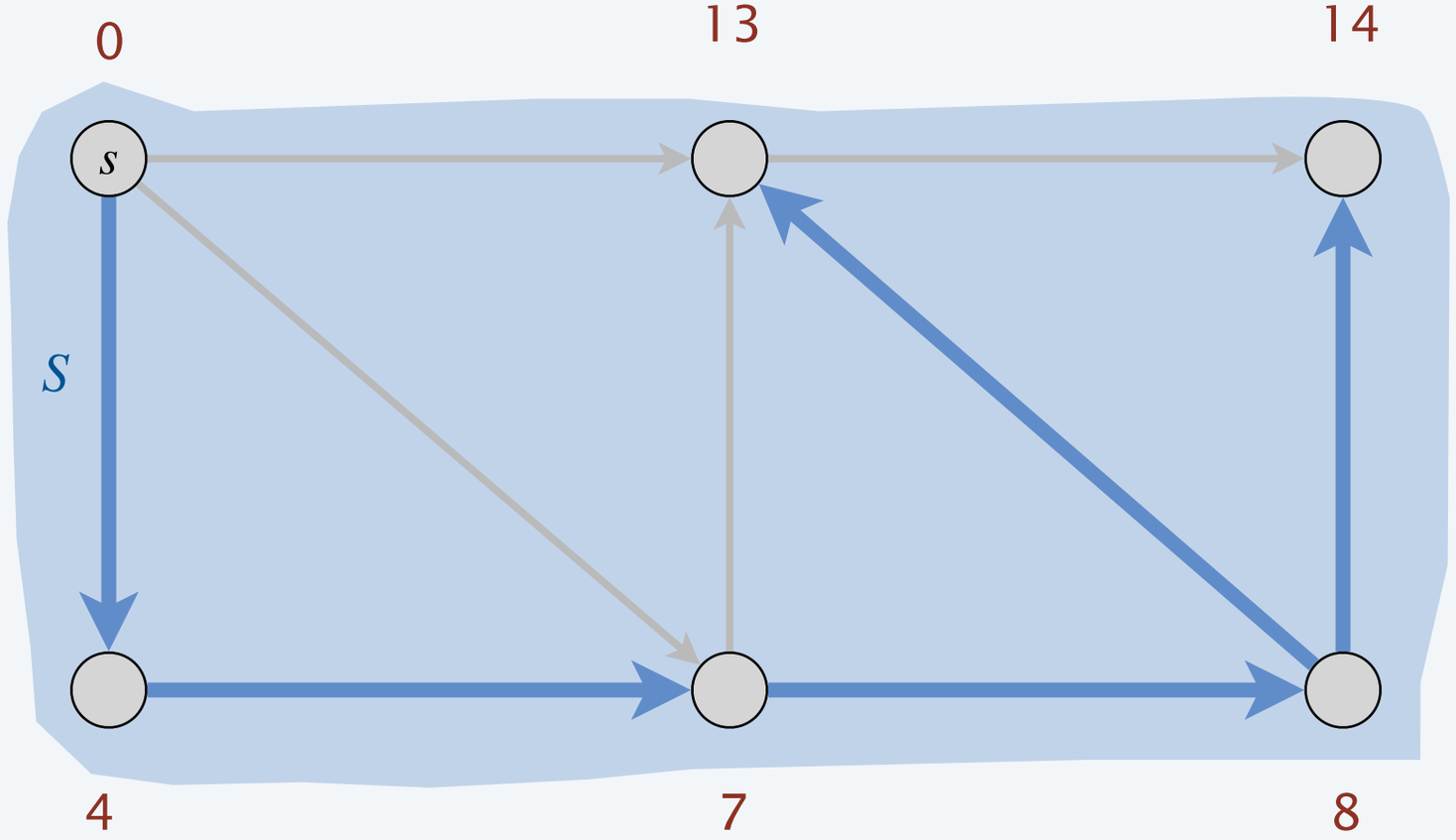


Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e\,=\,(u,v)\,:\,u \in S} d[u] + \ell_e$$

  add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.

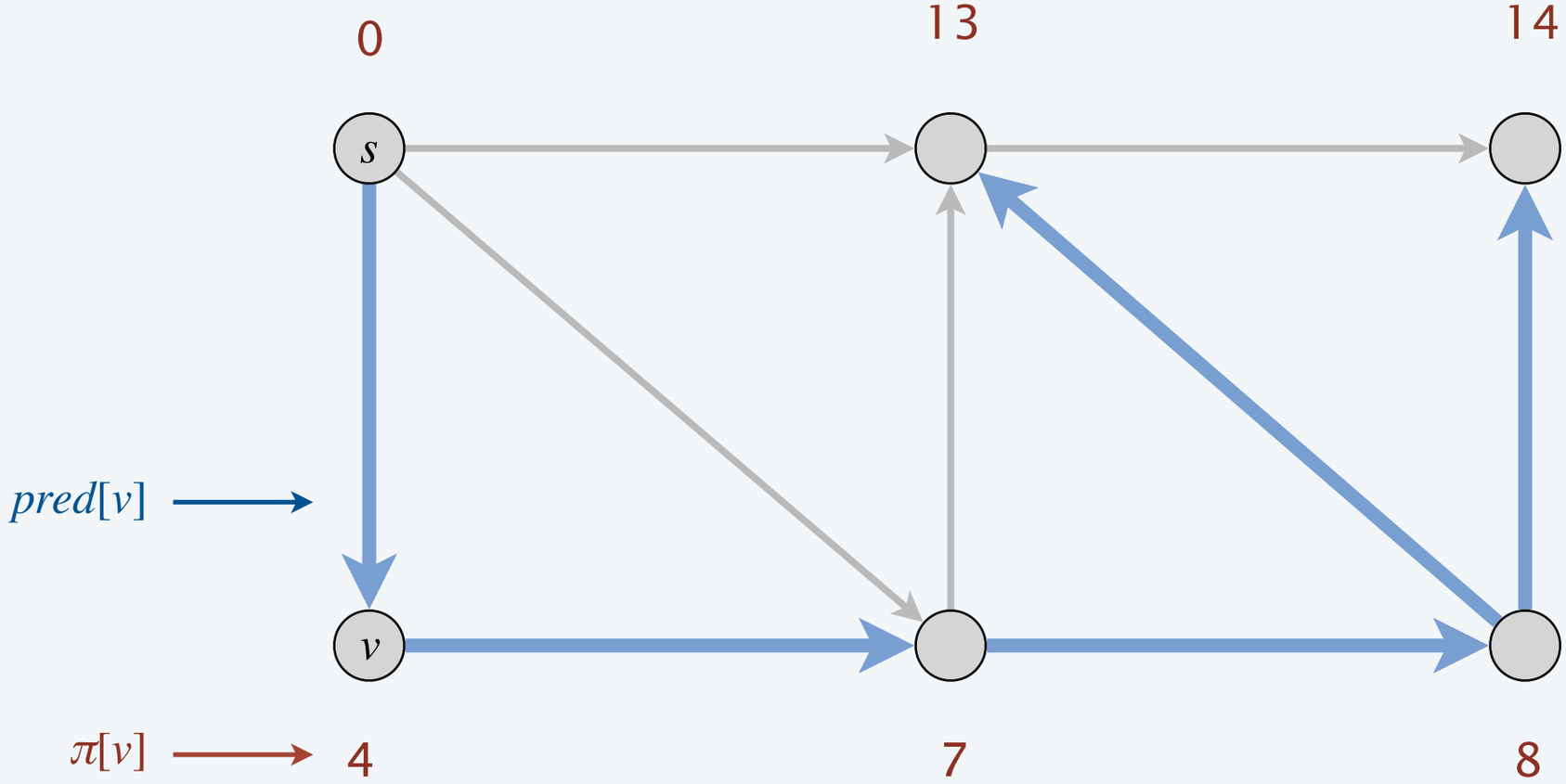


Dijkstra's algorithm demo

- Initialize $S \leftarrow \{ s \}$ and $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e = (u,v) \,:\, u \in S} \; d[u] + \ell_e$$

  add v to S; set $d[v] \leftarrow \pi(v)$ and $pred[v] \leftarrow$ argmin.



Dijkstra's algorithm: proof of correctness

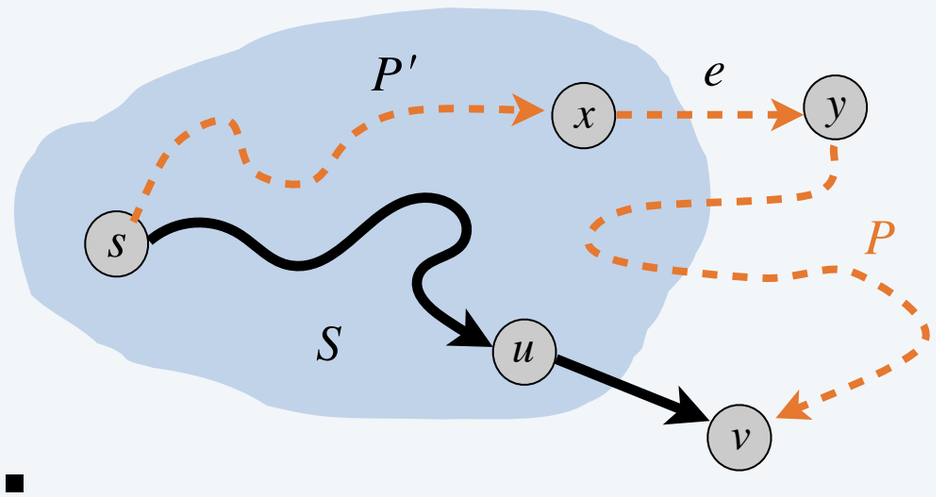
Invariant. For each node $u \in S$: $d[u]$ = length of a shortest $s \rightsquigarrow u$ path.

Pf. [by induction on $|S|$]

Base case: $|S| = 1$ is easy since $S = \{ s \}$ and $d[s] = 0$.

Inductive hypothesis: Assume true for $|S| \geq 1$.

- Let v be next node added to S, and let (u, v) be the final edge.
- A shortest $s \rightsquigarrow u$ path plus (u, v) is an $s \rightsquigarrow v$ path of length $\pi(v)$.
- Consider any other $s \rightsquigarrow v$ path P. We show that it is no shorter than $\pi(v)$.
- Let $e = (x, y)$ be the first edge in P that leaves S, and let P' be the subpath from s to x.
- The length of P is already $\geq \pi(v)$ as soon as it reaches y:



$$\ell(P) \geq \ell(P') + \ell_e \geq d[x] + \ell_e \geq \pi(y) \geq \pi(v) \quad \blacksquare$$

- $\ell(P) \geq \ell(P') + \ell_e$: non-negative lengths
- $\ell(P') + \ell_e \geq d[x] + \ell_e$: inductive hypothesis
- $d[x] + \ell_e \geq \pi(y)$: definition of $\pi(y)$
- $\pi(y) \geq \pi(v)$: Dijkstra chose v instead of y

Dijkstra's algorithm: efficient implementation

Critical optimization 1. For each unexplored node $v \notin S$:
explicitly maintain $\pi[v]$ instead of computing directly from definition

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

- For each $v \notin S$: $\pi(v)$ can only decrease (because set S increases).

- More specifically, suppose u is added to S and there is an edge $e = (u, v)$ leaving u. Then, it suffices to update:

$$\pi[v] \leftarrow \min \{\, \pi[v],\; \pi[u] + \ell_e) \,\}$$

recall: for each $u \in S$,
$\pi[u] = d[u]$ = length of shortest $s \rightsquigarrow u$ path

Critical optimization 2. Use a min-oriented priority queue (PQ)
to choose an unexplored node that minimizes $\pi[v]$.

Dijkstra's algorithm: efficient implementation

Implementation.

- Algorithm maintains $\pi[v]$ for each node v.
- Priority queue stores unexplored nodes, using $\pi[\cdot]$ as priorities.
- Once u is deleted from the PQ, $\pi[u]$ = length of a shortest $s \rightsquigarrow u$ path.

DIJKSTRA (V, E, ℓ, s)

FOREACH $v \neq s$: $\pi[v] \leftarrow \infty, pred[v] \leftarrow null;\ \pi[s] \leftarrow 0.$

Create an empty priority queue pq.

FOREACH $v \in V$: INSERT$(pq, v, \pi[v])$.

WHILE (IS-NOT-EMPTY(pq))

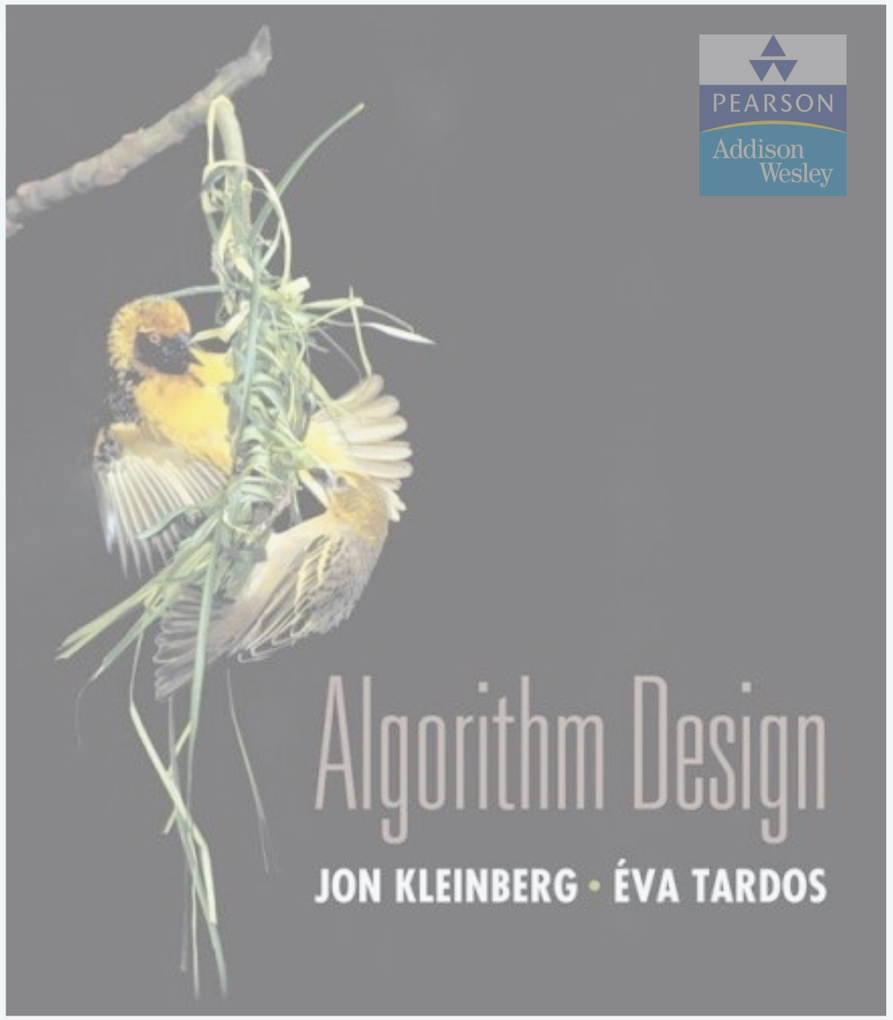
  $u \leftarrow$ DEL-MIN(pq).

  FOREACH edge $e = (u, v) \in E$ leaving u:

    IF $(\pi[v] > \pi[u] + \ell_e)$

      DECREASE-KEY$(pq,\ v,\ \pi[u] + \ell_e)$.

      $\pi[v] \leftarrow \pi[u] + \ell_e\ ;\ pred[v] \leftarrow e.$



4. Greedy Algorithms III

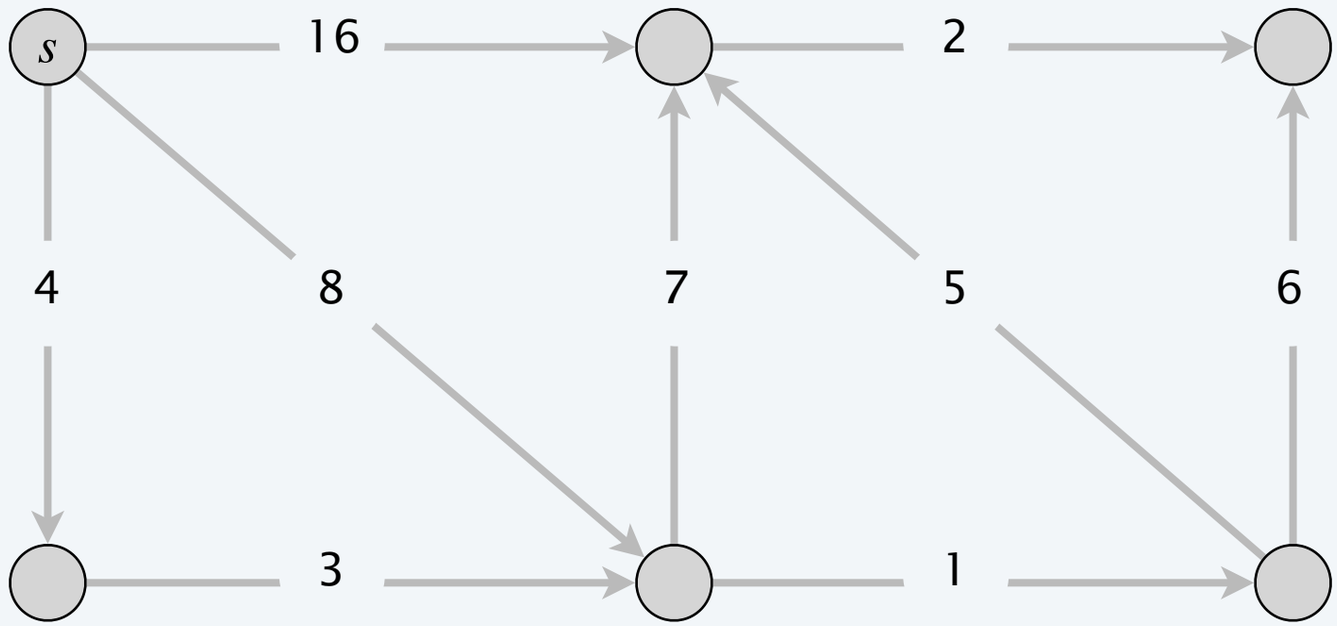
- *Dijkstra's algorithm demo (efficient implementation)*

Dijkstra's algorithm demo (efficient implementation)

Initialization.

- For all $v \neq s$: $\pi[v] \leftarrow \infty$.
- For all $v \neq s$: $pred[v] \leftarrow null$.
- $S \leftarrow \emptyset$ and $\pi[s] \leftarrow 0$.

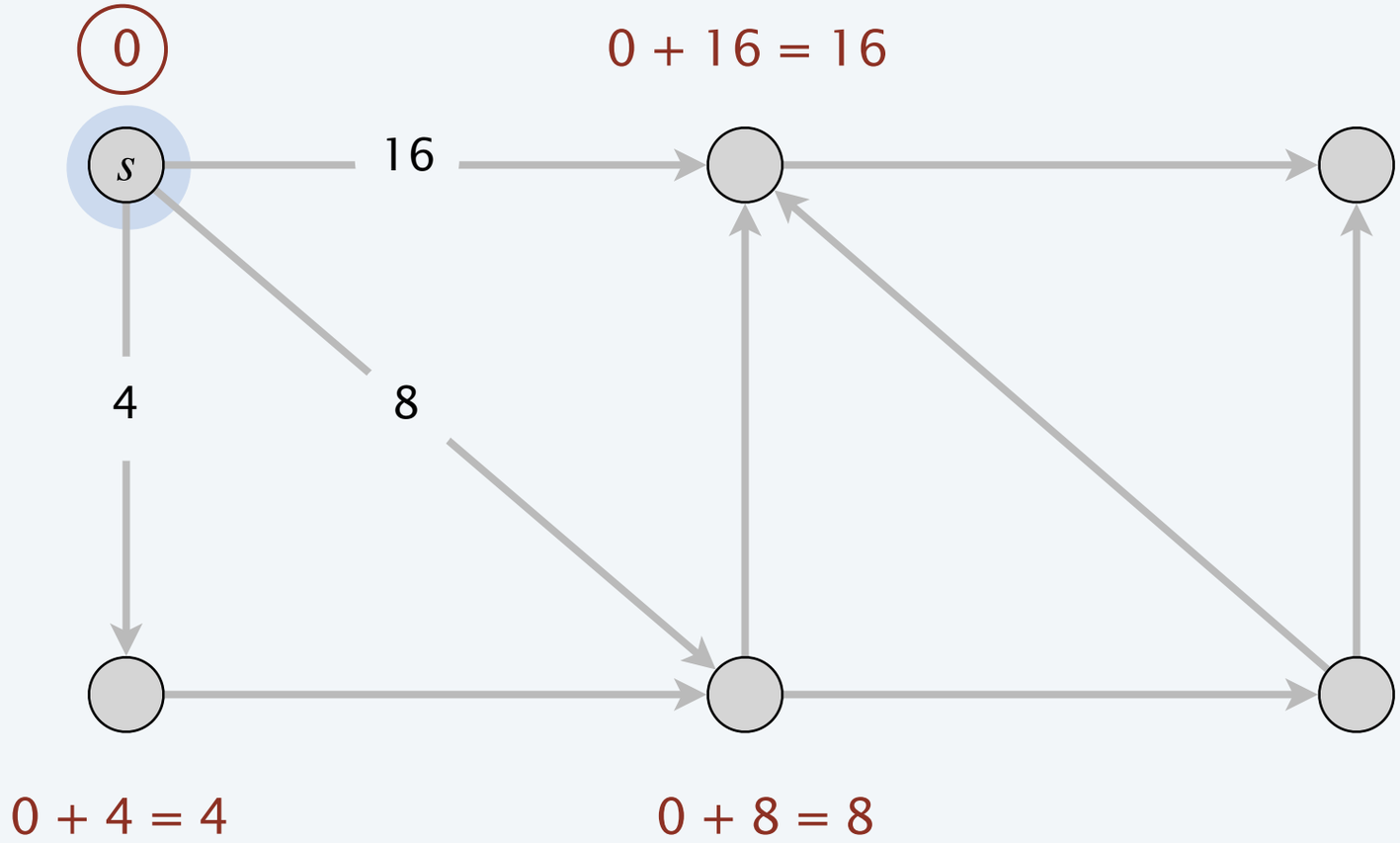
$\pi[s] \longrightarrow 0$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

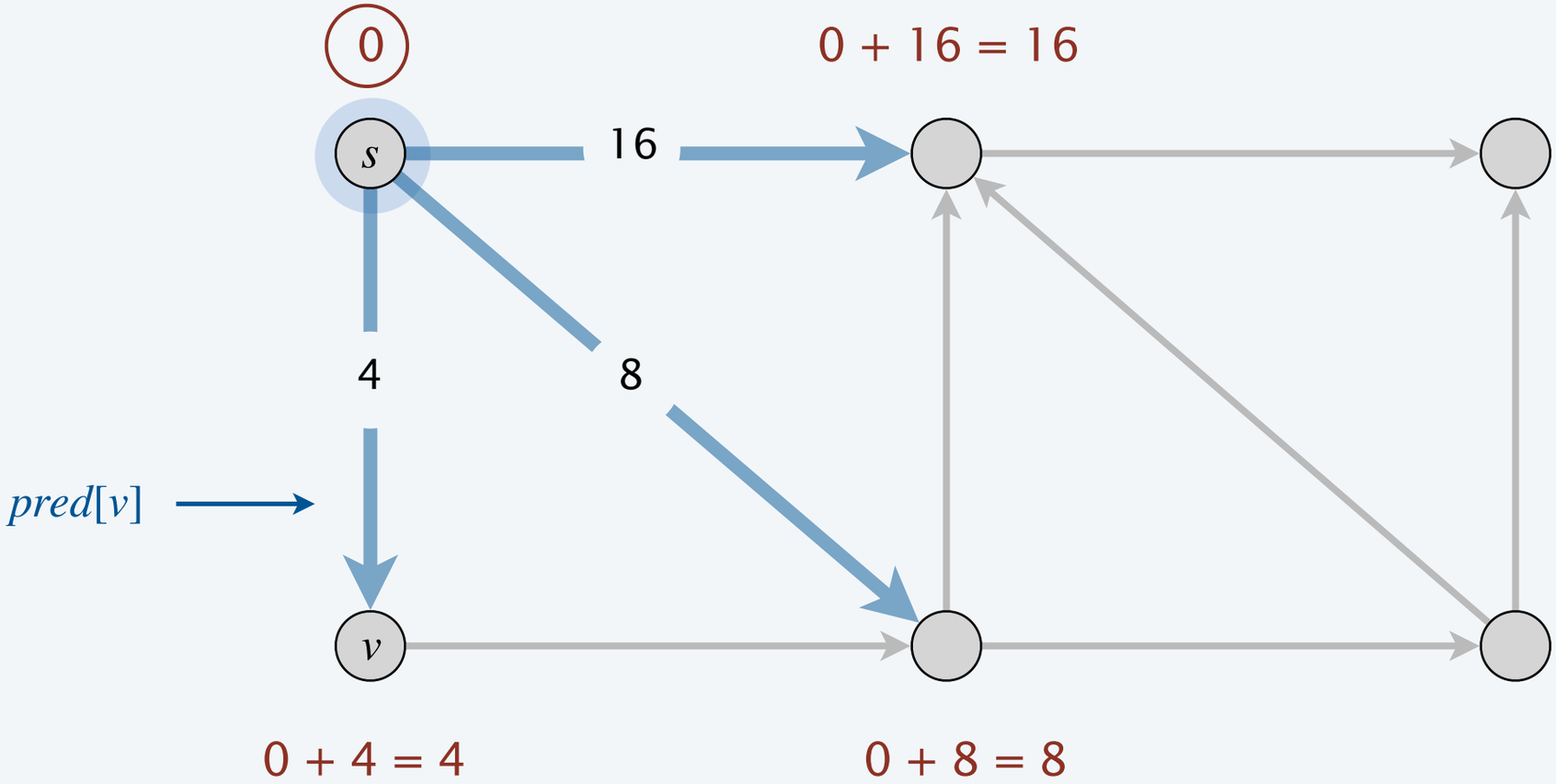
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

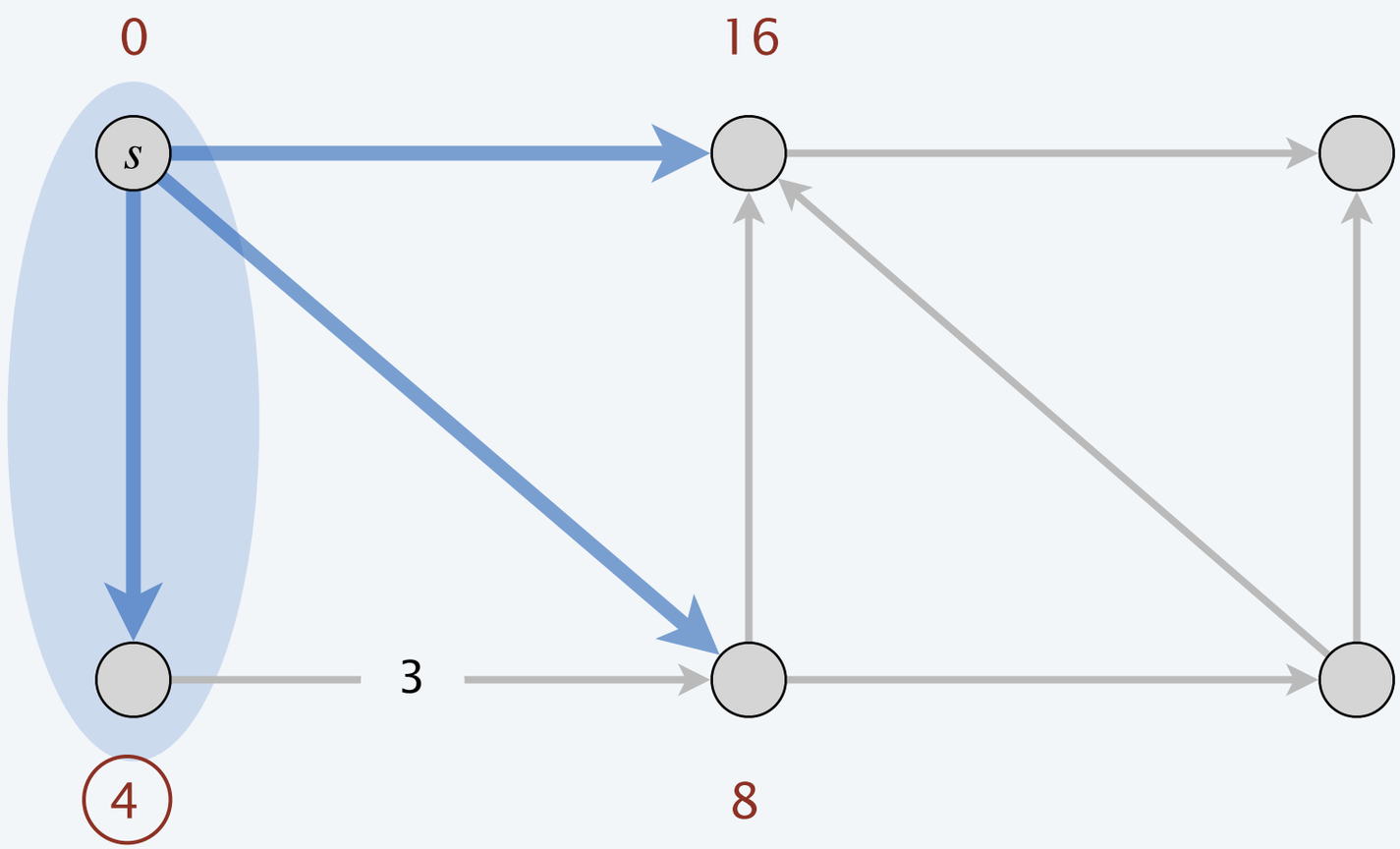
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

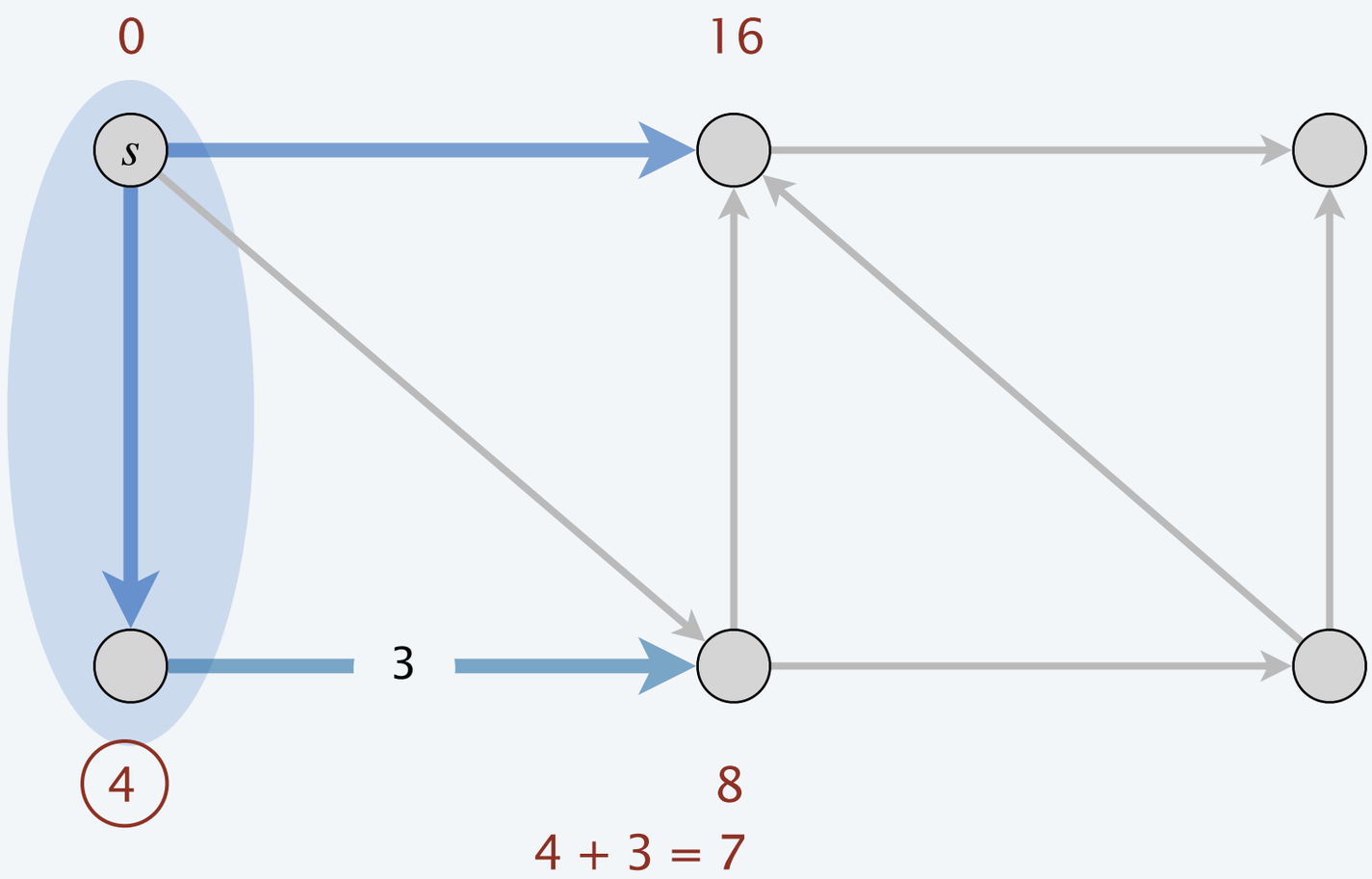
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

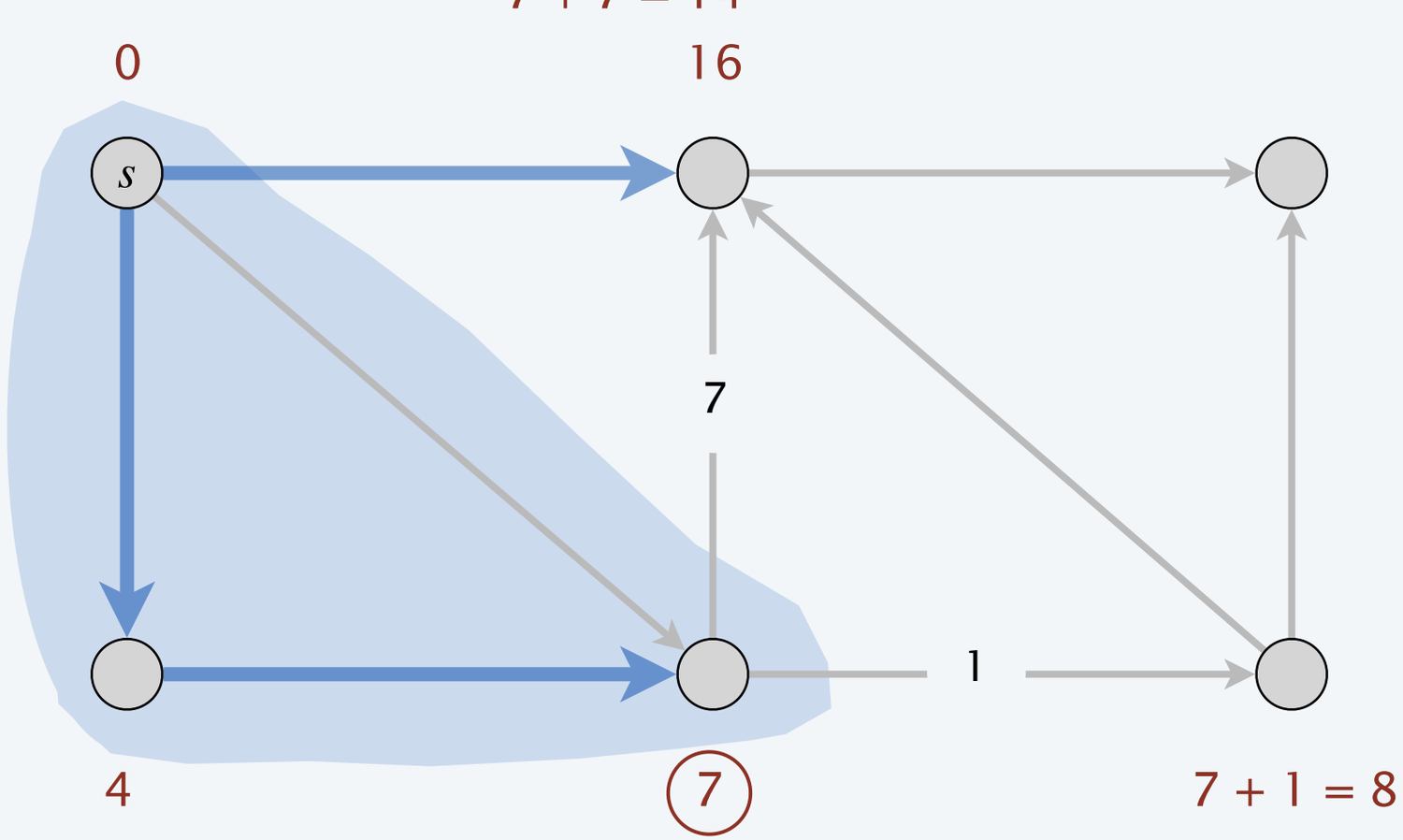
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

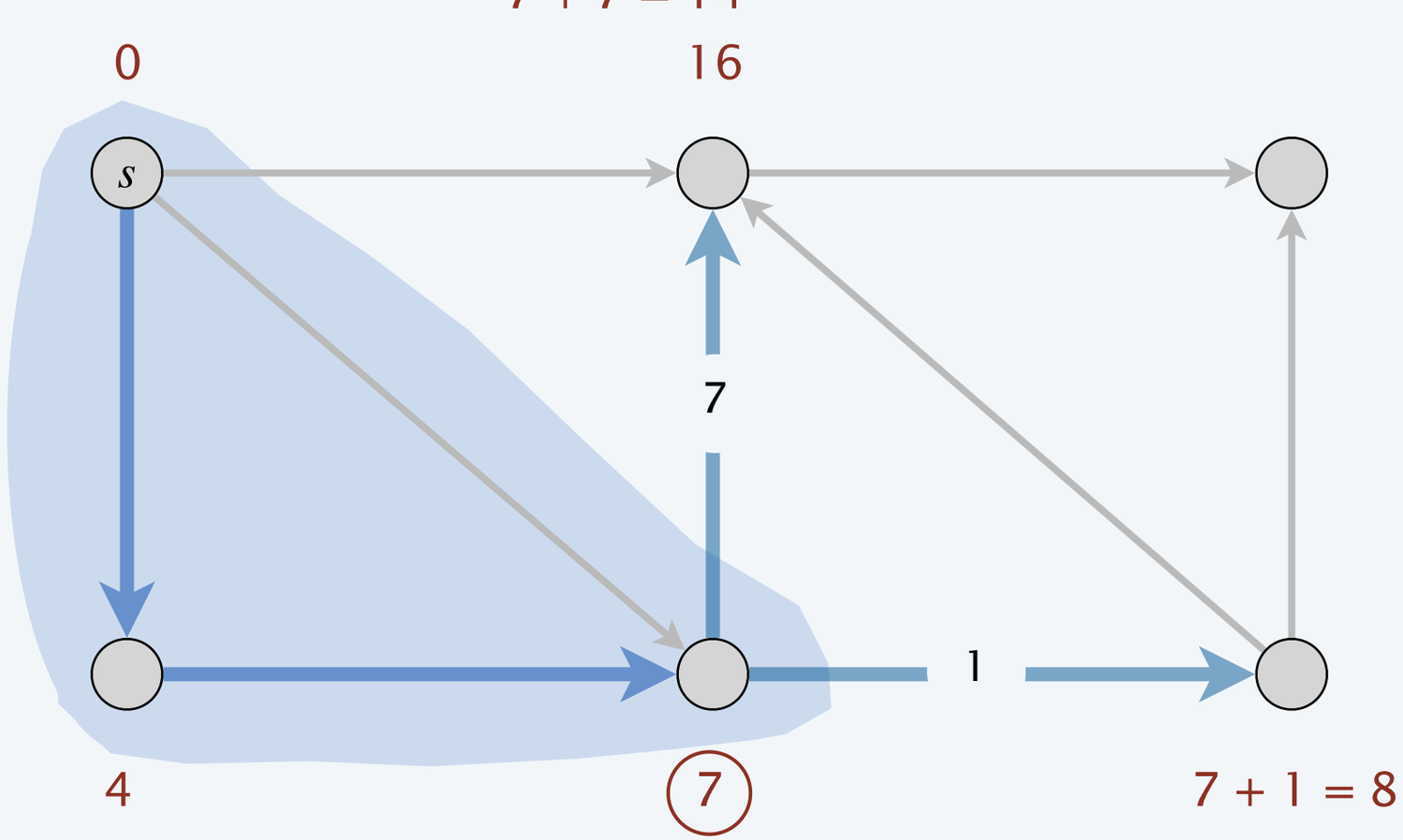
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

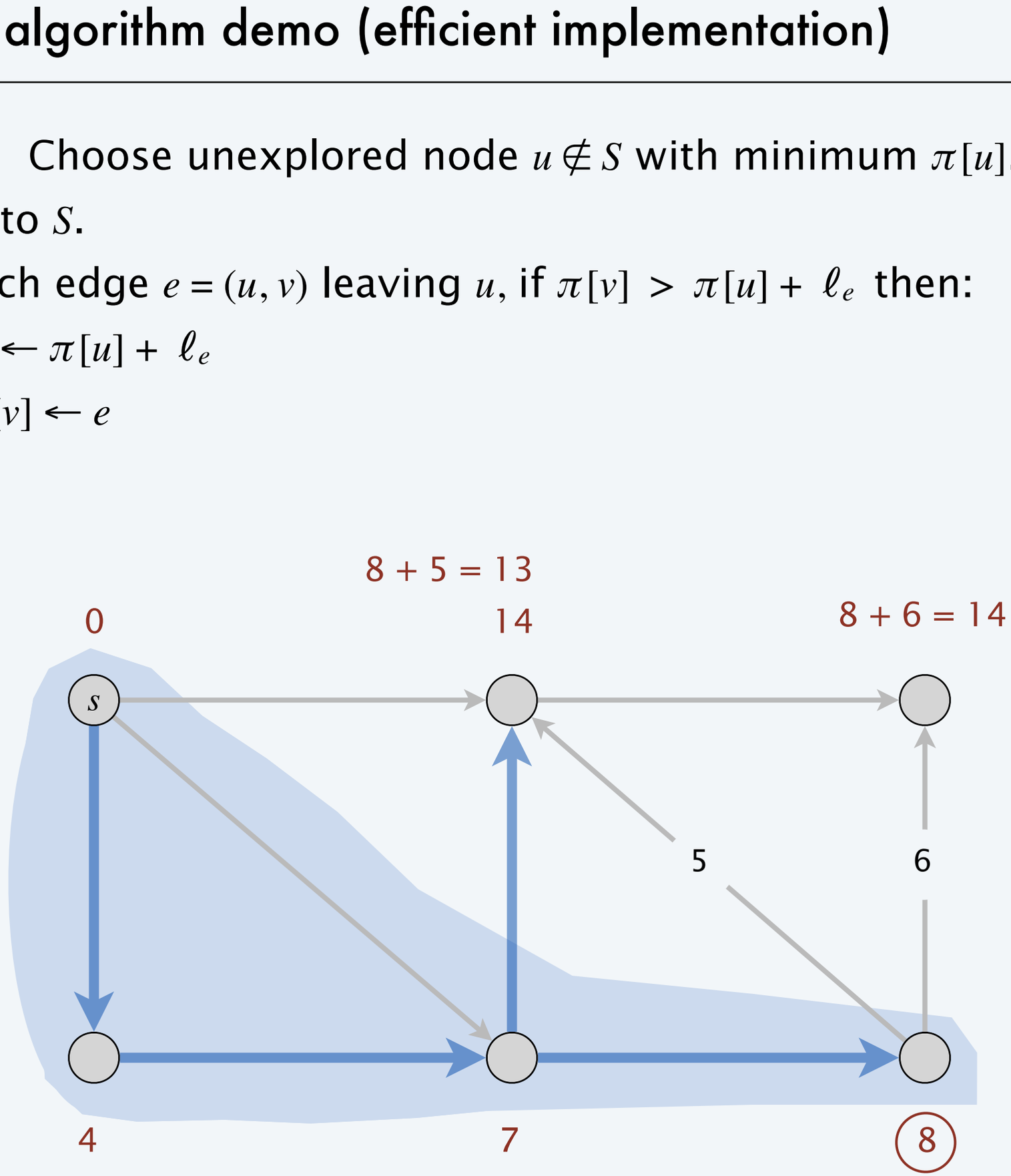
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

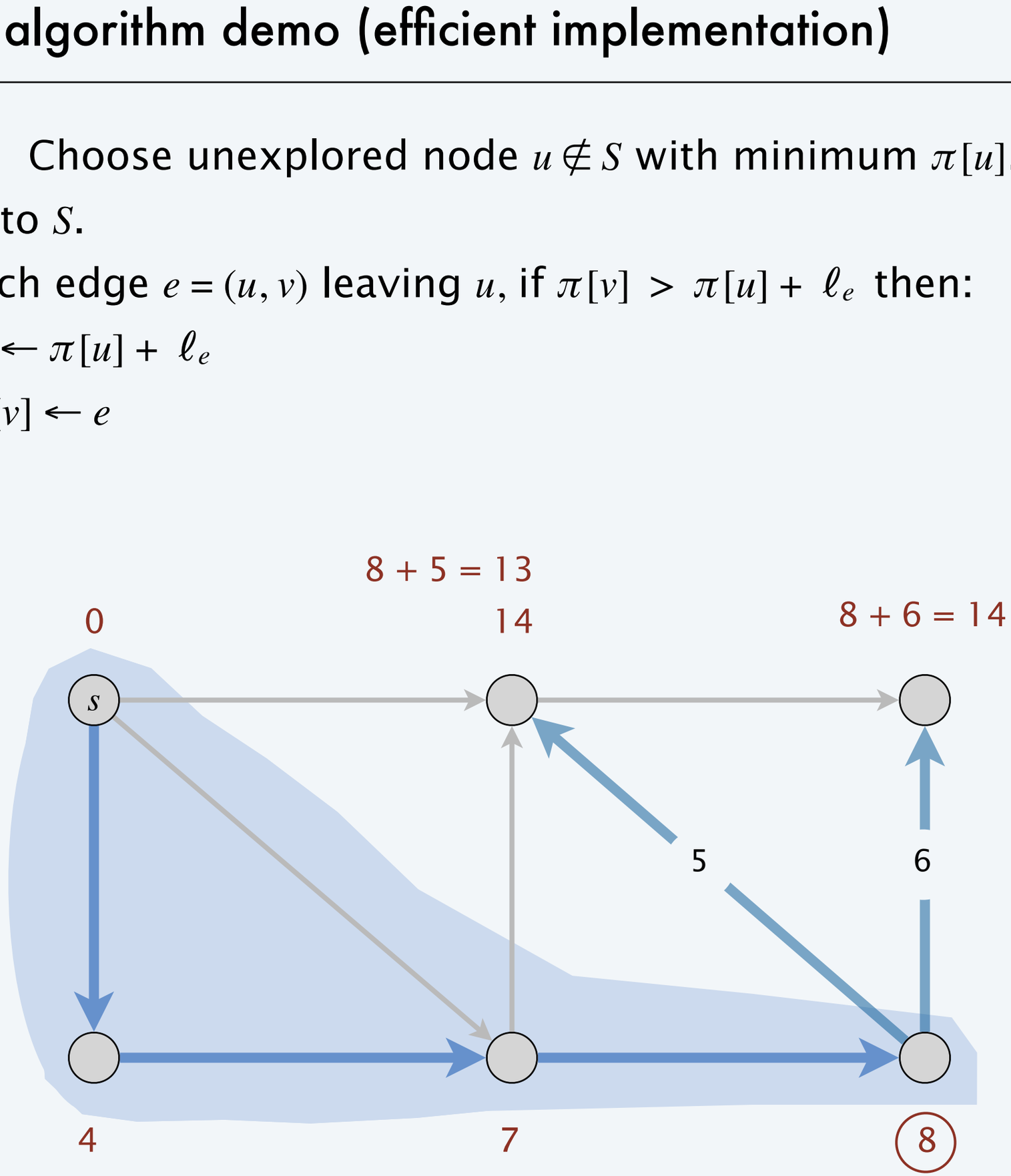
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

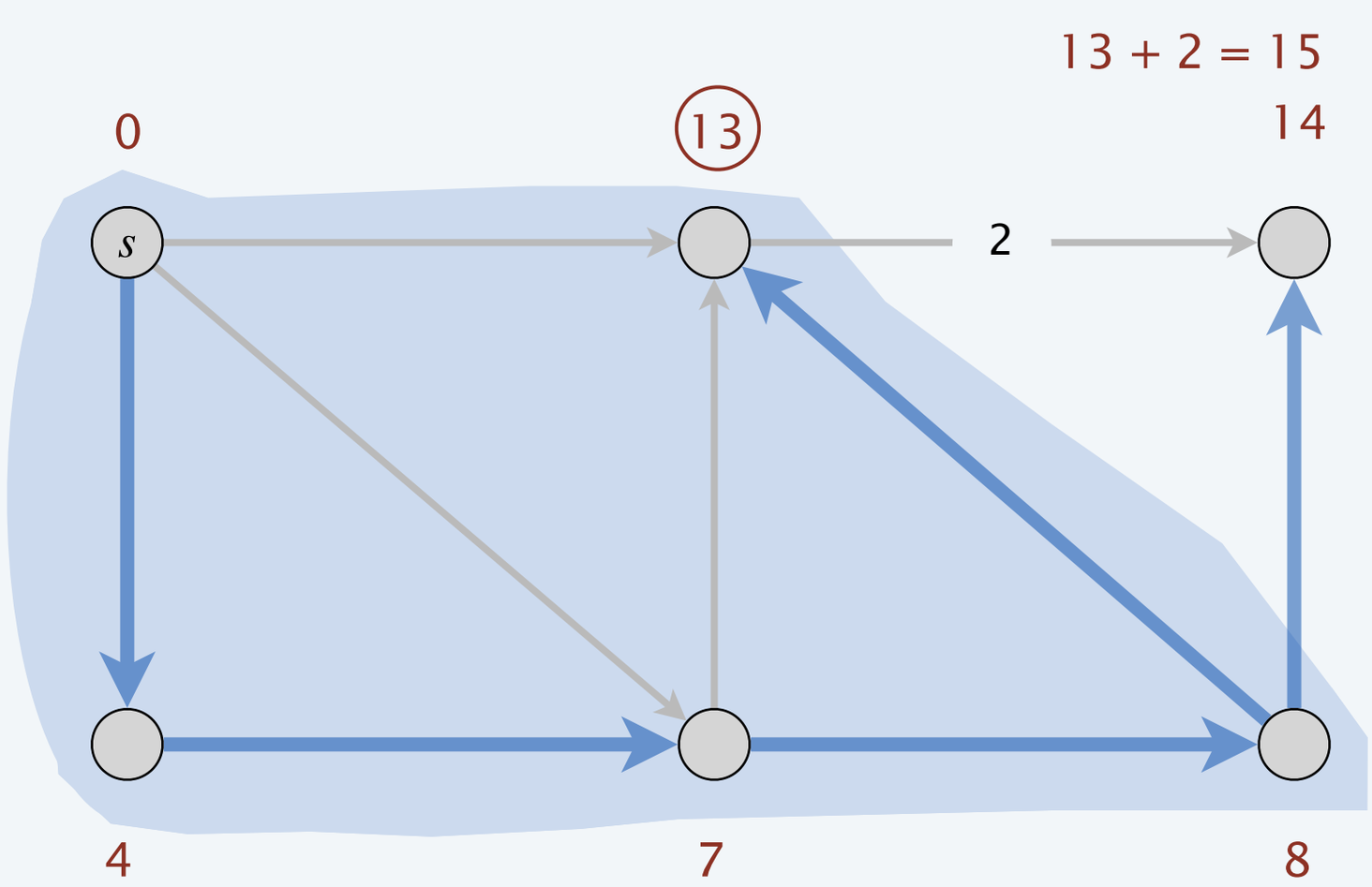
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

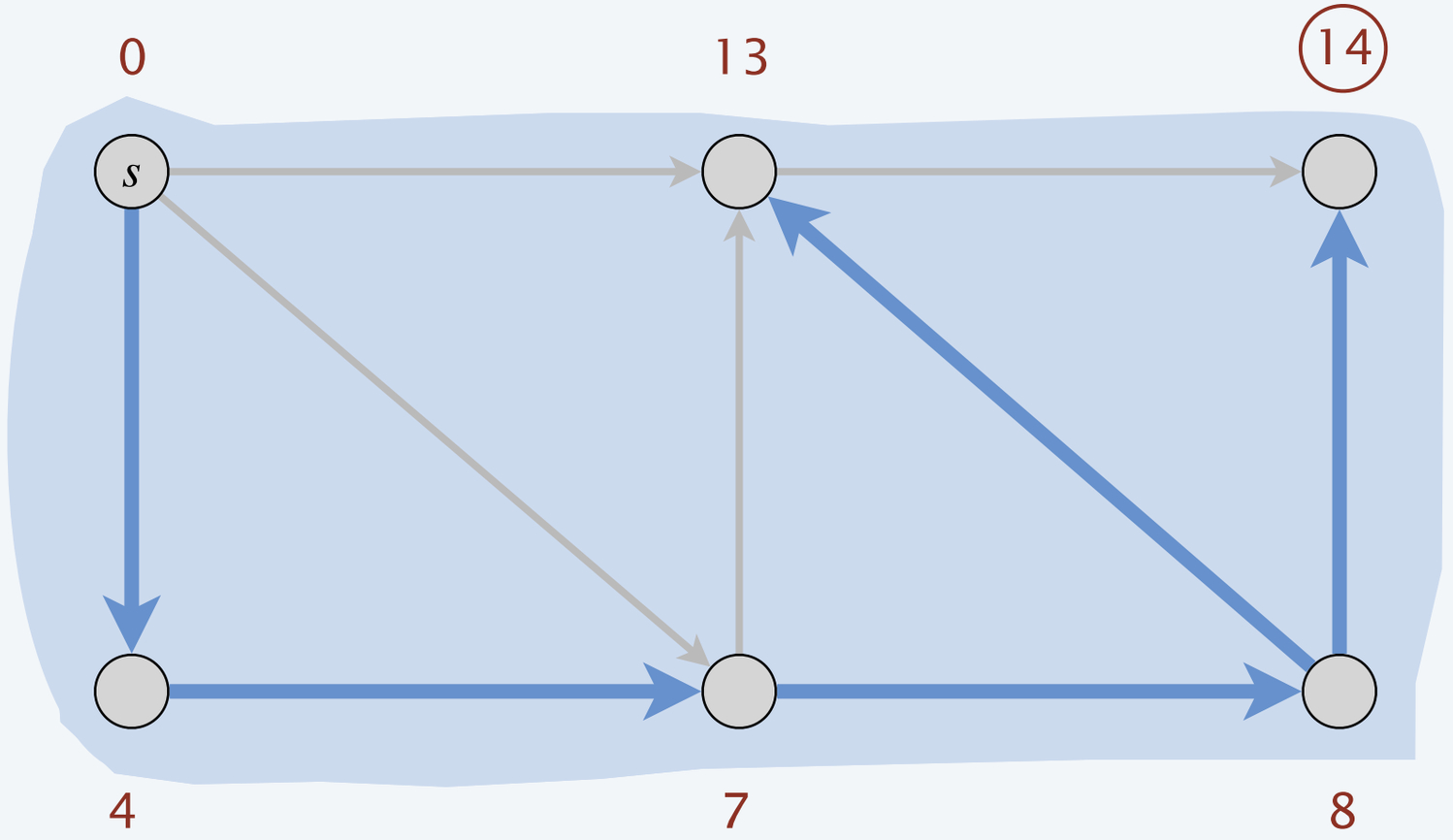
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

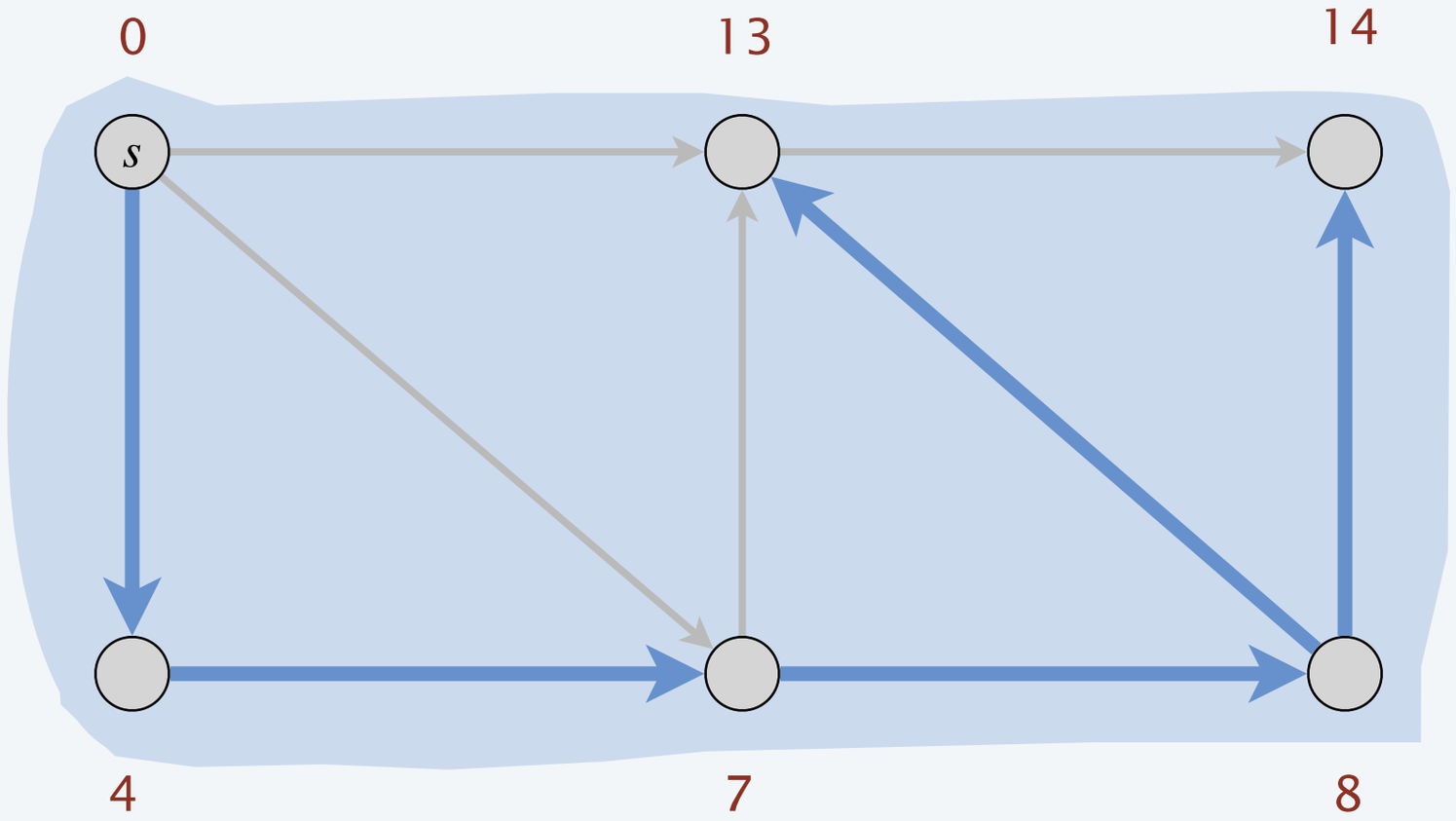
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Basic step. Choose unexplored node $u \notin S$ with minimum $\pi[u]$.

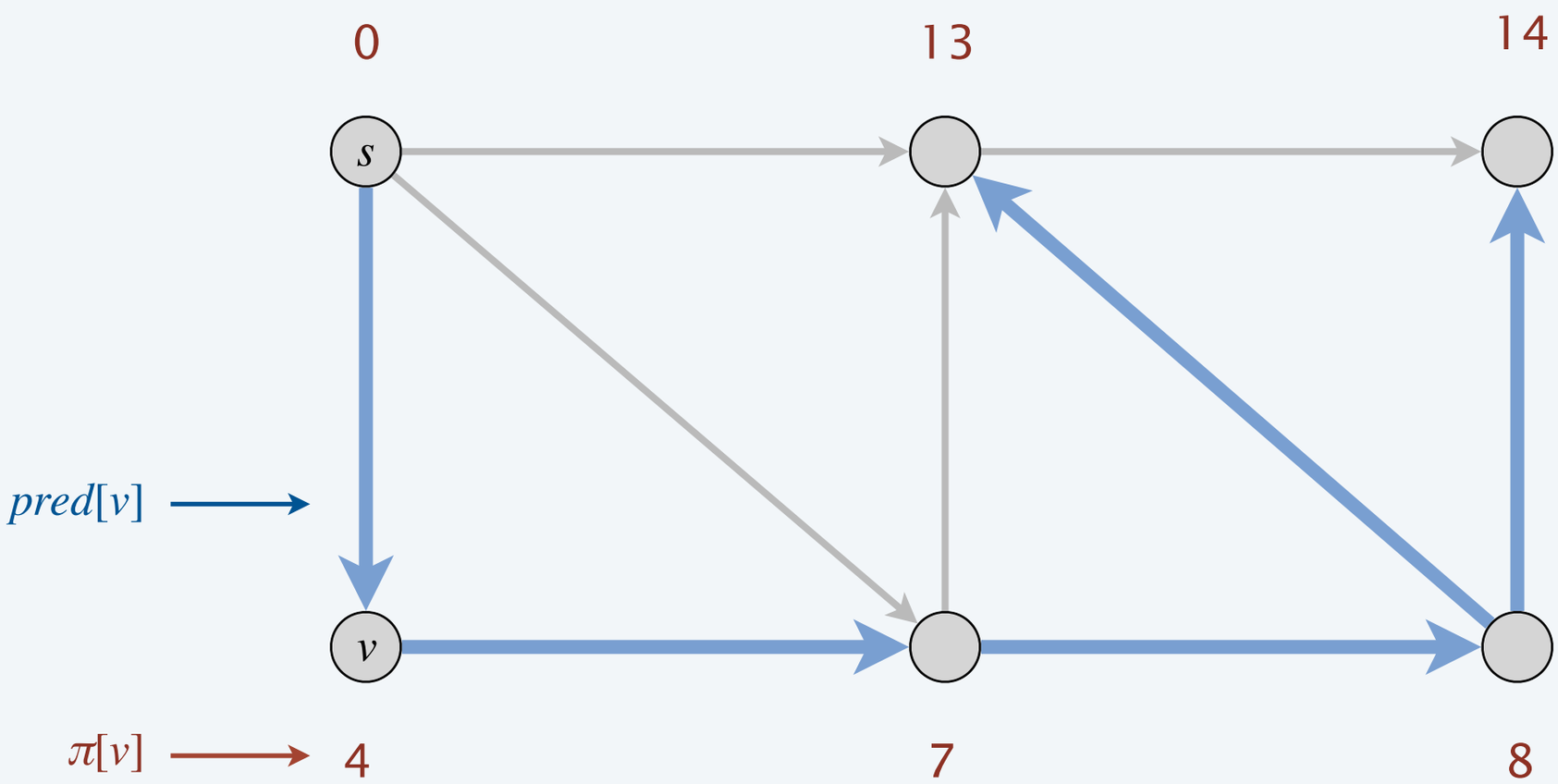
- Add u to S.
- For each edge $e = (u, v)$ leaving u, if $\pi[v] > \pi[u] + \ell_e$ then:
  - $\pi[v] \leftarrow \pi[u] + \ell_e$
  - $pred[v] \leftarrow e$



Dijkstra's algorithm demo (efficient implementation)

Termination.

- $\pi[v]$ = length of a shortest $s \rightsquigarrow v$ path.
- $pred[v]$ = last edge on a shortest $s \rightsquigarrow v$ path.



Dijkstra's algorithm: which priority queue?

Performance. Depends on PQ: n INSERT, n DELETE-MIN, $\leq m$ DECREASE-KEY.

- Array implementation optimal for dense graphs. ⟵ $\Theta(n^2)$ edges
- Binary heap much faster for sparse graphs. ⟵ $\Theta(n)$ edges
- 4-way heap worth the trouble in performance-critical situations.

priority queue	INSERT	DELETE-MIN	DECREASE-KEY	total
node-indexed array (A[i] = priority of i)	$O(1)$	$O(n)$	$O(1)$	$O(n^2)$
binary heap	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(m \log n)$
d-way heap (Johnson 1975)	$O(d \log_d n)$	$O(d \log_d n)$	$O(\log_d n)$	$O(m \log_{m/n} n)$
Fibonacci heap (Fredman–Tarjan 1984)	$O(1)$	$O(\log n)$ †	$O(1)$ †	$O(m + n \log n)$
integer priority queue (Thorup 2004)	$O(1)$	$O(\log \log n)$	$O(1)$	$O(m + n \log \log n)$

assumes $m \geq n$ † amortized

Shortest paths: quiz 3

How to solve the the single-source shortest paths problem in undirected graphs with positive edge lengths?

A. Replace each undirected edge with two antiparallel edges of same length. Run Dijkstra's algorithm in the resulting digraph.

B. Modify Dijkstra's algorithms so that when it processes node u, it consider all edges incident to u (instead of edges leaving u).

C. Either A or B.

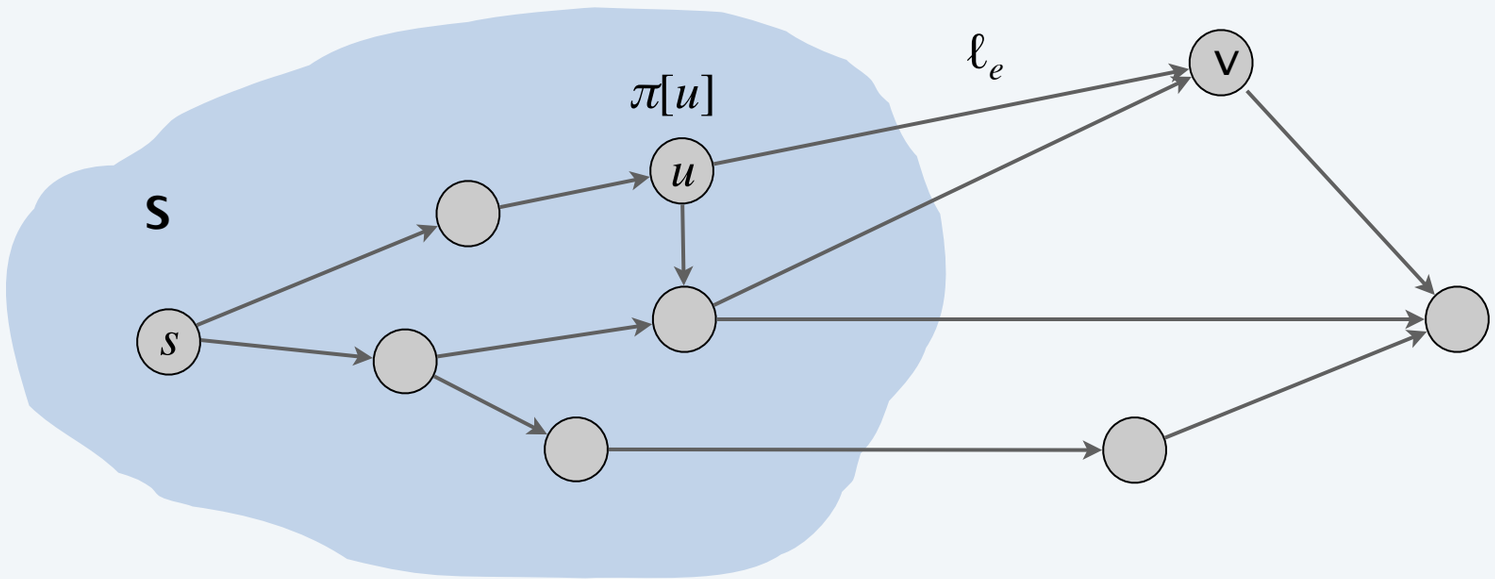
D. Neither A nor B.

Extensions of Dijkstra's algorithm

Dijkstra's algorithm and proof extend to several related problems:

- Shortest paths in undirected graphs: $\pi[v] \leq \pi[u] + \ell(u, v)$.
- Maximum capacity paths: $\pi[v] \geq \min \{ \pi[u], c(u, v) \}$.
- Maximum reliability paths: $\pi[v] \geq \pi[u] \times \gamma(u, v)$.
- ...

Key algebraic structure. Closed semiring (min-plus, bottleneck, Viterbi, ...).



$$
\begin{aligned}
a + b &= b + a \\
a + (b + c) &= (a + b) + c \\
a + 0 &= a \\
a \cdot (b \cdot c) &= (a \cdot b) \cdot c \\
a \cdot 0 &= 0 \cdot a = 0 \\
a \cdot 1 &= 1 \cdot a = a \\
a \cdot (b + c) &= a \cdot b + a \cdot c \\
(a + b) \cdot c &= a \cdot c + b \cdot c \\
a^* &= 1 + a \cdot a^* = 1 + a^* \cdot a
\end{aligned}
$$

Edsger Dijkstra

> *“ What’s the shortest way to travel from Rotterdam to Groningen? It is the algorithm for the shortest path, which I designed in about 20 minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path.”* — Edsger Dijsktra