CS5800 – ALGORITHMS

MODULE 6. GREEDY ALGORITHMS - I

Lesson 3: Shortest Paths

Ravi Sundaram

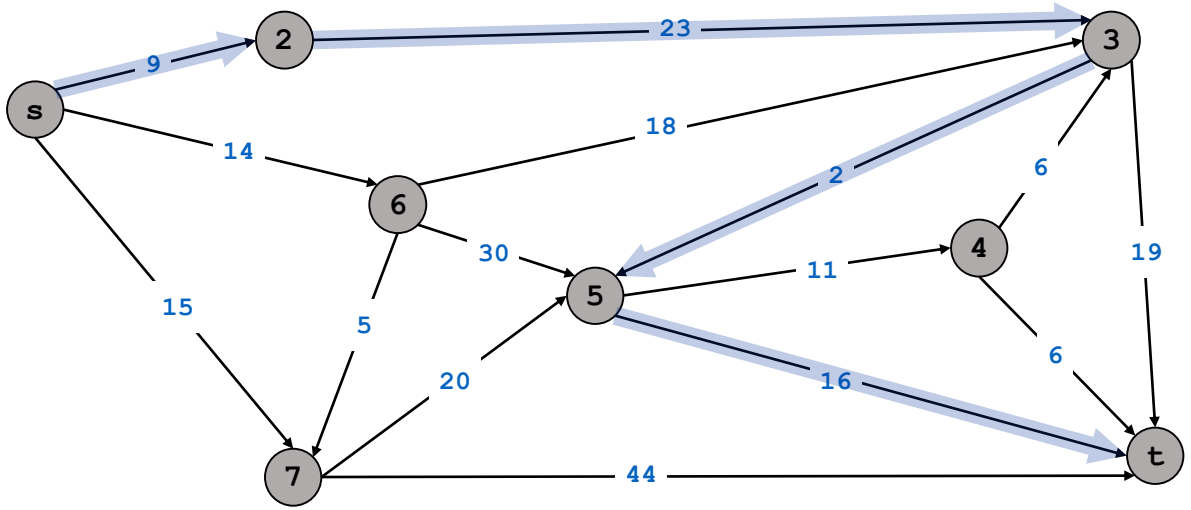
Topics

- Shortest Paths problem
- Dijkstra’s algorithm
- Proof of Correctness
- Implementation
- Demo
- Summary

Shortest Path Problem

- Shortest path network.
  - Directed graph G = (V, E).
  - Source s, destination t.
  - Length l_e = length of edge e.
- Shortest path problem: find shortest directed path from s to t.

cost of path = sum of edge costs in path



Cost of path s-2-3-5-t
= 9 + 23 + 2 + 16
= 50.

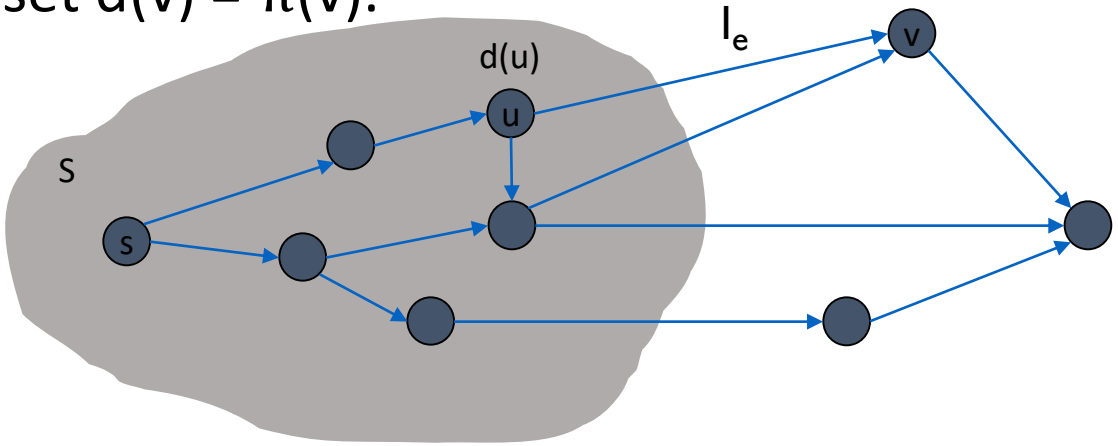
Dijkstra's Algorithm

- Dijkstra's algorithm.
  - Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
  - Initialize S = { s }, d(s) = 0.
  - Repeatedly choose unexplored node v which minimizes

shortest path to some u in explored part, followed by a single edge (u, v)

$$\pi(v) = \min_{e=(u,v):u \in S} d(u) + le$$

add v to S and set d(v) = π(v).



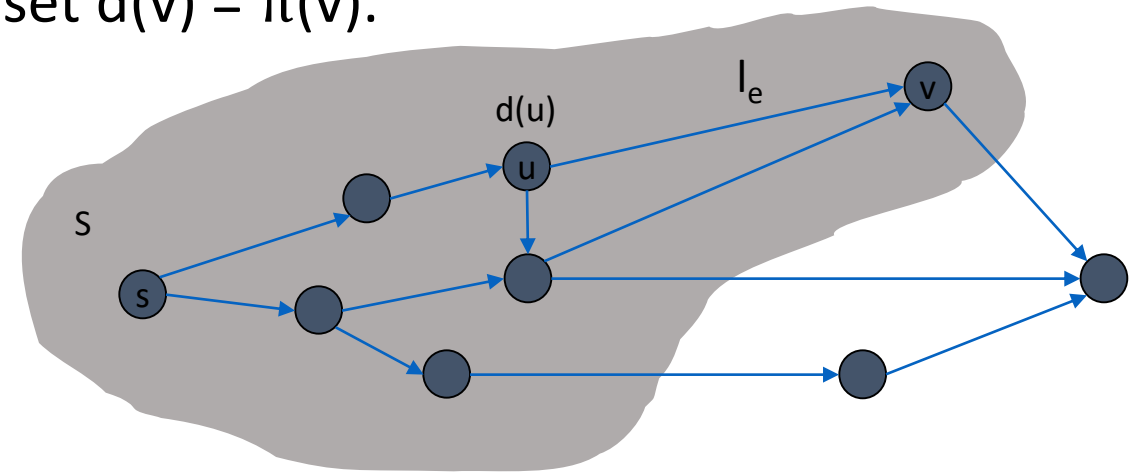
Dijkstra's Algorithm

- Dijkstra's algorithm.
  - Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
  - Initialize S = { s }, d(s) = 0.
  - Repeatedly choose unexplored node v which minimizes

shortest path to some u in explored part, followed by a single edge (u, v)

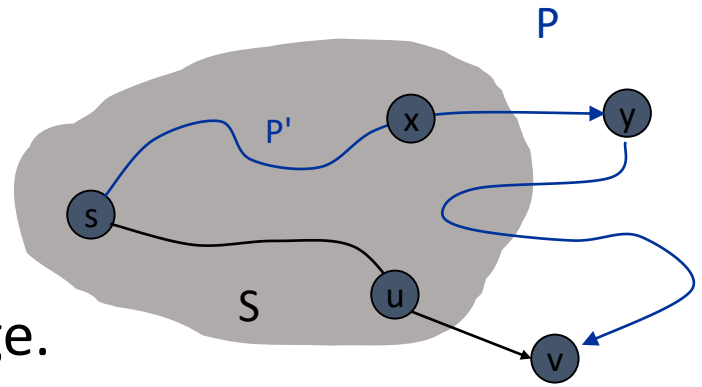
$$\pi(v) = \min_{e=(u,v):u \,\varepsilon\, S} d(u) + le$$

add v to S and set d(v) = π(v).



Dijkstra's Algorithm: Proof of Correctness

- Invariant. For each node $u \in S$, $d(u)$ is the length of the shortest s-u path.
- Pf. (by induction on $|S|$)
- Base case: $|S| = 1$ is trivial.
- Inductive hypothesis: Assume true for $|S| = k \geq 1$.
  - Let v be next node added to S and let u-v be the chosen edge.
  - The shortest s-u path plus (u, v) is an s-v path of length $\pi(v)$.
  - Consider any s-v path P. We'll see that it's no shorter than $\pi(v)$.
  - Let x-y be the first edge in P that leaves S, and let P' be the sub-path to x.
  - P is already too long as soon as it leaves S.



$$l(P) \geq l(P') + l(x,y) \geq d(x) + l(x, y) \geq \pi(y) \geq \pi(v)$$

nonnegative weights

inductive hypothesis

defn of $\pi(y)$

Dijkstra chose v instead of y

Dijkstra's Algorithm: Implementation

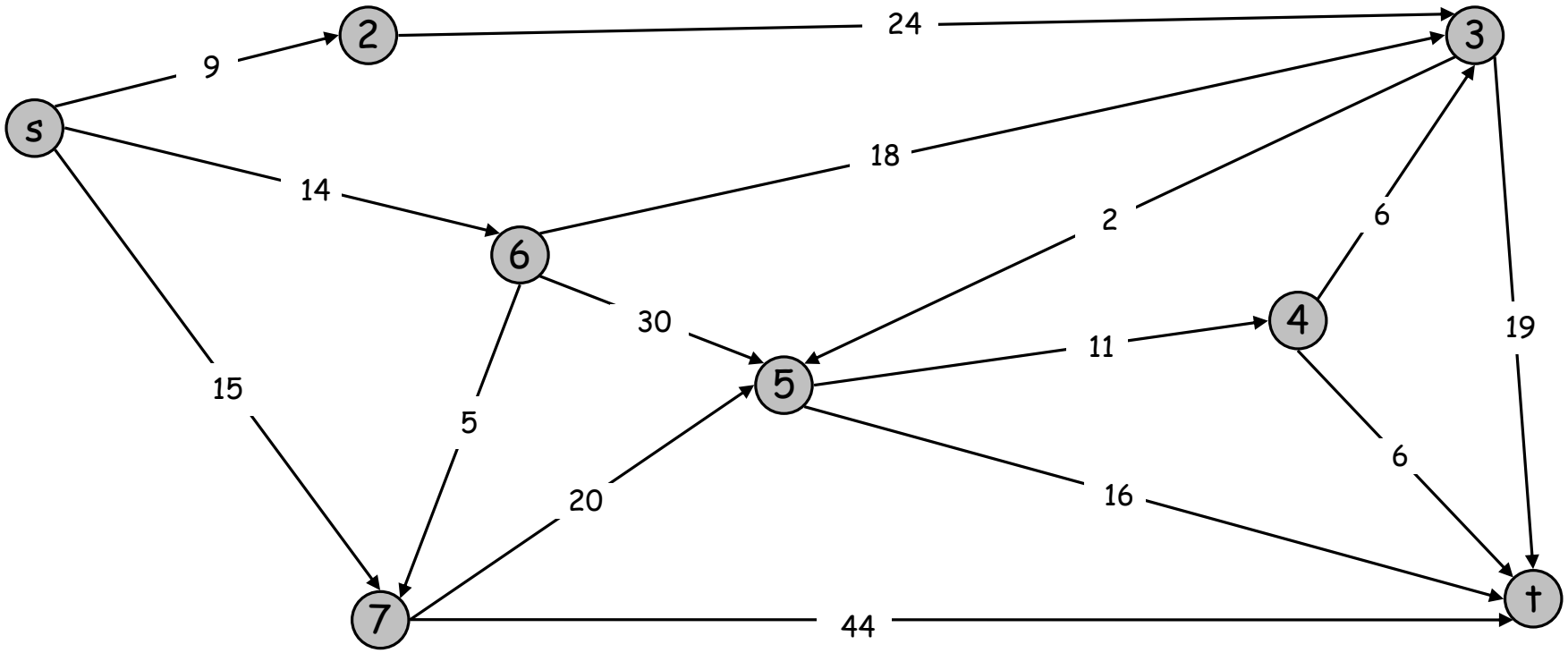
- For each unexplored node, explicitly maintain $\pi(v) = \min_{e=(u,v):u \in S} d(u) + l_e$
  - Next node to explore = node with minimum $\pi(v)$.
  - When exploring v, for each incident edge e = (v, w), update
    $\pi(w) = \min \{\pi(w), \pi(v) + l_e\}$
- Efficient implementation. Maintain a priority queue of unexplored nodes, prioritized by $\pi(v)$.

PQ Operation	Dijkstra	Array	Binary heap	d-way Heap	Fib heap †
Insert	n	n	log n	$d \log_d n$	1
ExtractMin	n	n	log n	$d \log_d n$	log n
ChangeKey	m	1	log n	$\log_d n$	1
IsEmpty	n	1	1	1	1
Total		n^2	m log n	$m \log_{m/n} n$	m + n log n

† Individual ops are amortized bounds

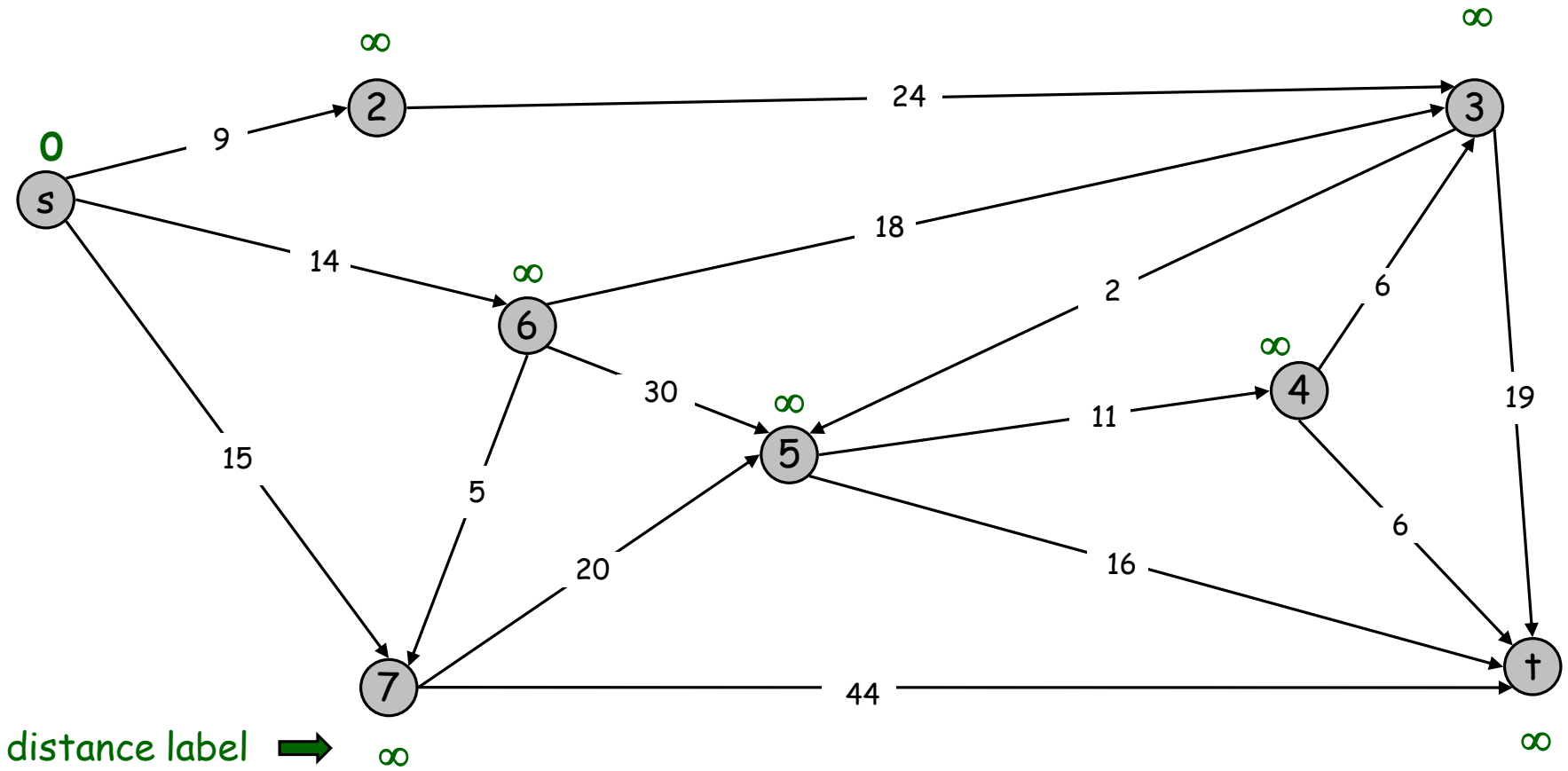
Dijkstra's Shortest Path Algorithm

Find shortest path from s to t.



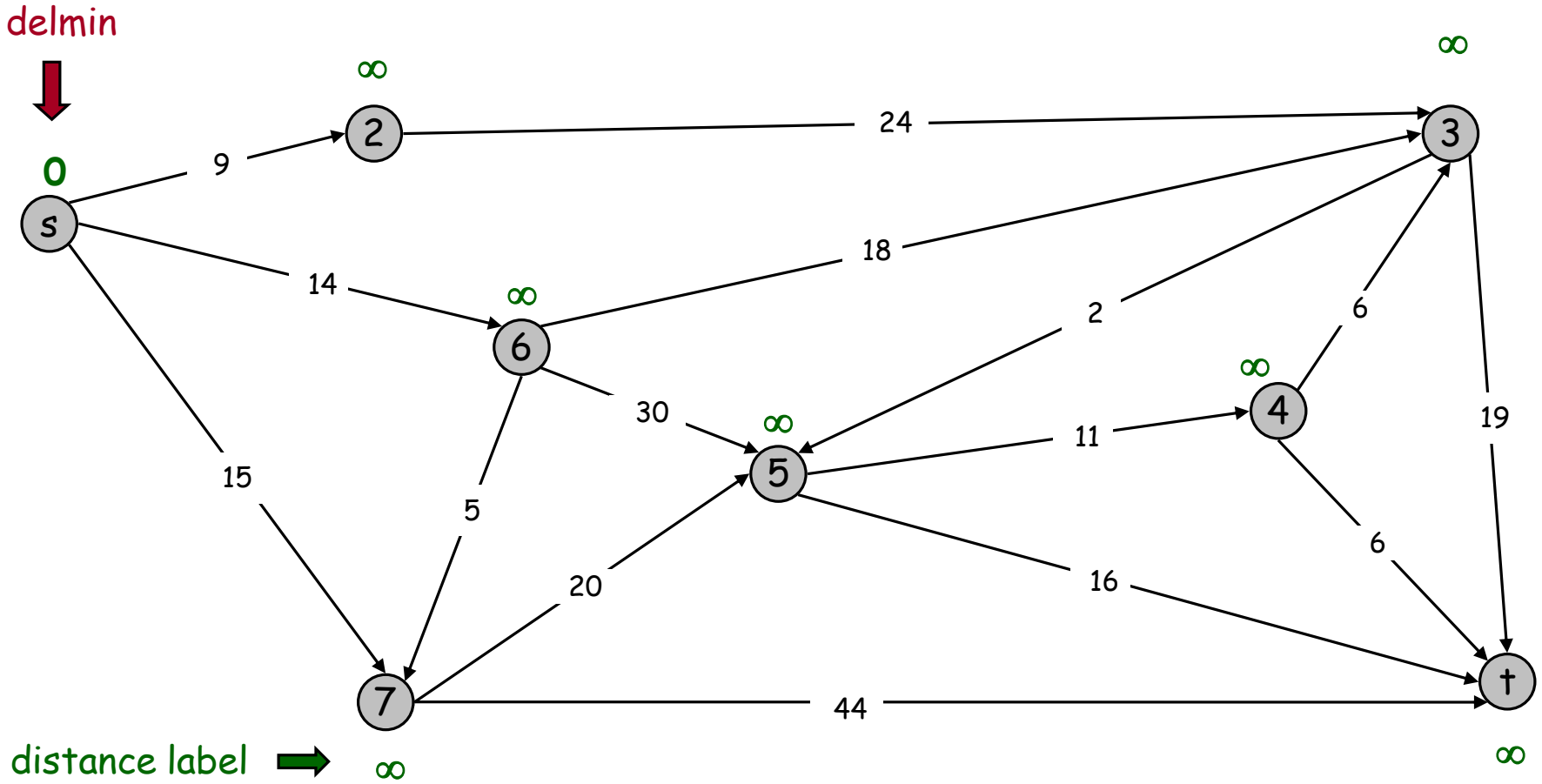
Dijkstra's Shortest Path Algorithm

S = { }
PQ = { s, 2, 3, 4, 5, 6, 7, t }



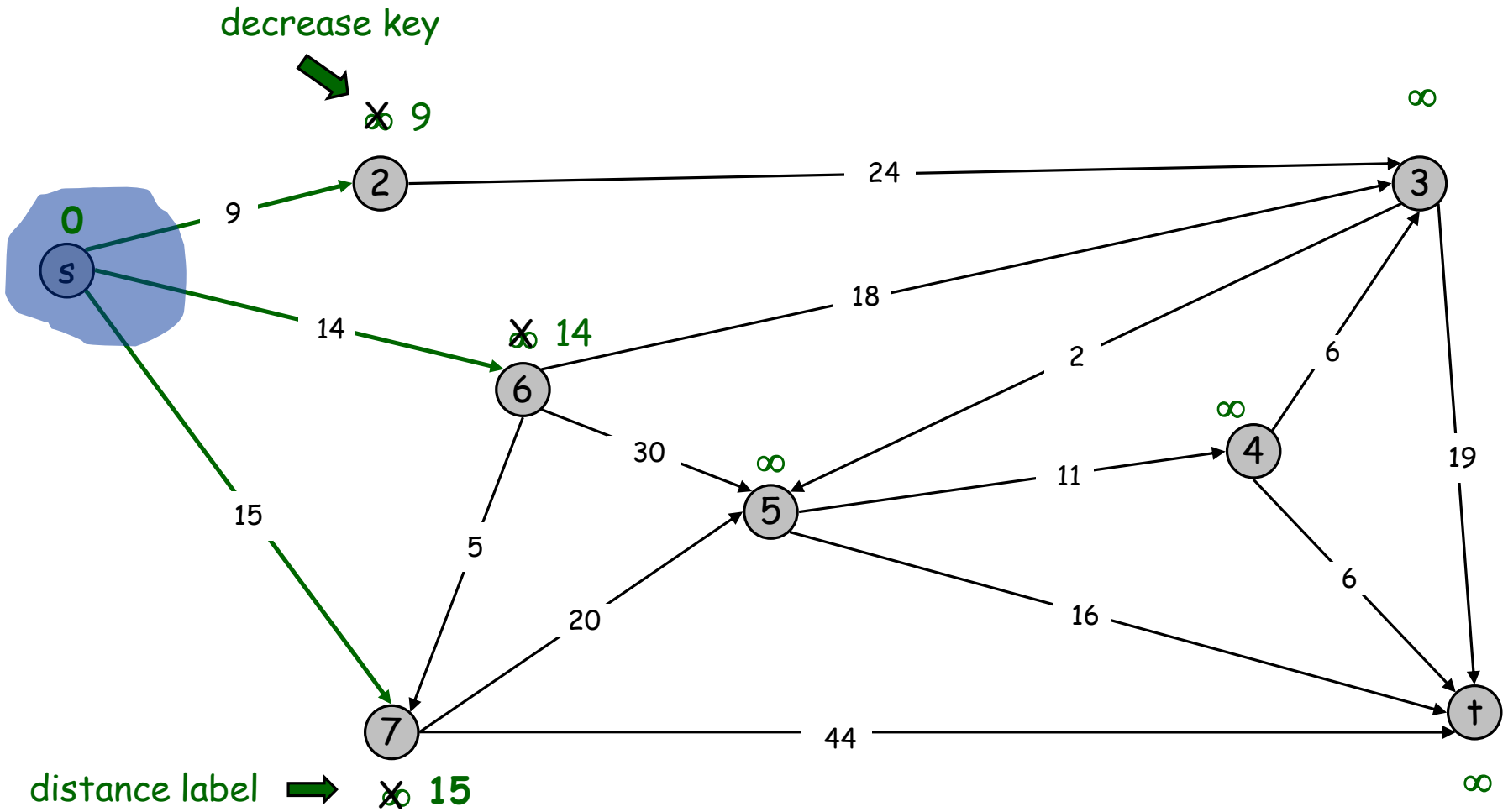
Dijkstra's Shortest Path Algorithm

S = { }
PQ = { s, 2, 3, 4, 5, 6, 7, t }



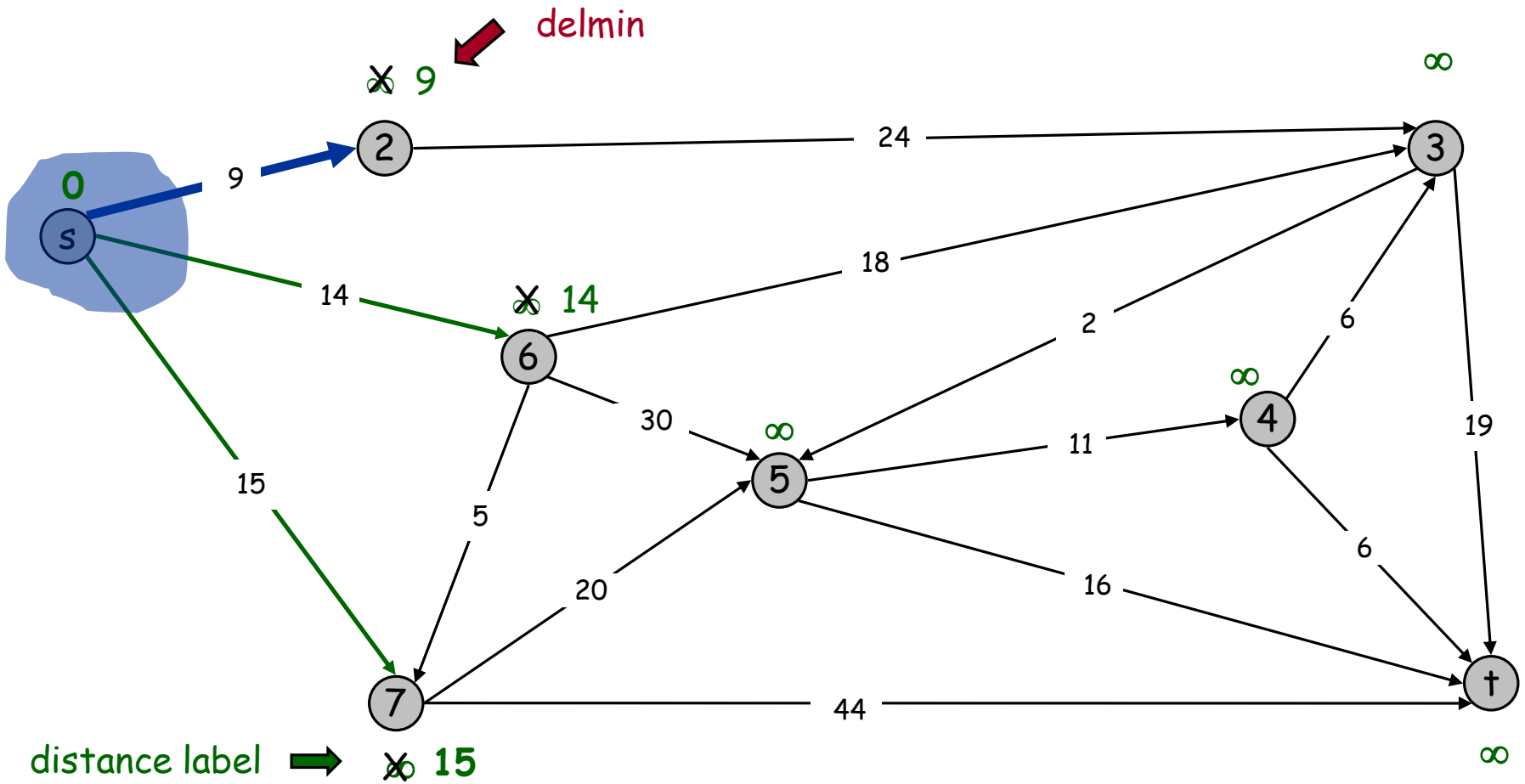
Dijkstra's Shortest Path Algorithm

S = { s }
PQ = { 2, 3, 4, 5, 6, 7, t }



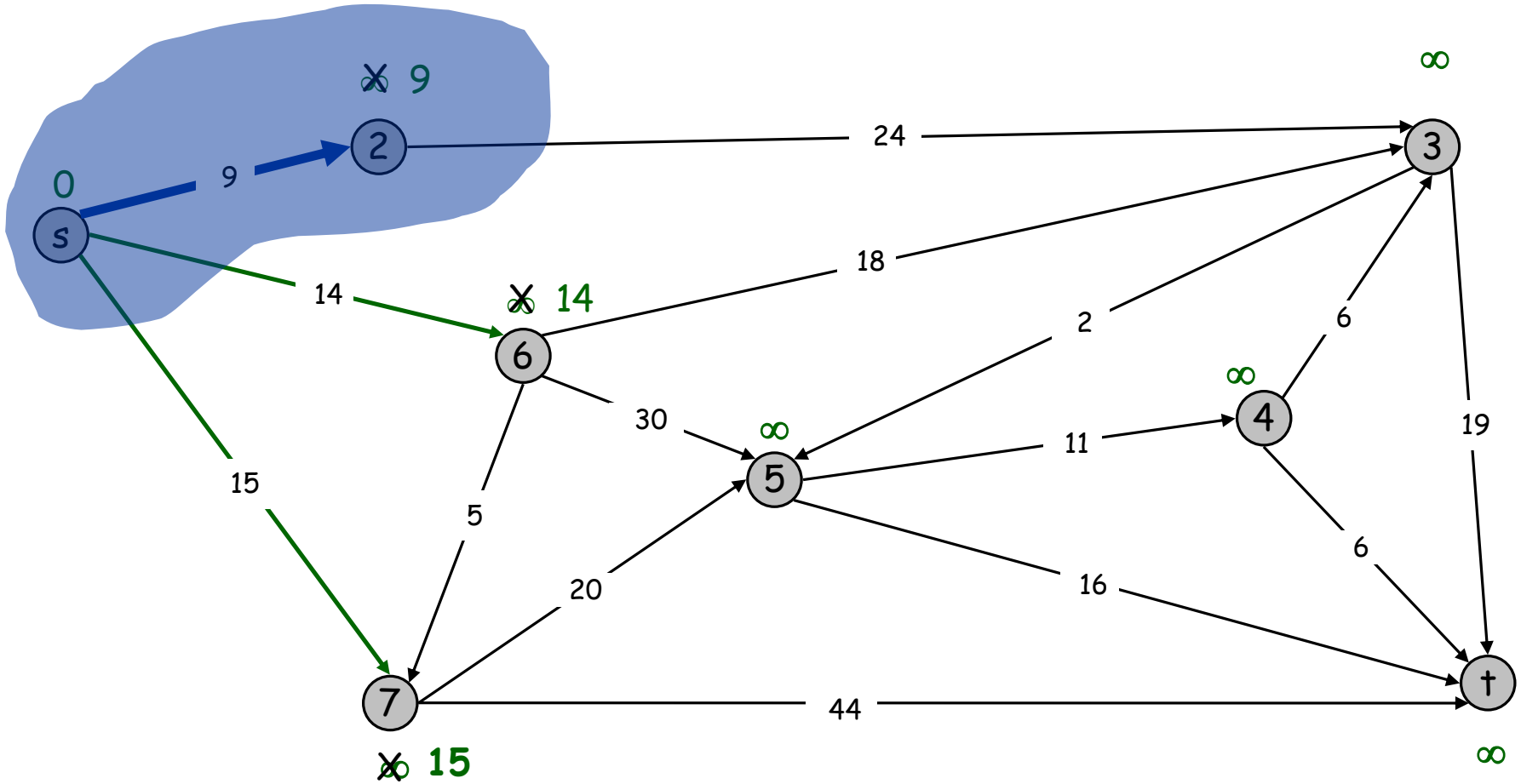
Dijkstra's Shortest Path Algorithm

S = { s }
PQ = { 2, 3, 4, 5, 6, 7, t }



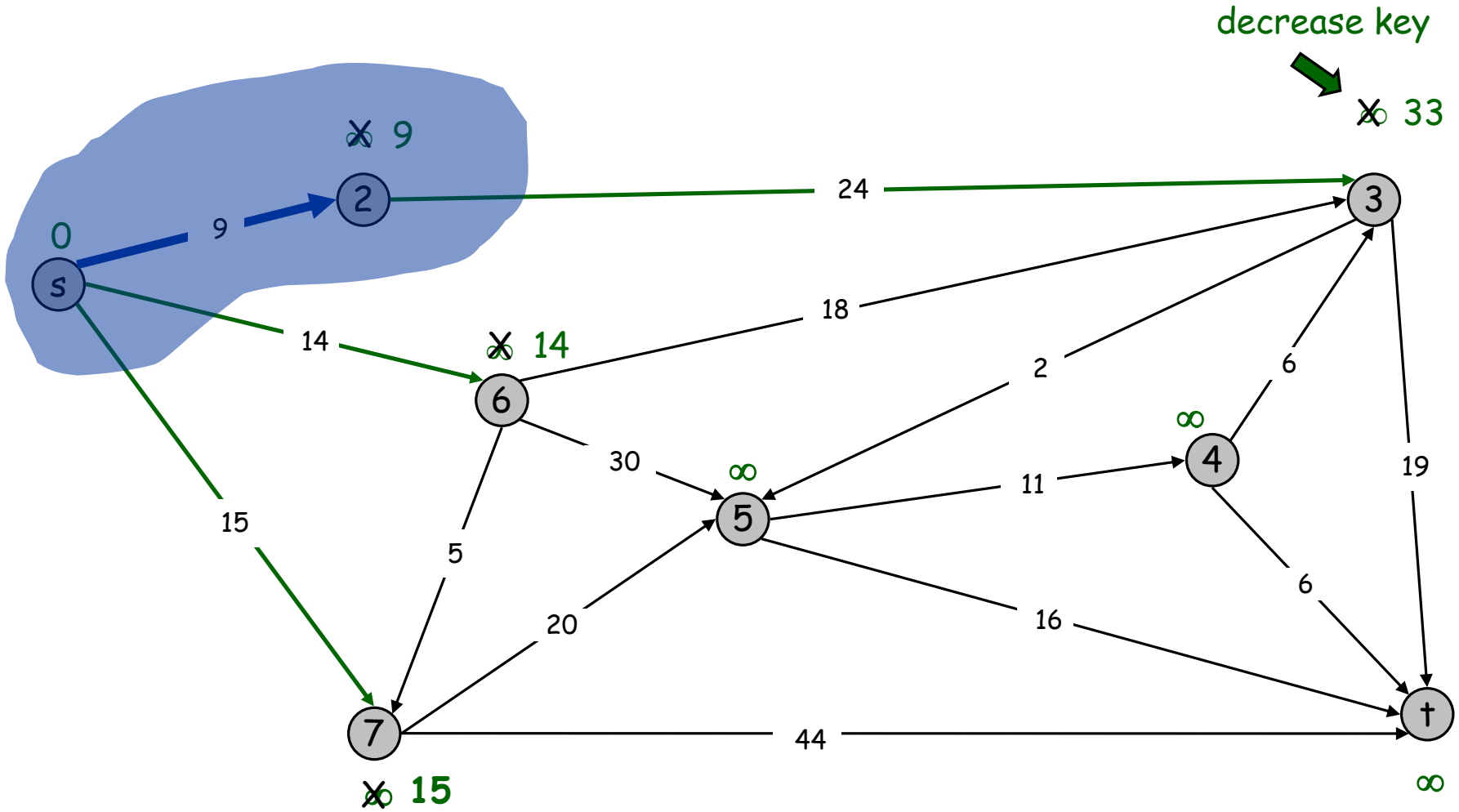
Dijkstra's Shortest Path Algorithm

S = { s, 2 }
PQ = { 3, 4, 5, 6, 7, t }



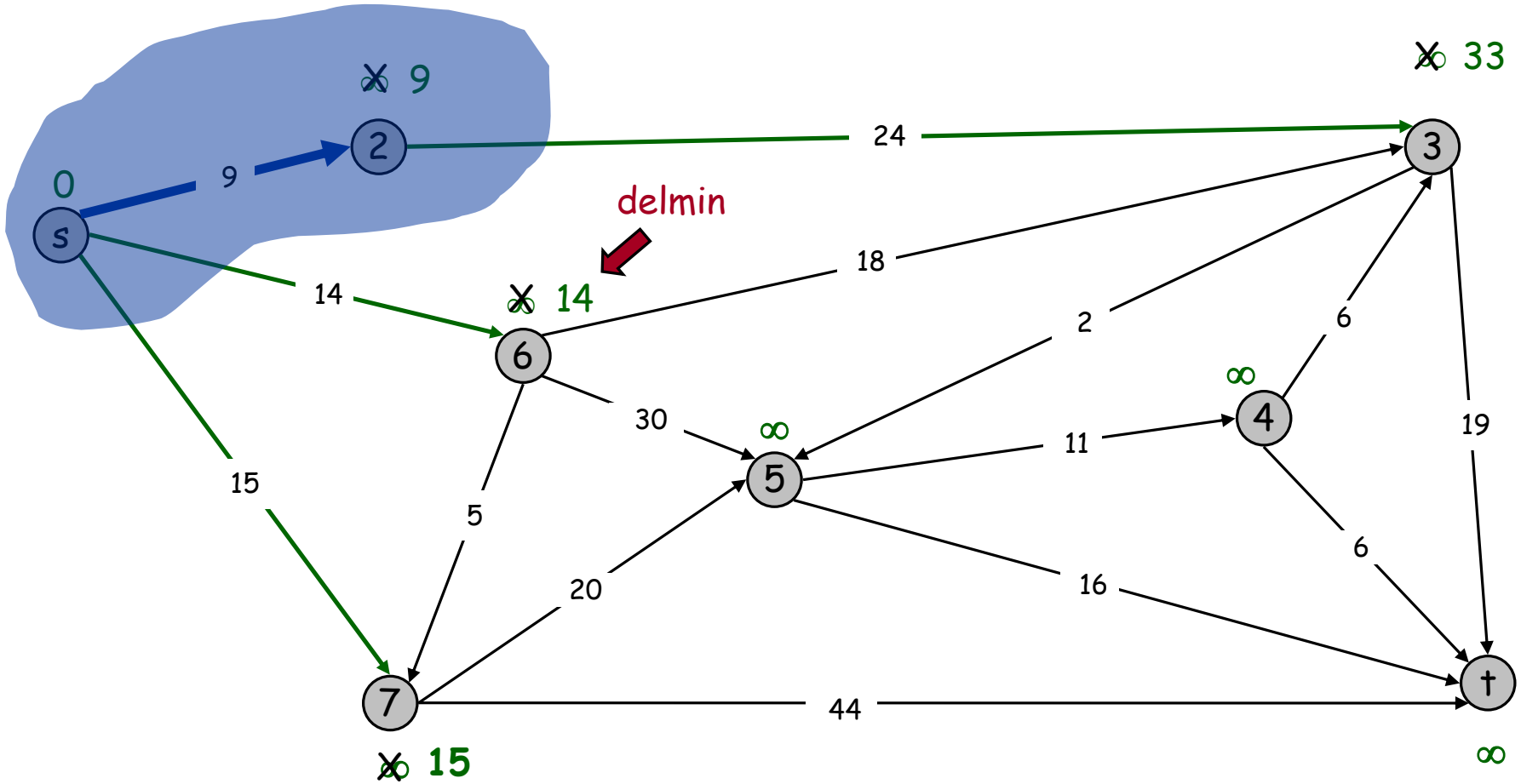
Dijkstra's Shortest Path Algorithm

S = { s, 2 }
PQ = { 3, 4, 5, 6, 7, t }



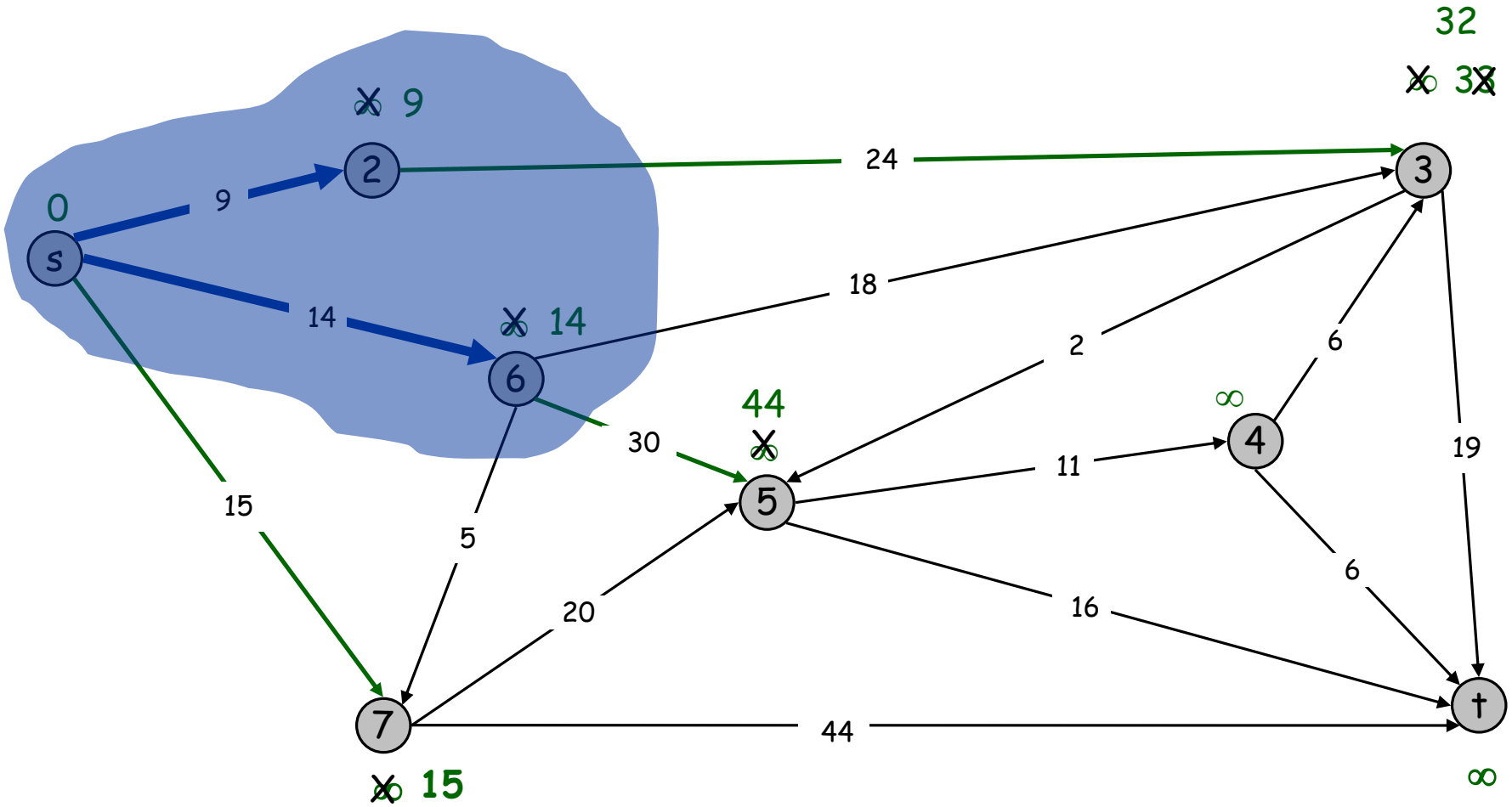
Dijkstra's Shortest Path Algorithm

S = { s, 2 }
PQ = { 3, 4, 5, 6, 7, t }



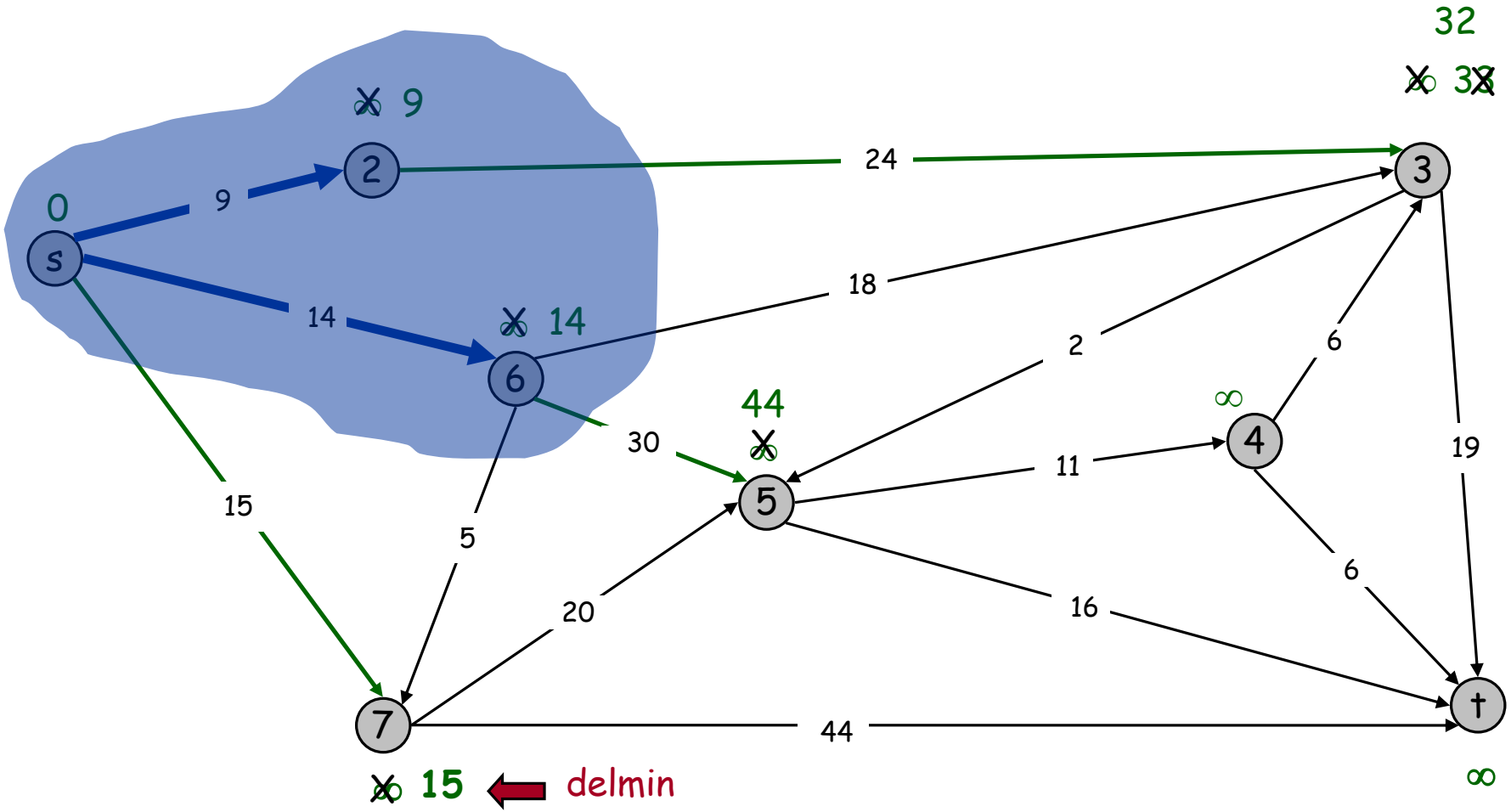
Dijkstra's Shortest Path Algorithm

S = { s, 2, 6 }
PQ = { 3, 4, 5, 7, t }



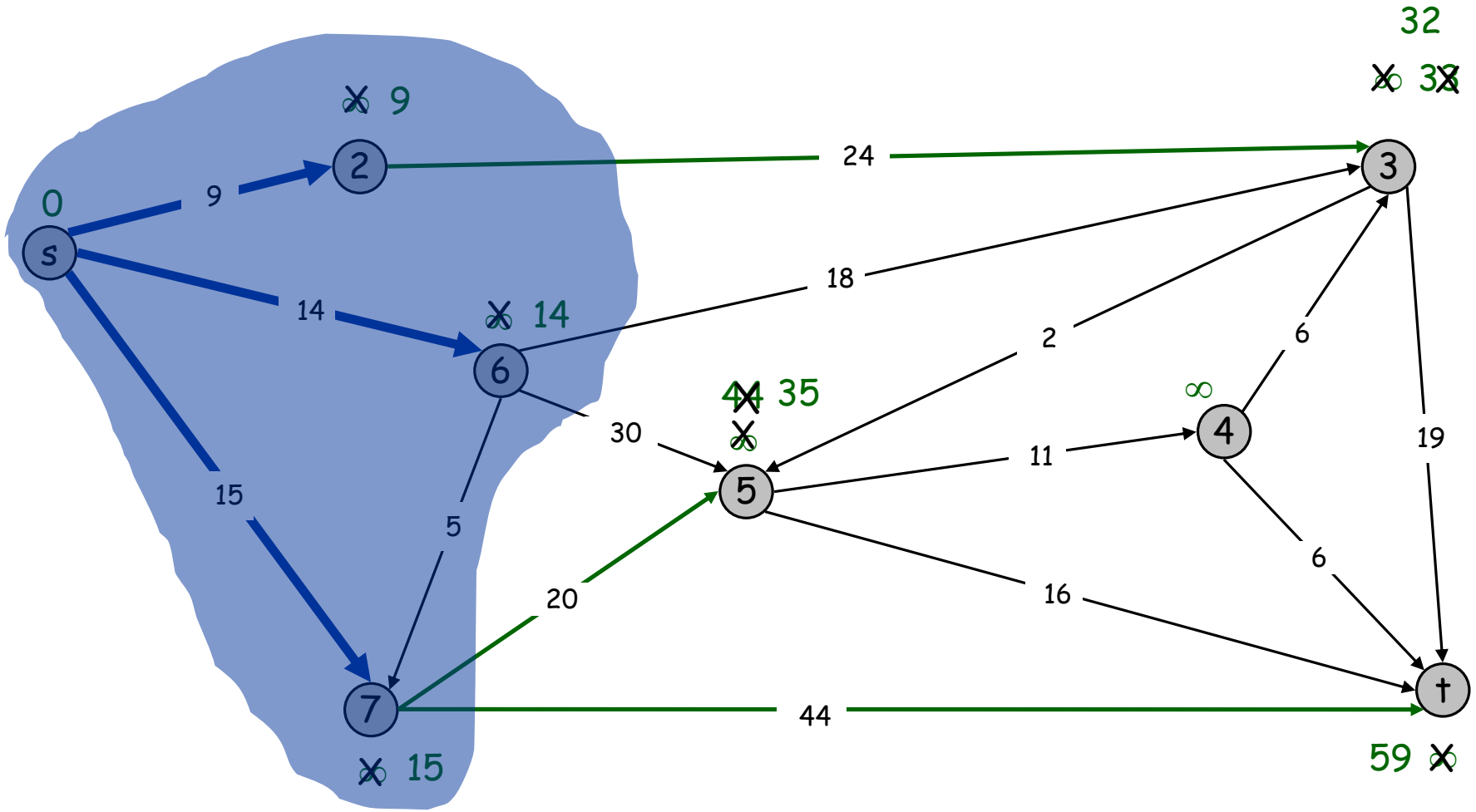
Dijkstra's Shortest Path Algorithm

S = { s, 2, 6 }
PQ = { 3, 4, 5, 7, t }



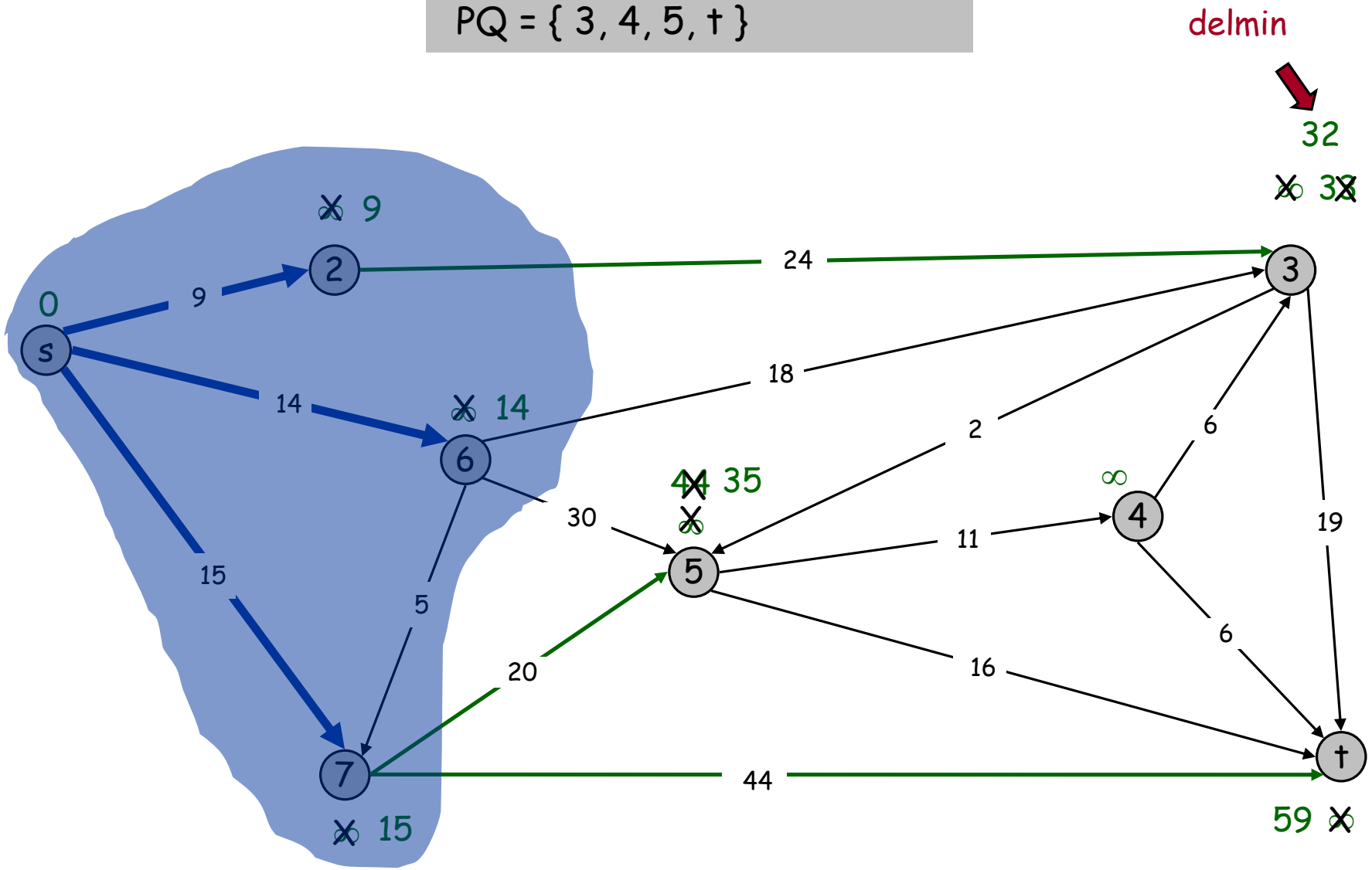
Dijkstra's Shortest Path Algorithm

S = { s, 2, 6, 7 }
PQ = { 3, 4, 5, t }



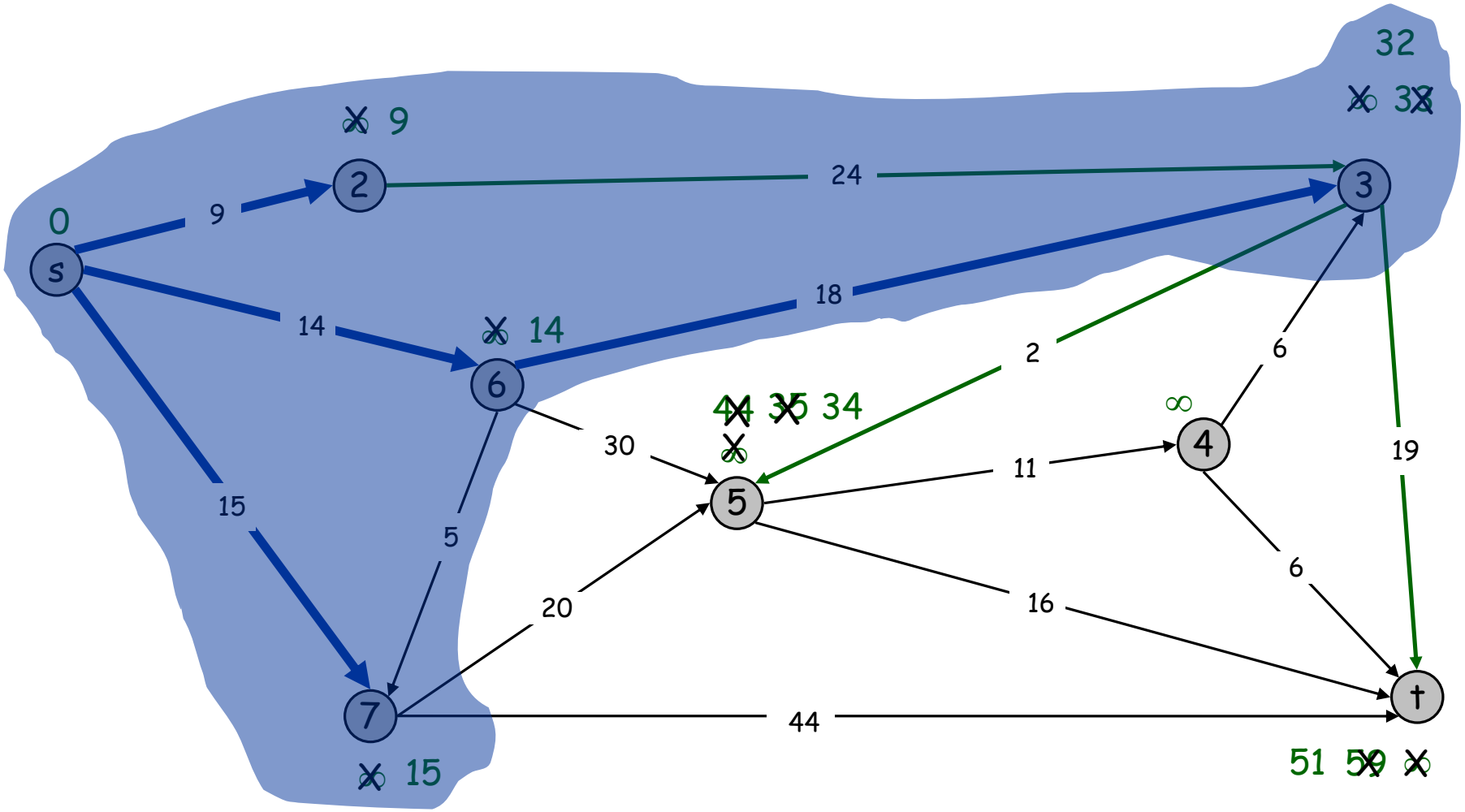
Dijkstra's Shortest Path Algorithm

S = { s, 2, 6, 7 }
PQ = { 3, 4, 5, t }



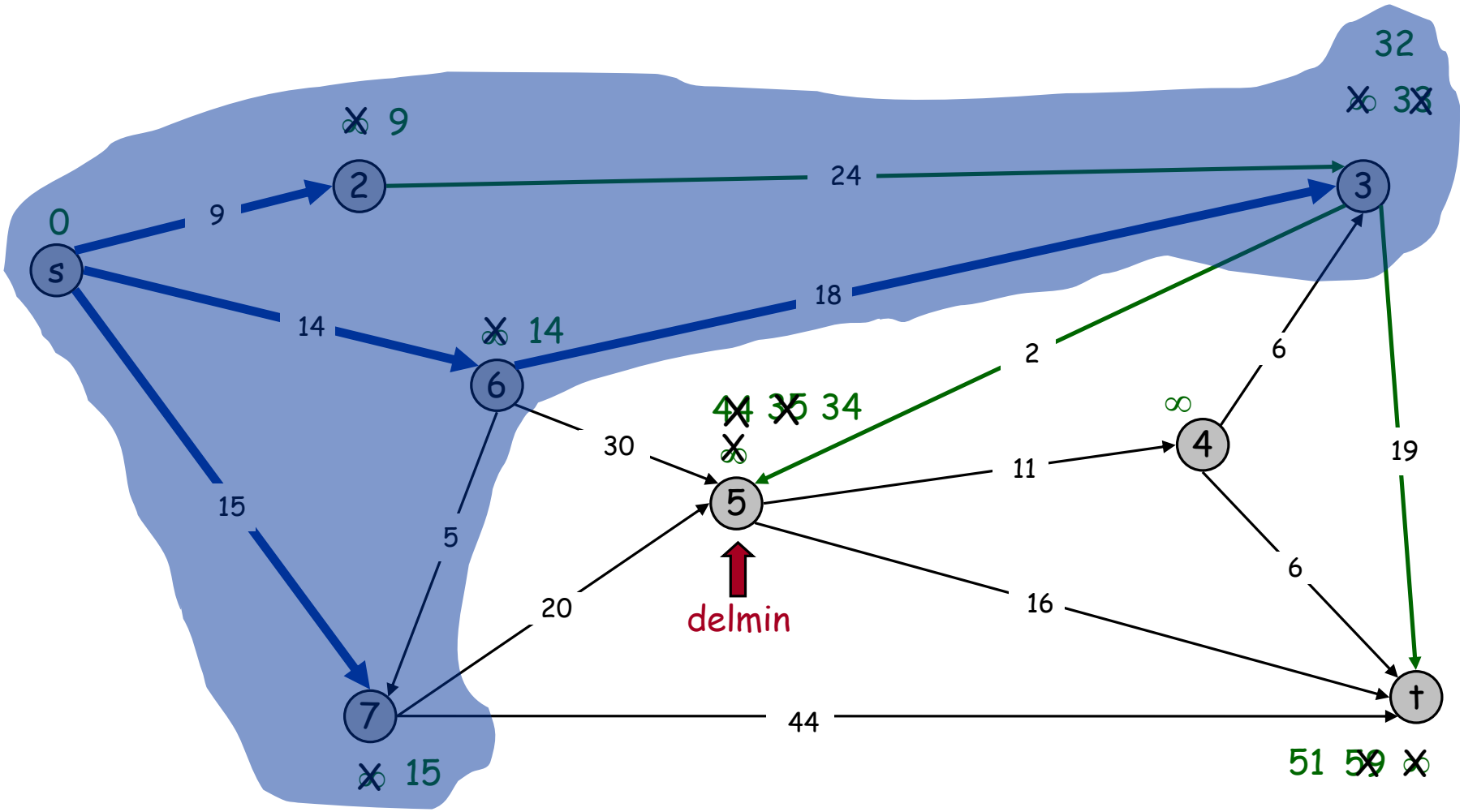
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 6, 7 }
PQ = { 4, 5, t }



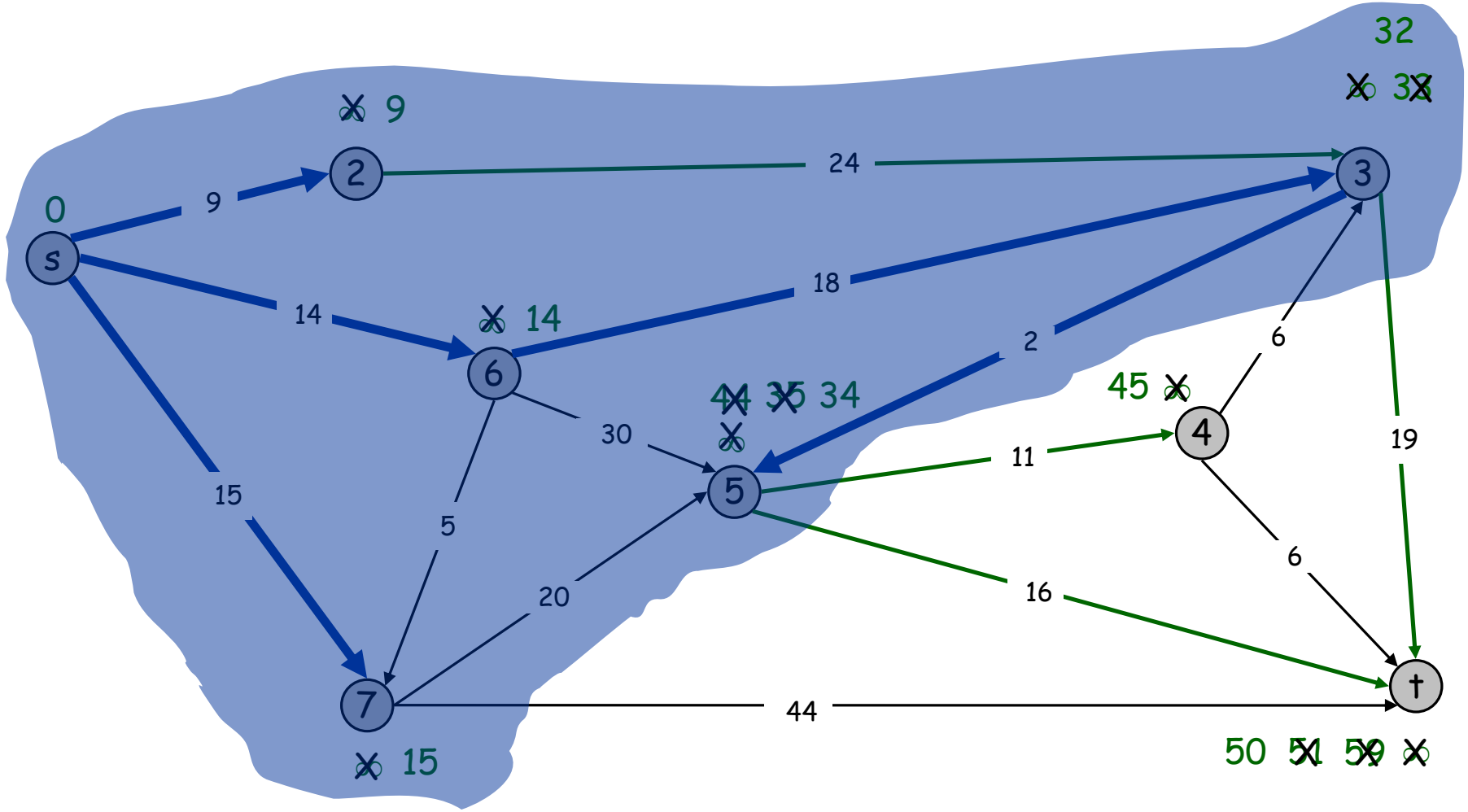
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 6, 7 }
PQ = { 4, 5, t }



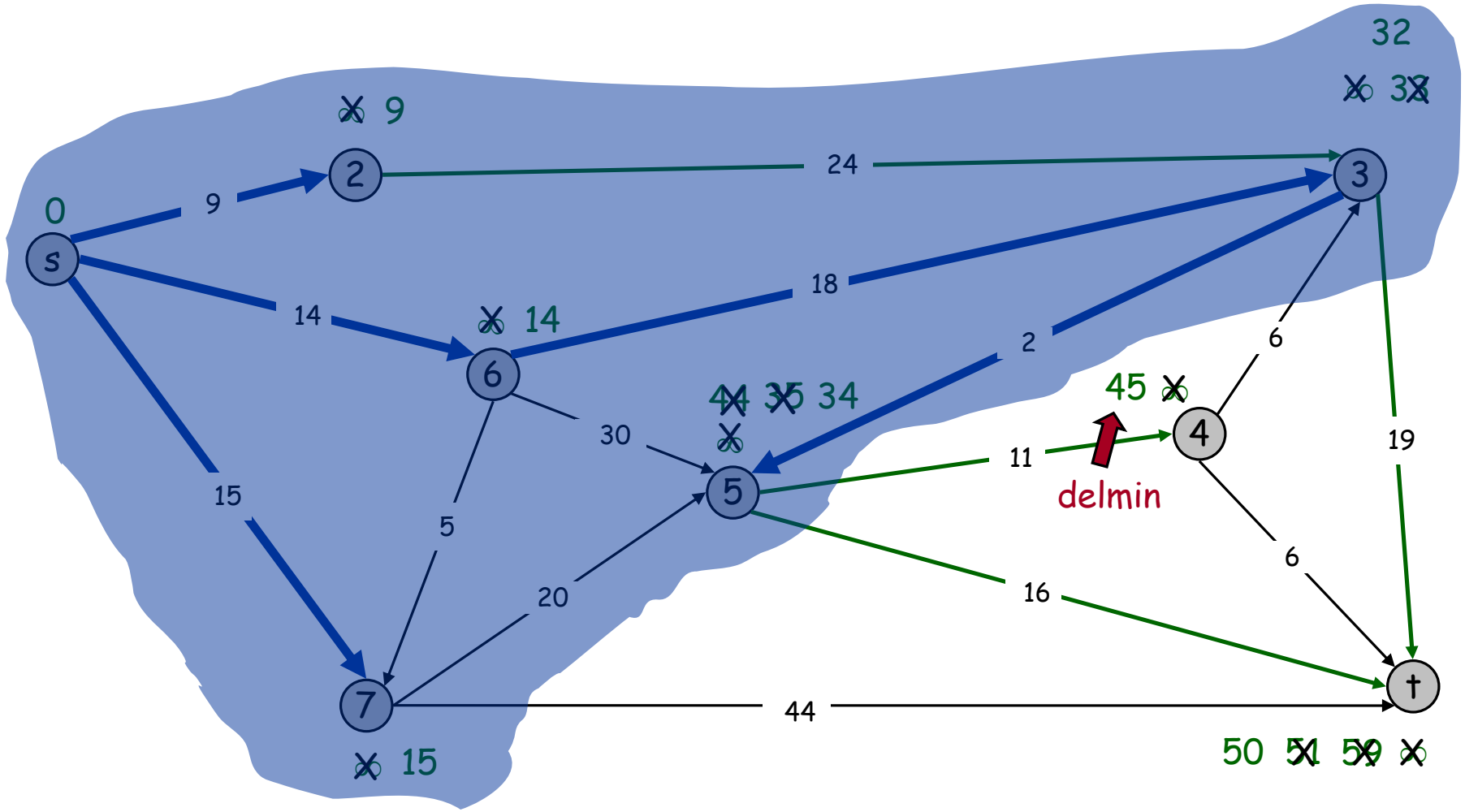
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 5, 6, 7 }
PQ = { 4, t }



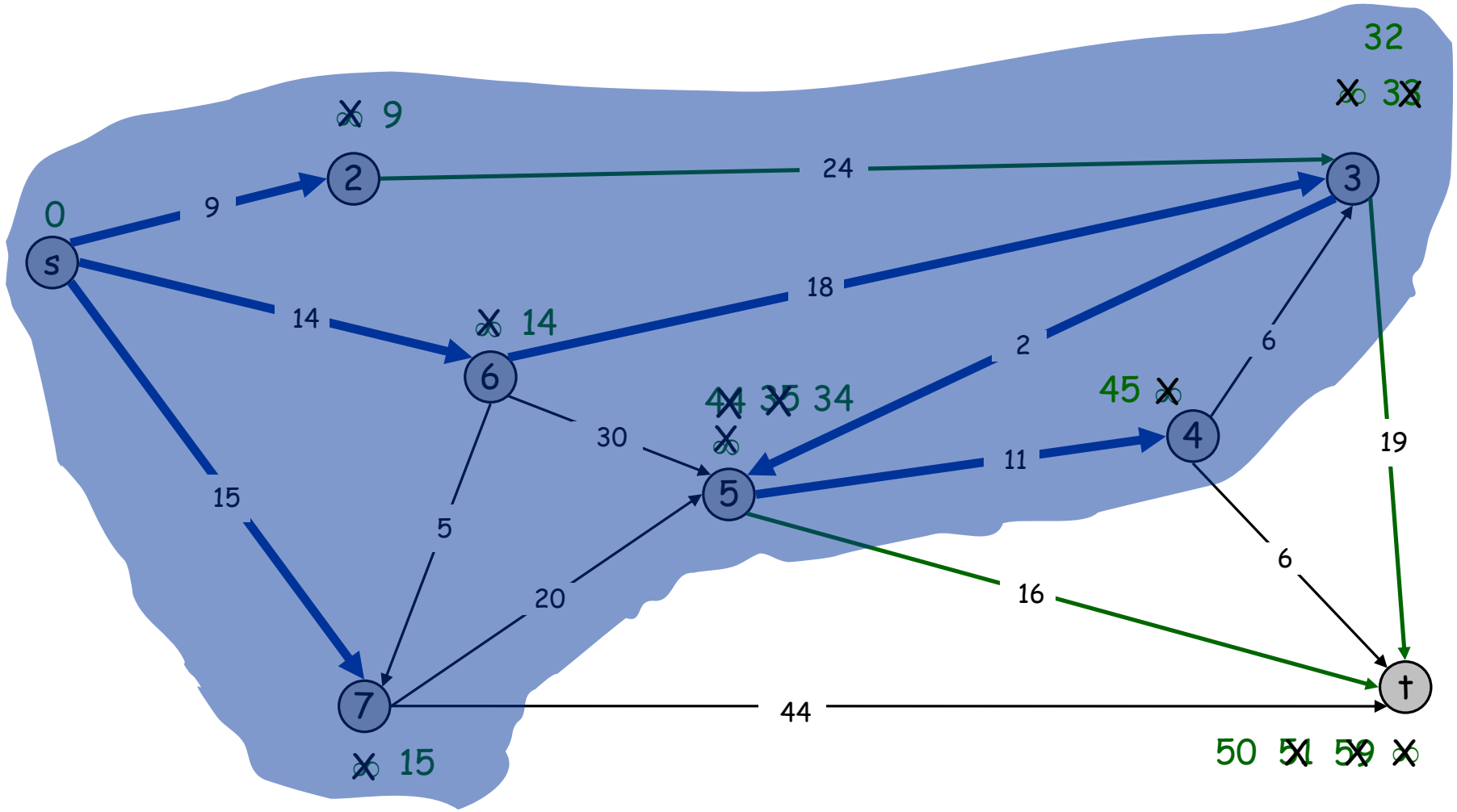
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 5, 6, 7 }
PQ = { 4, t }



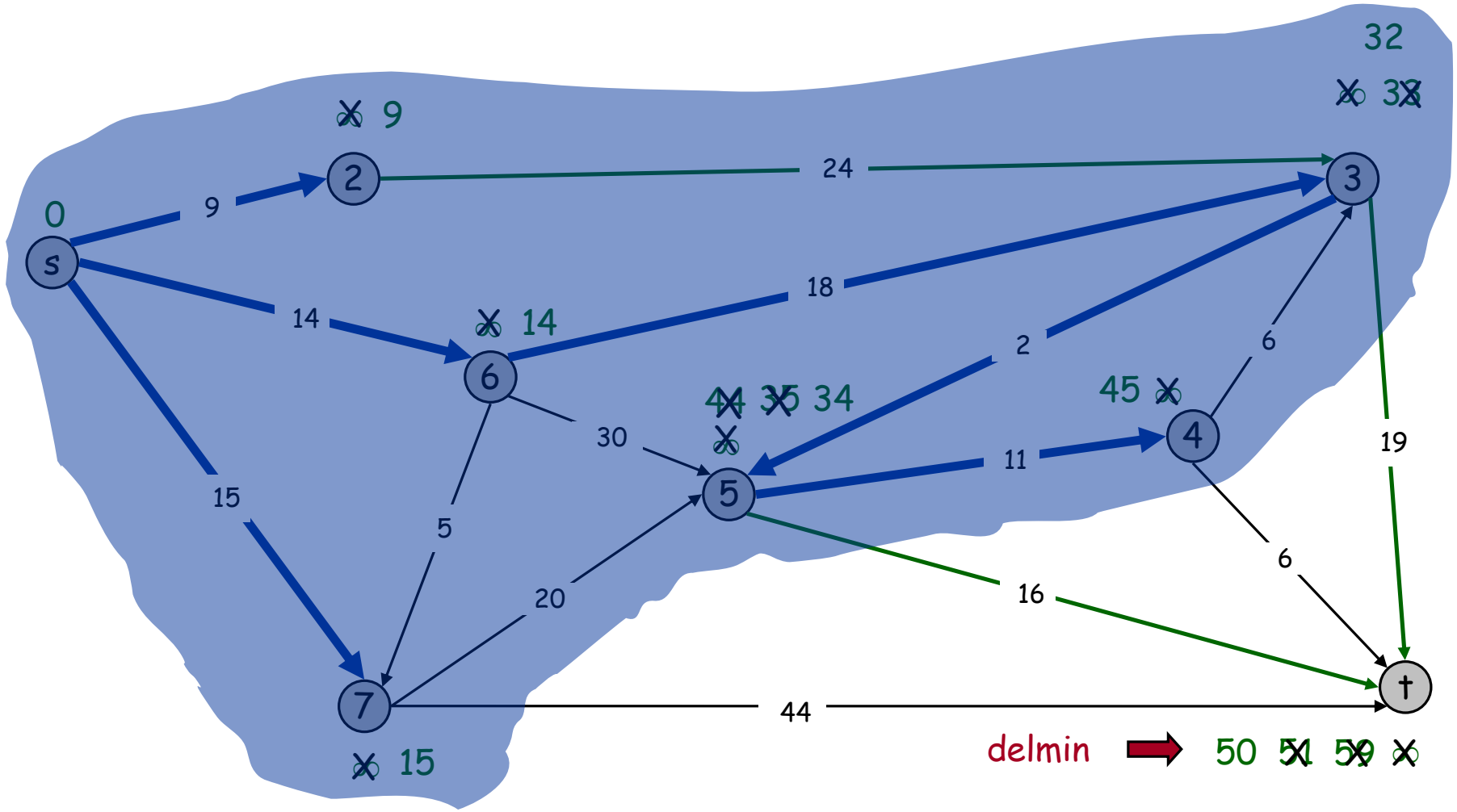
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 4, 5, 6, 7 }
PQ = { t }



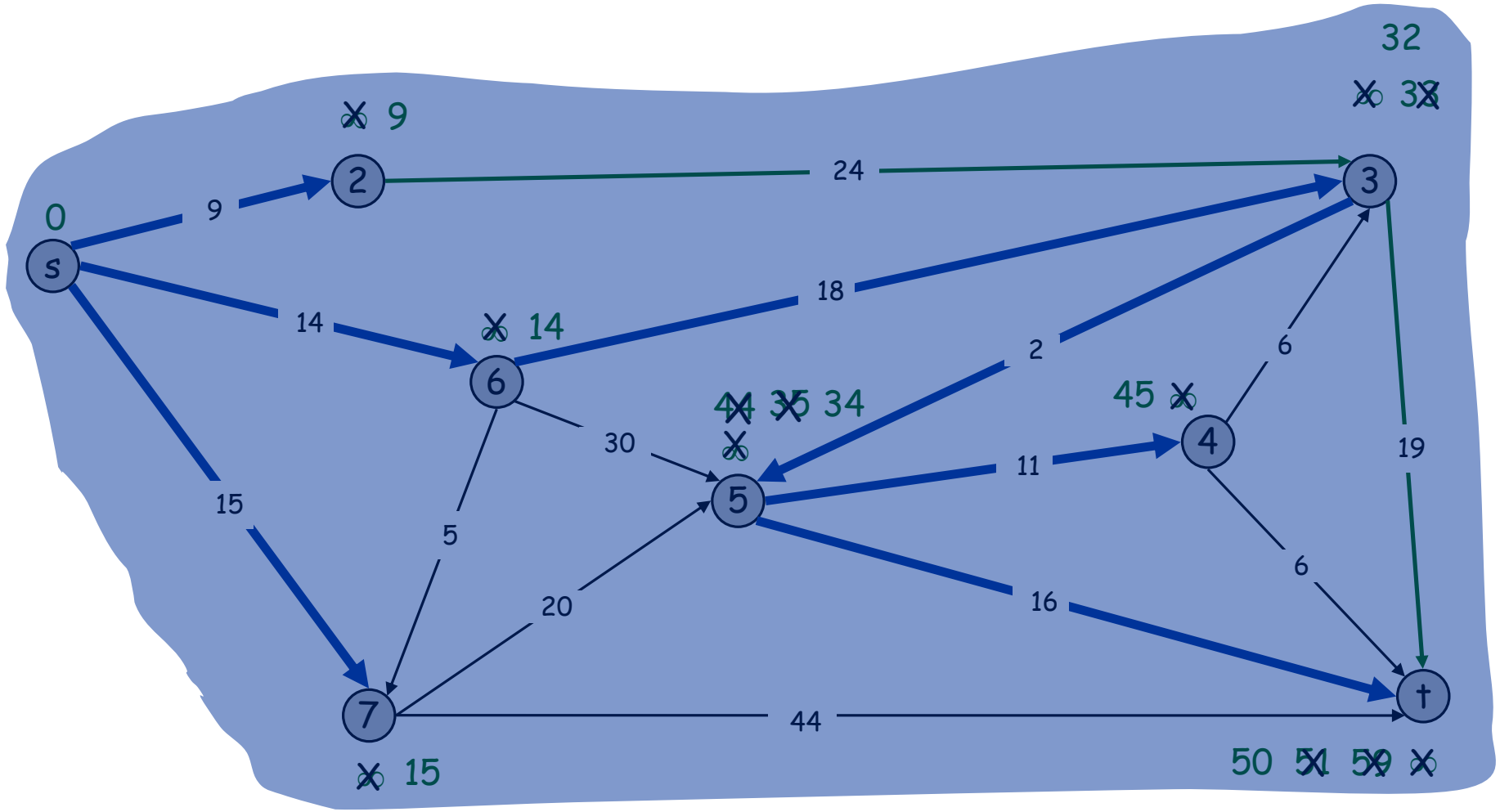
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 4, 5, 6, 7 }
PQ = { t }



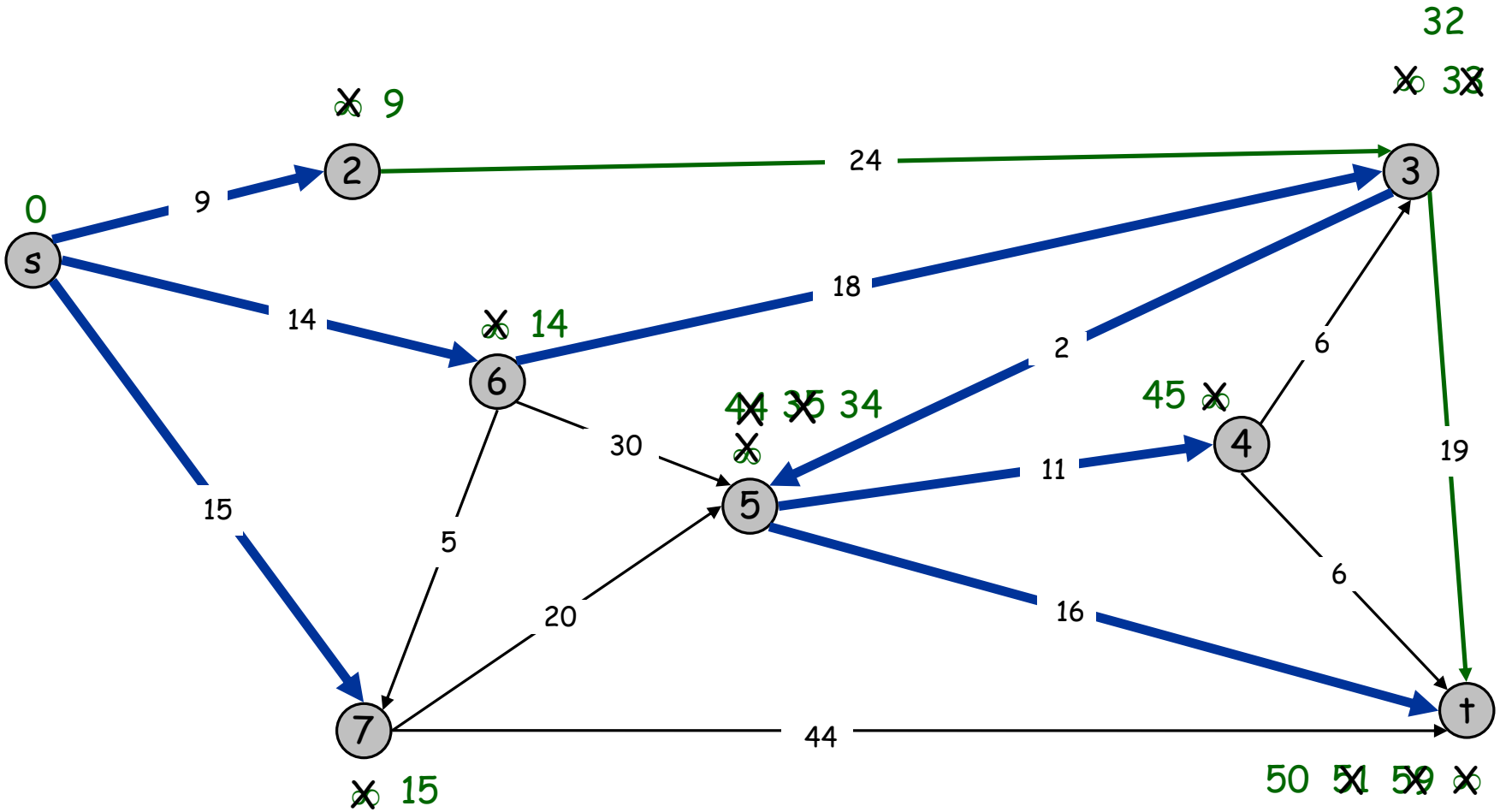
Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 4, 5, 6, 7, t }
PQ = { }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 4, 5, 6, 7, t }
PQ = { }



Summary

- Just as BFS finds the shortest path in unweighted graphs, Dijkstra's algorithm works for graphs with nonnegative edge weights
- Dijkstra's algorithm is a greedy algorithm that, in each iteration, explores the closest unexplored vertex
- Dijkstra's algorithm implemented using binary heaps takes O(mlogn) time
- Main Takeaway: Dijkstra's algorithm is a greedy approach that works for weighted shortest paths and heaps help speed up the implementation.