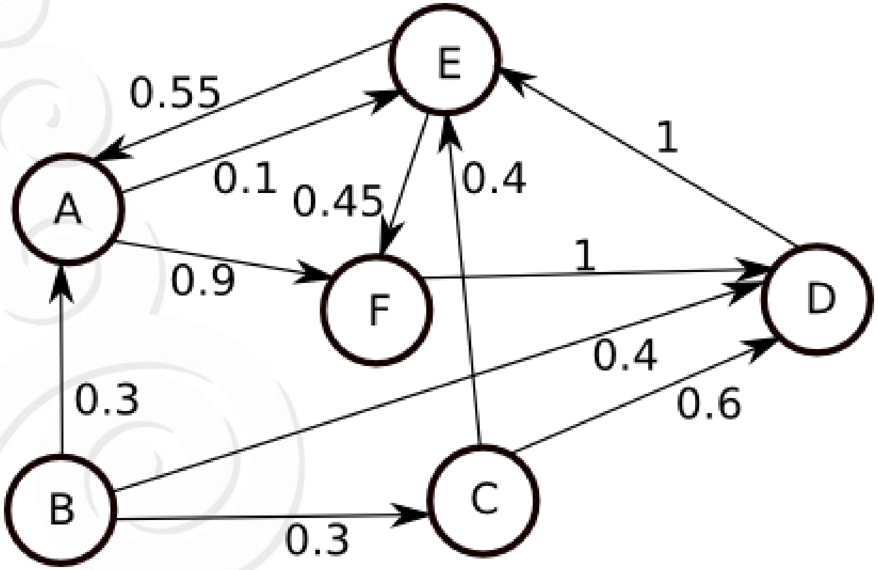
Weighted Graphs

Wahyono, Ph.D.
Department of Computer Science and Electronics
Faculty of Mathematics and Natural Sciences
Universitas Gadjah Mada, Yogyakarta, Indonesia
Email: wahyo@ugm.ac.id

Weighted Graphs

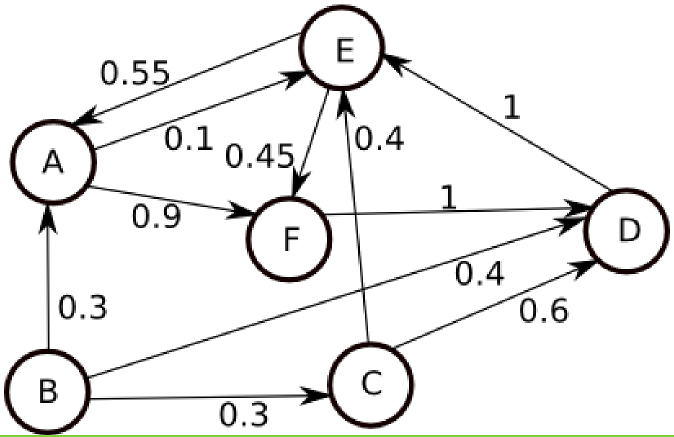
- Once again, a graph where edges have weights, which quantifies the relationship
- These graphs can be directed or undirected



Weighted Graph: Adjacency List

- The adjacency list for a weighted graph contains edge weights
  - Instead of 0 and 1
- If there is no edge connecting vertices i and j, a weight of INFINITY is used (not 0!)
  - Because '0' can also be a weight
  - Also most applications of weighted graphs are to find minimum spanning trees or shortest path (we'll look at this)
- Also remember if the graph is undirected, redundant information should be stored

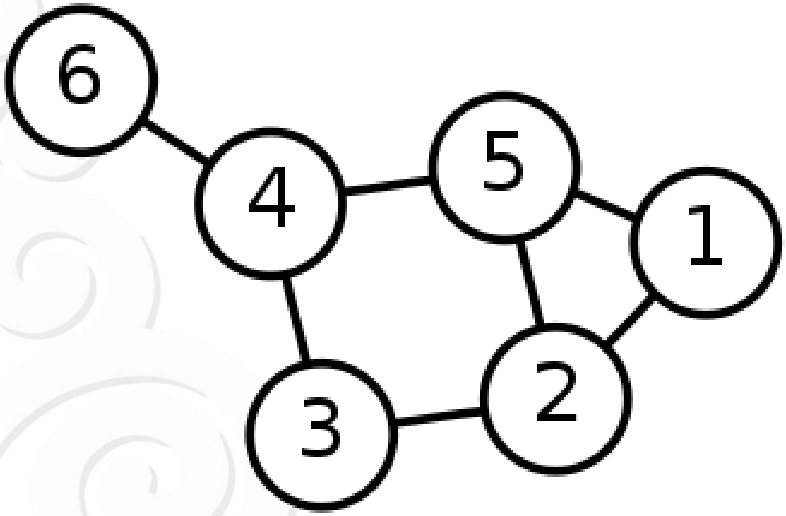
Weighted Graph: Adjacency List



	A	B	C	D	E	F
A	INF	INF	INF	INF	0.1	0.9
B	0.3	INF	0.3	0.4	INF	INF
C	INF	INF	INF	0.6	0.4	INF
D	INF	INF	INF	INF	1	INF
E	0.55	INF	INF	INF	INF	0.45
F	INF	INF	INF	1	INF	INF

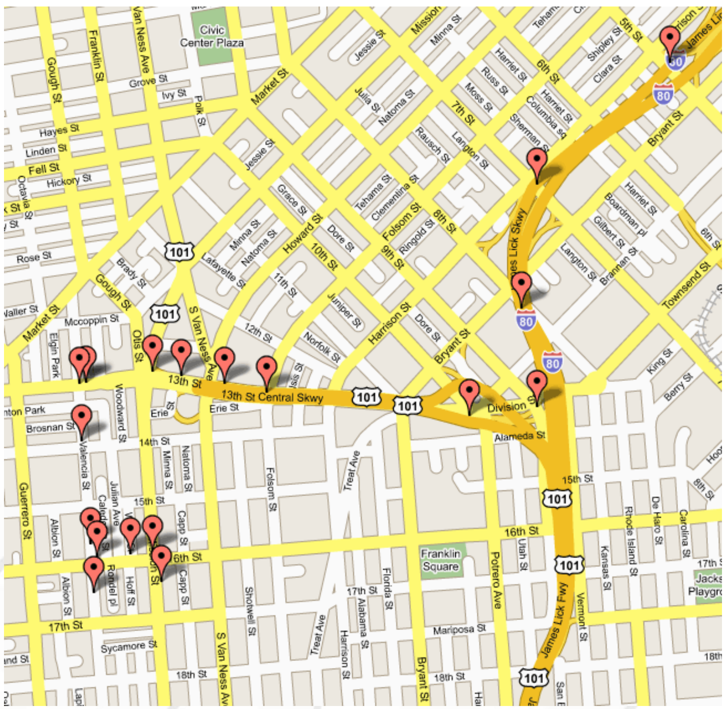
Shortest Path Problem

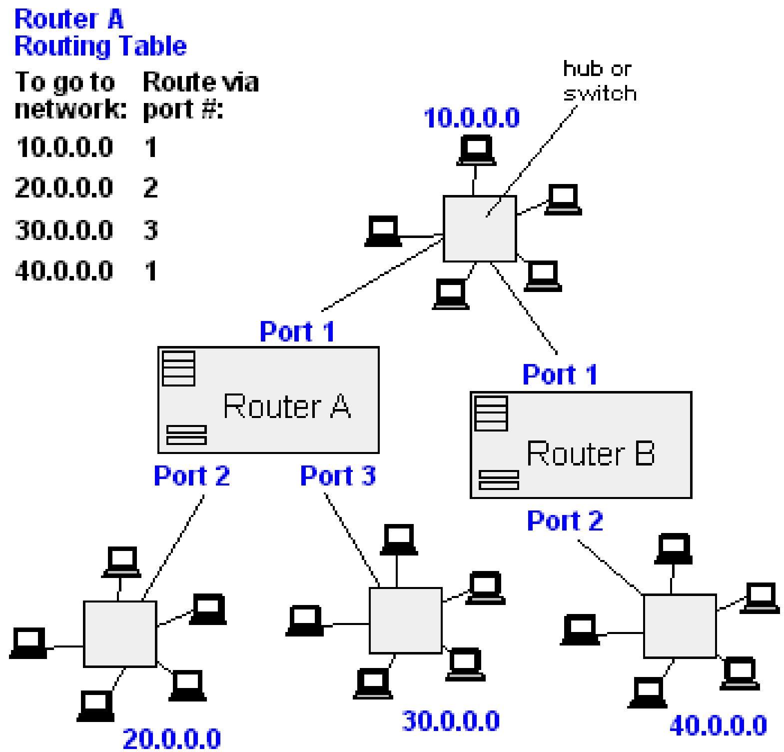
Shortest Path Problem - The problem of finding shortest paths from a source vertex *v* to all other vertices in the graph.



Applications

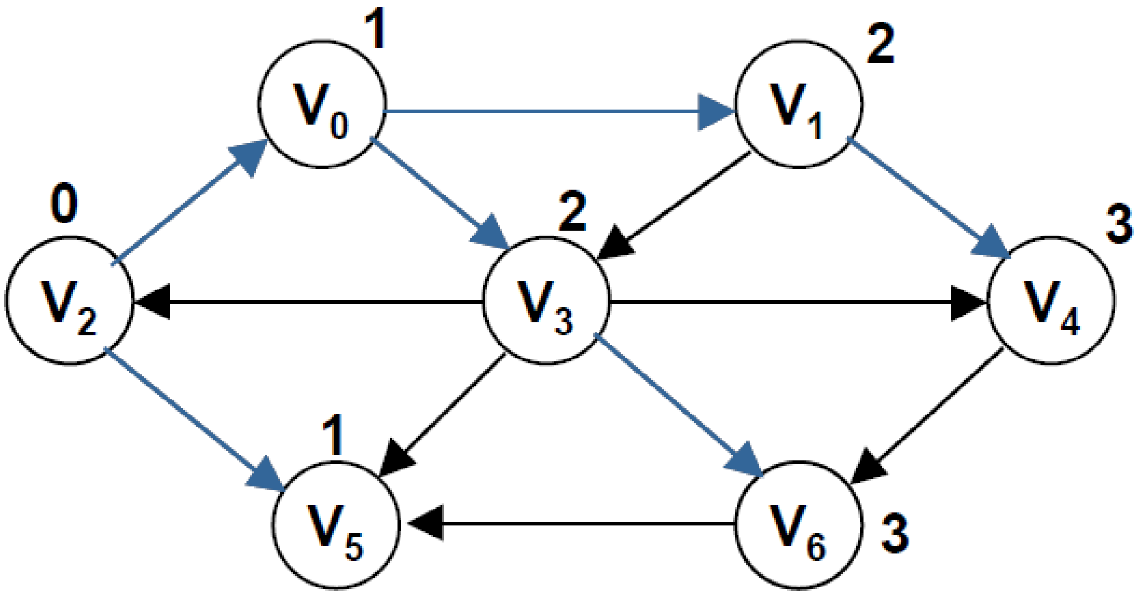
- Maps (Map Quest, Google Maps)
- Routing Systems





Shortest Path: unweighted graph

- **Starting vertex: V_2**
- **Use BFS (Breath First Search) instead of DFS (Depth First Search) to find shortest path in unweighted graph (or each edge have the same weight)**

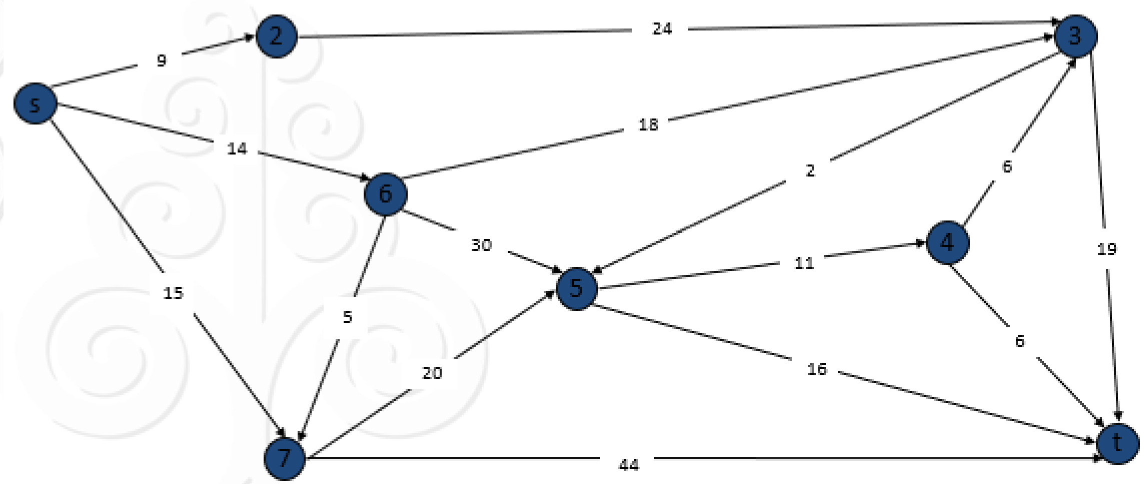


Shortest Path: positive weighted graph

- In many applications, e.g., transportation networks, the edges of a graph have different weights.
- Dijkstra's algorithm finds shortest paths from a start vertex *s* to all the other vertices in a graph with
  - non-negative edge weights
- Dijkstra's algorithm uses a greedy method

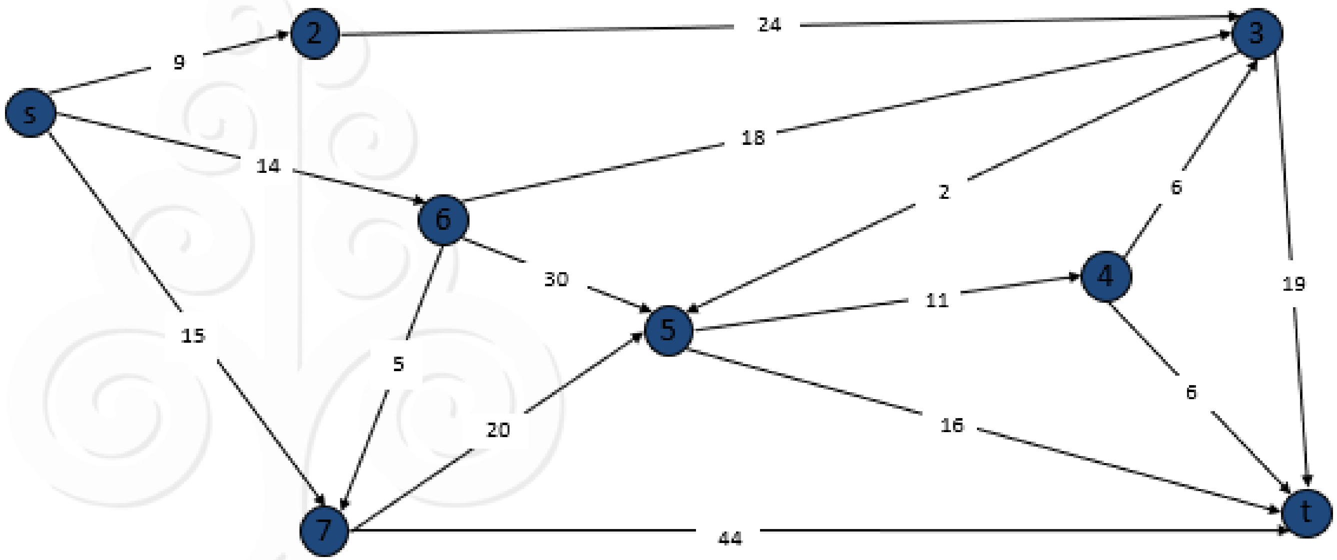
Dijkstra’s Algorithm

- Given a weighted graph, find the shortest path (in terms of edge weights) between two vertices in the graph
- Numerous applications
  - Cheapest airline fare between departure and arrival cities
  - Shortest driving distance in terms of mileage



Dijkstra's Algorithm

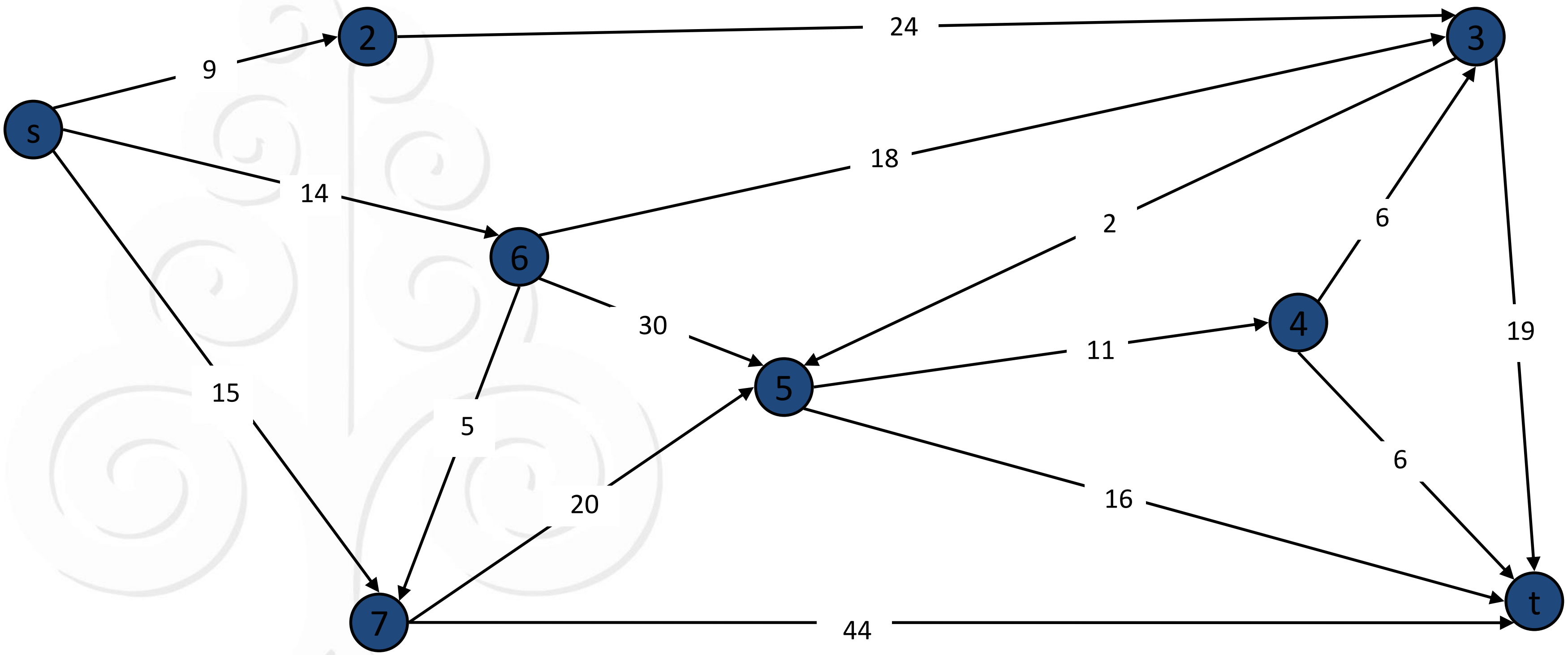
- Suppose in the graph below, we wanted the shortest path from s to t



- Idea: Maintain a table of the current shortest paths from s to all other vertices (and the route it takes)
  - When finished, the table will hold the shortest path from B to all other vertices

Dijkstra's Shortest Path Algorithm

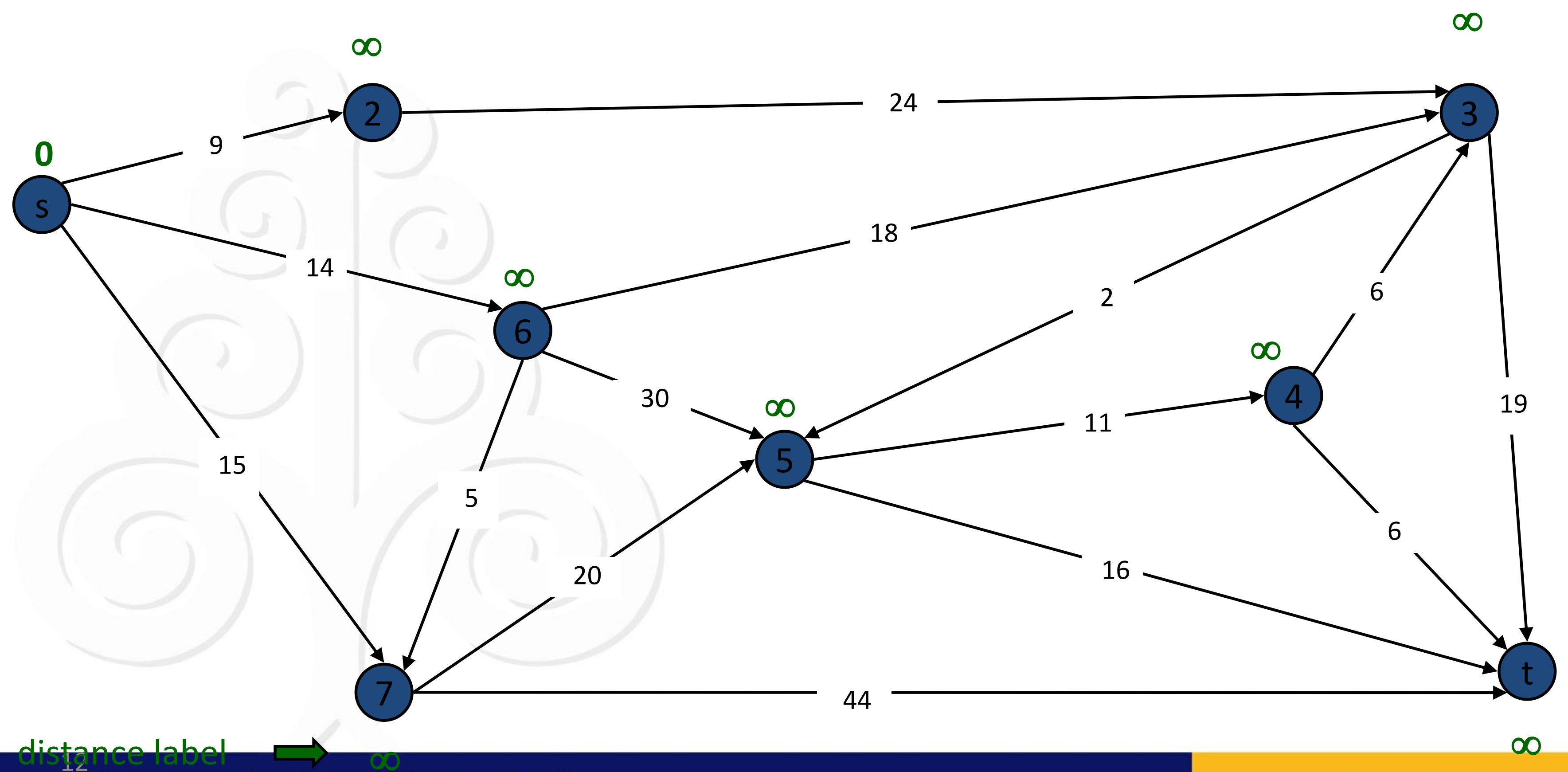
- Find shortest path from s to t.



Dijkstra's Shortest Path Algorithm

S = { }

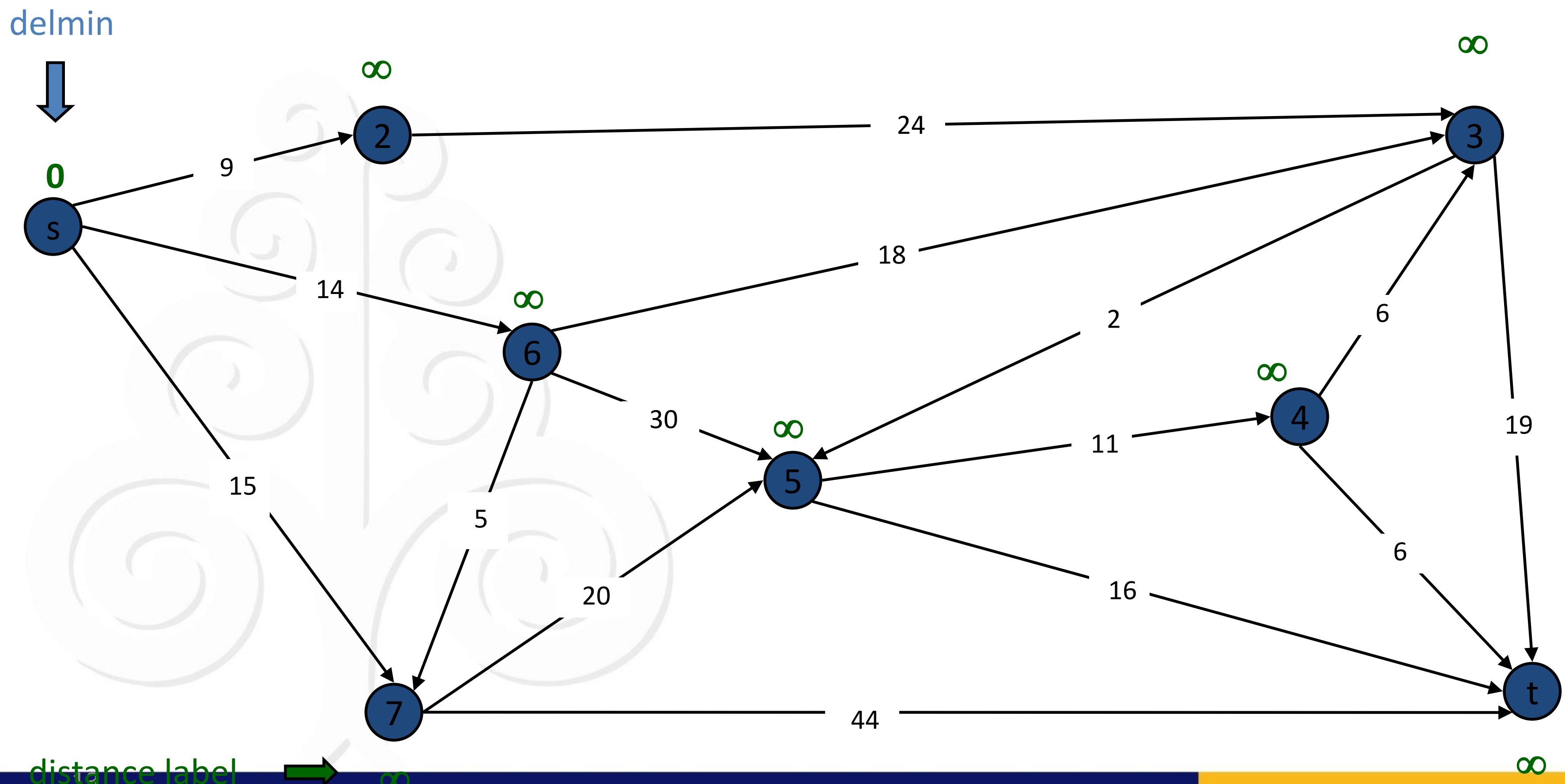
PQ = { s, 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { }

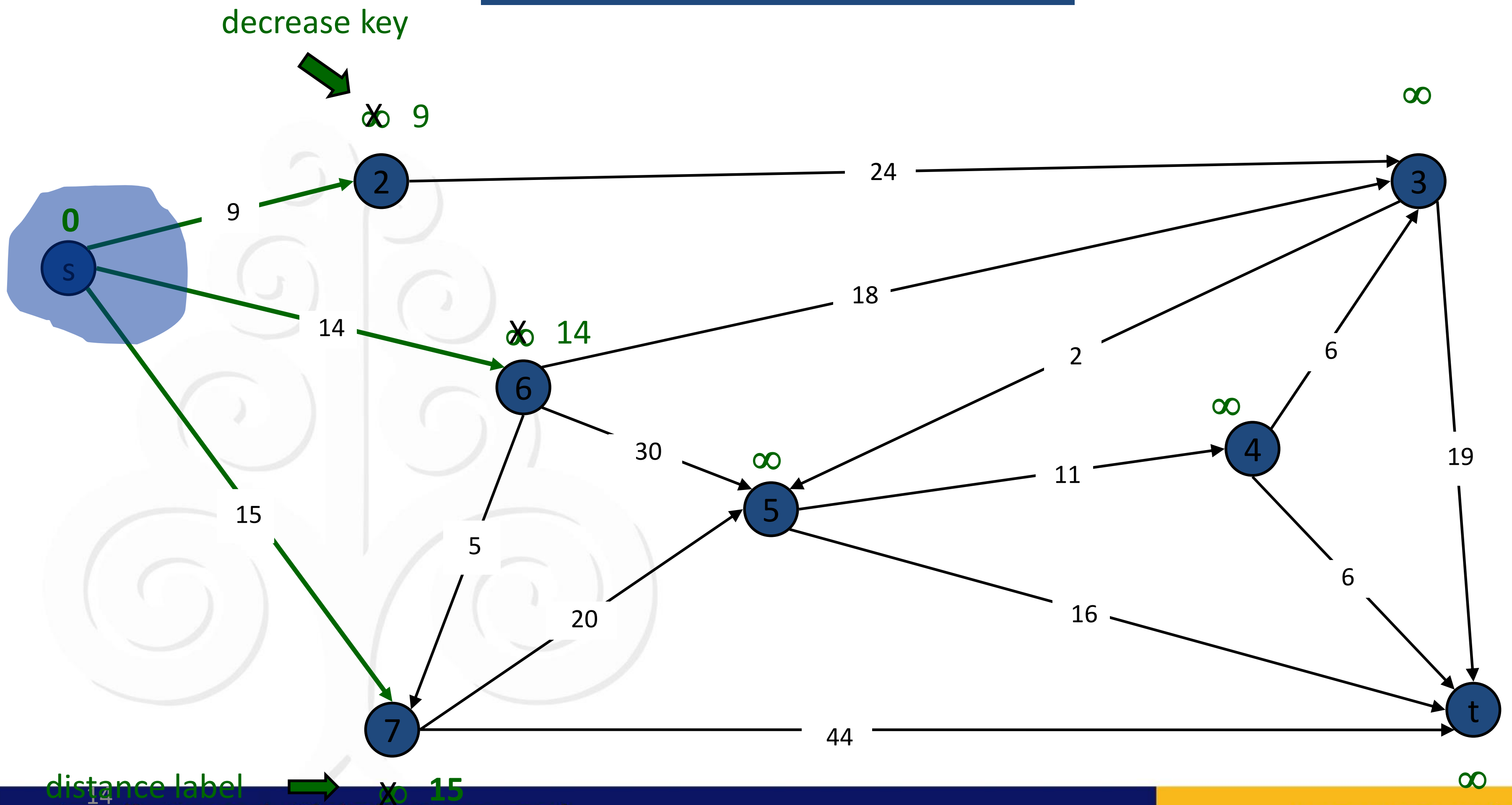
PQ = { s, 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s }

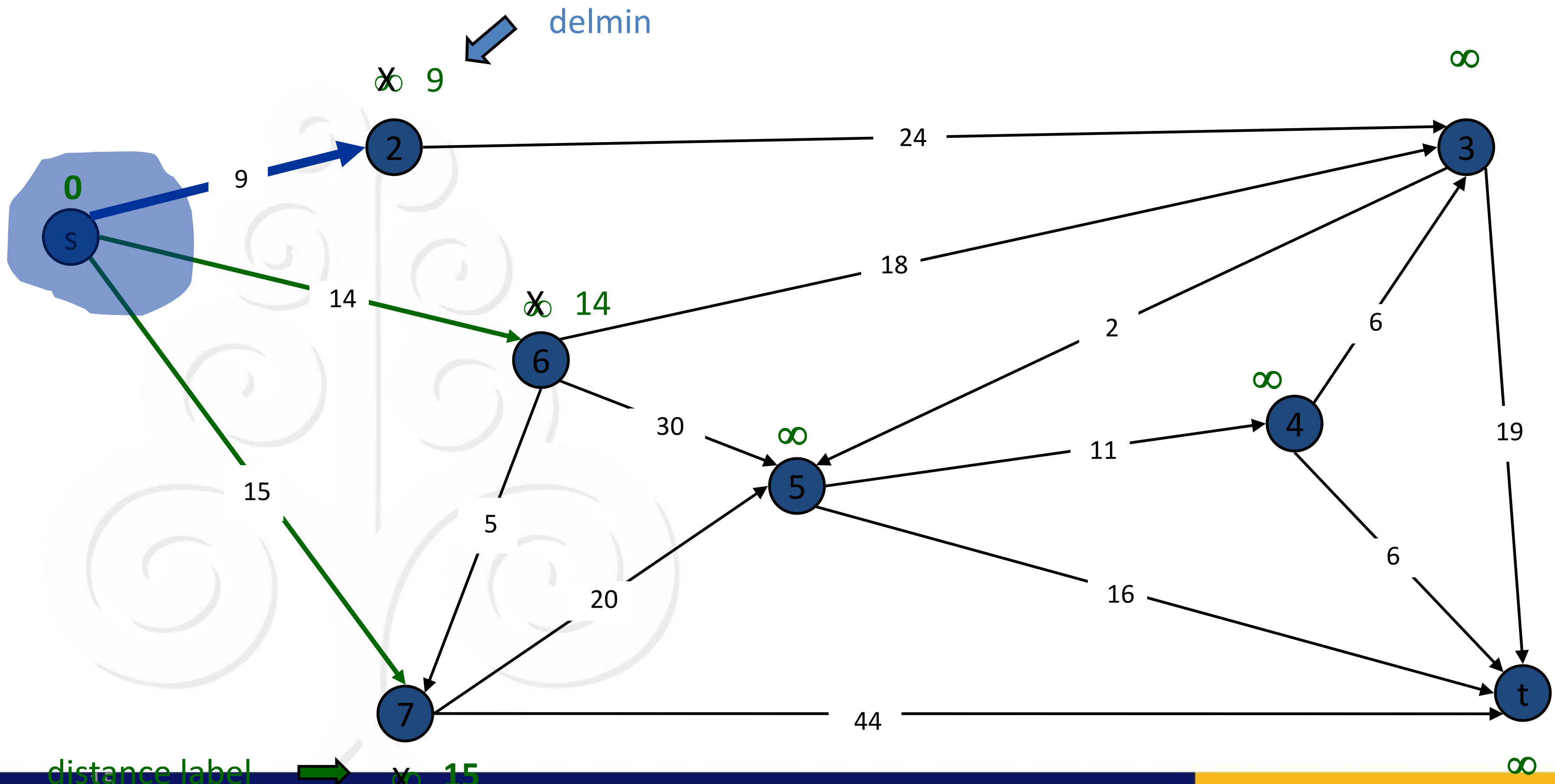
PQ = { 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s }

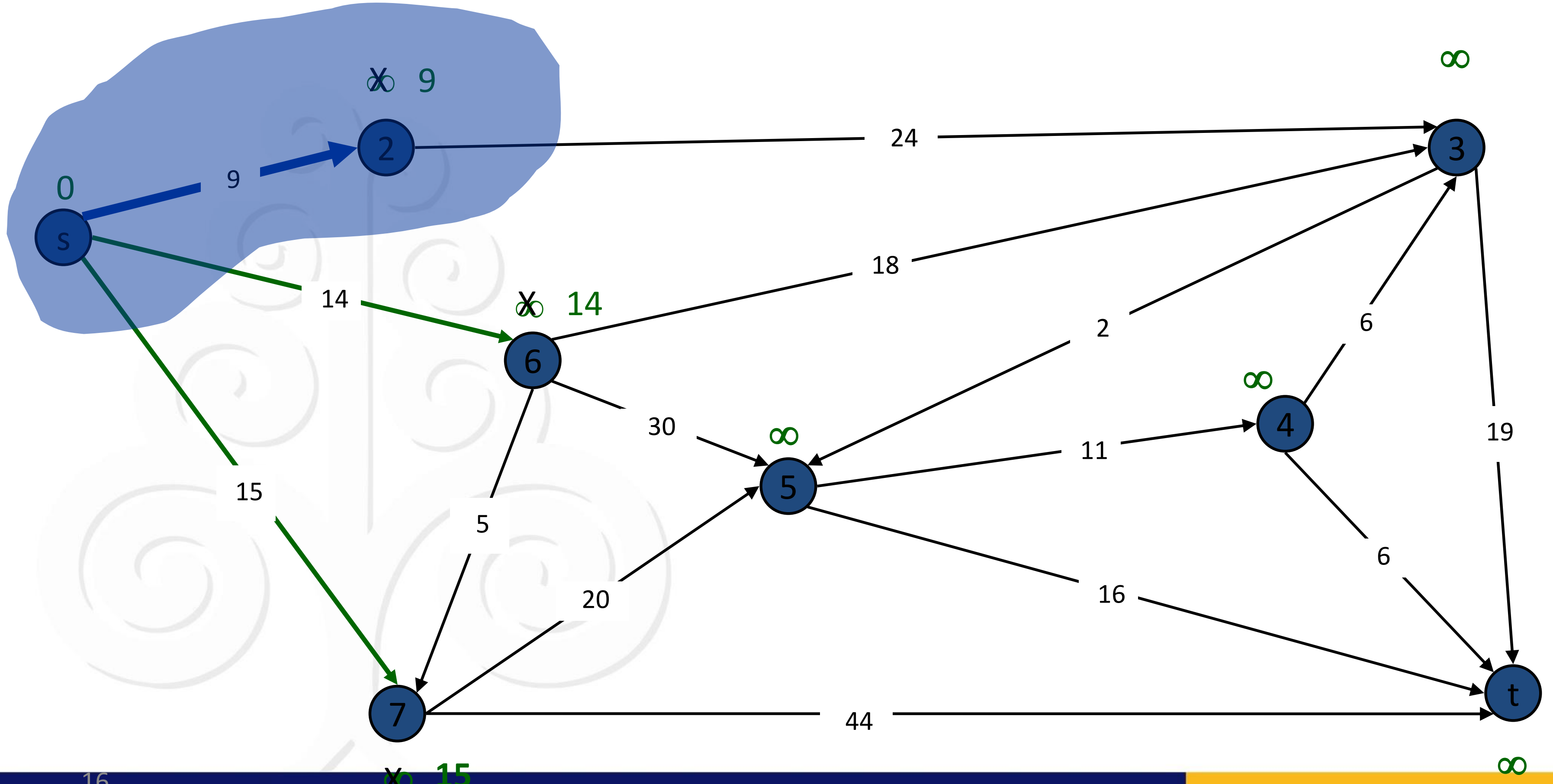
PQ = { 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2 }

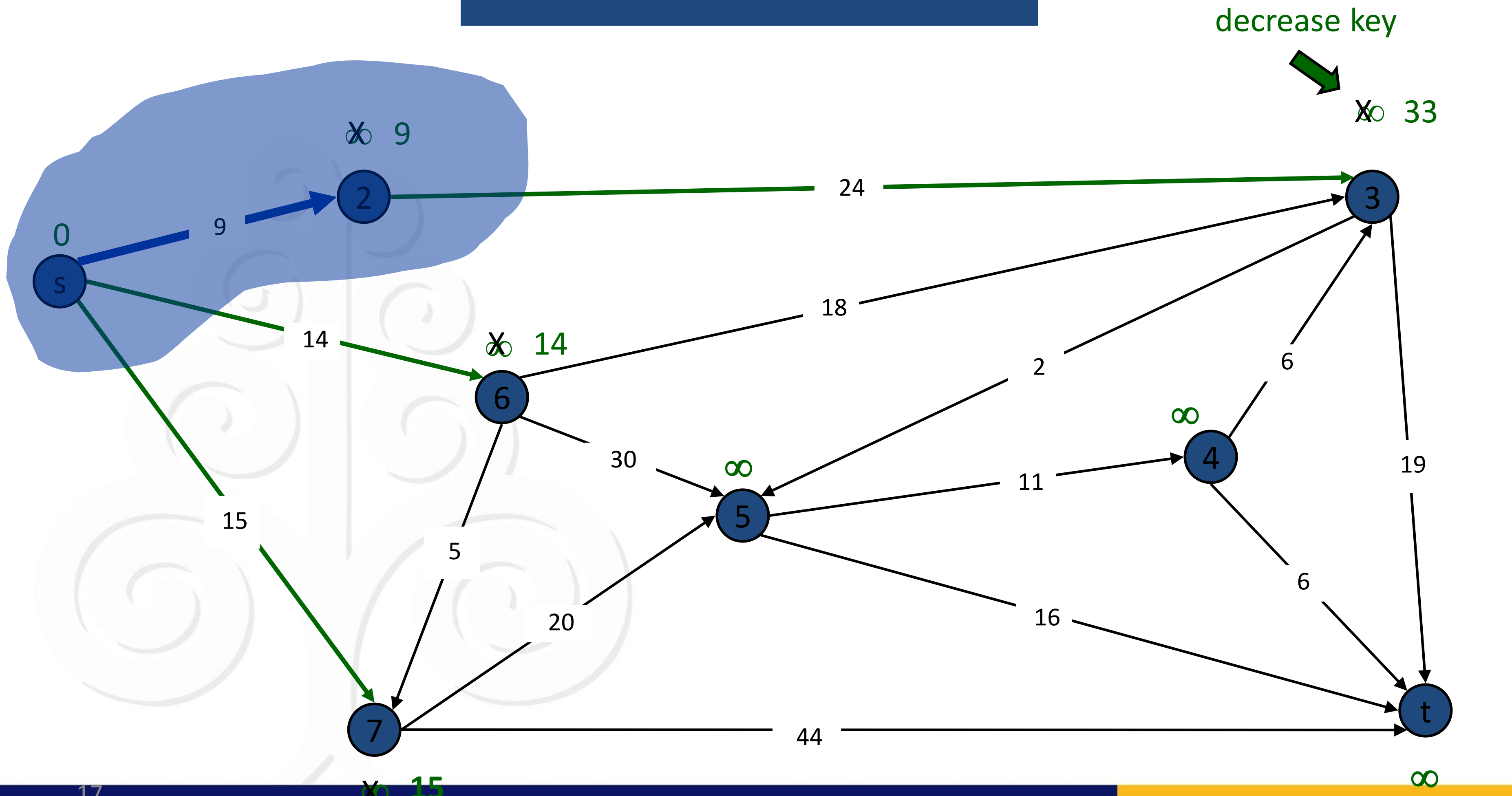
PQ = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2 }

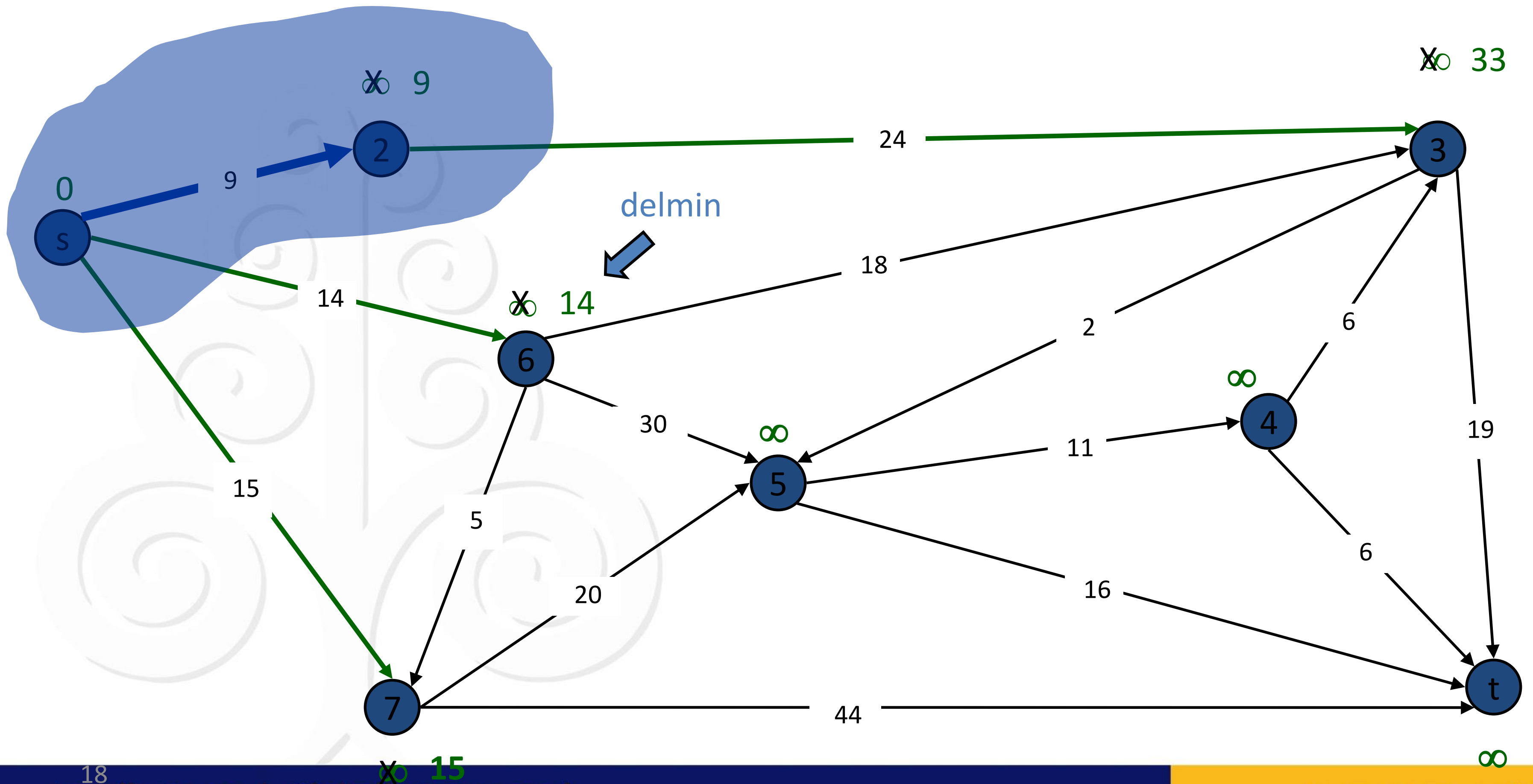
PQ = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2 }

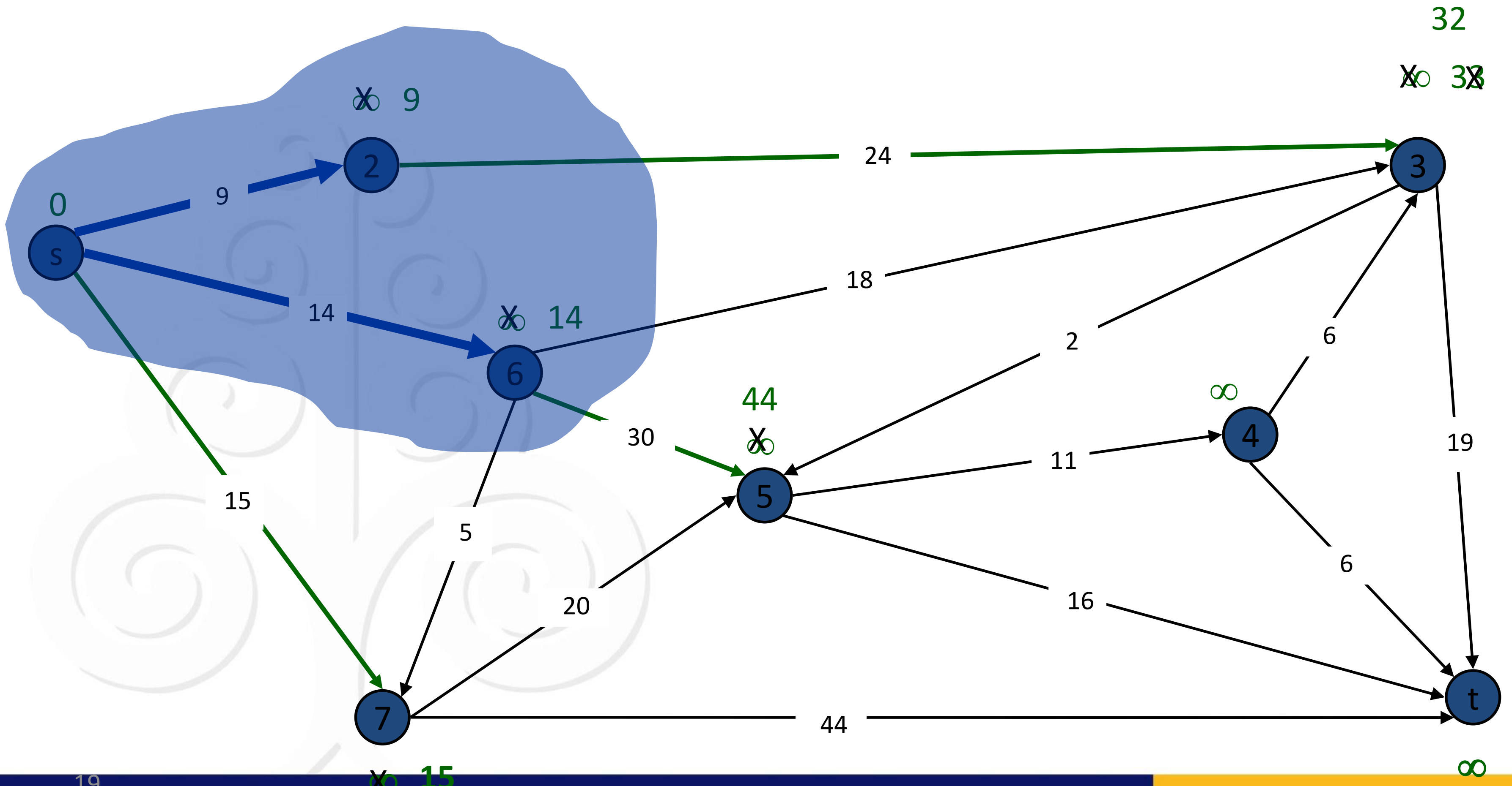
PQ = { 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 6 }

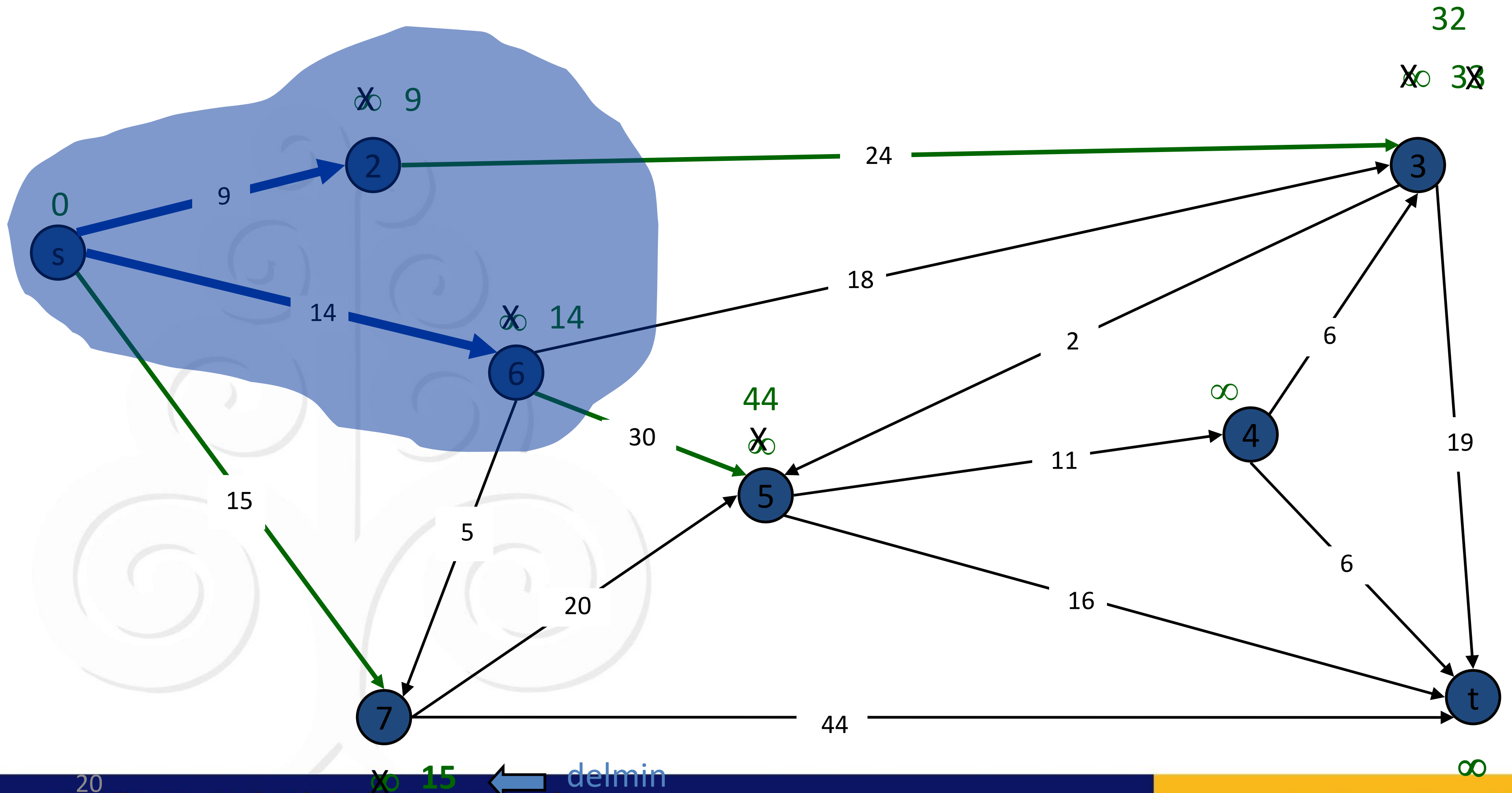
PQ = { 3, 4, 5, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 6 }

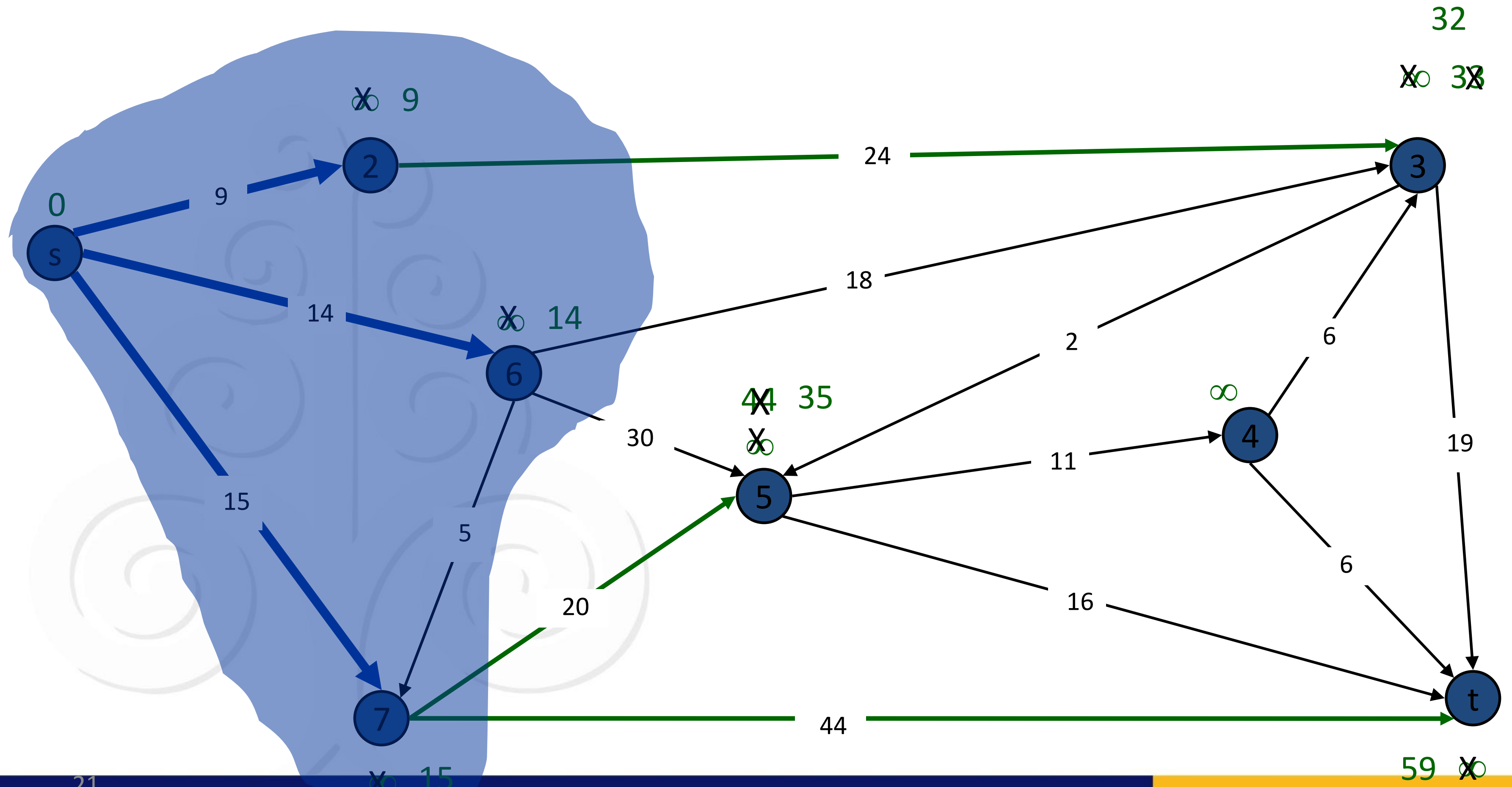
PQ = { 3, 4, 5, 7, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 6, 7 }

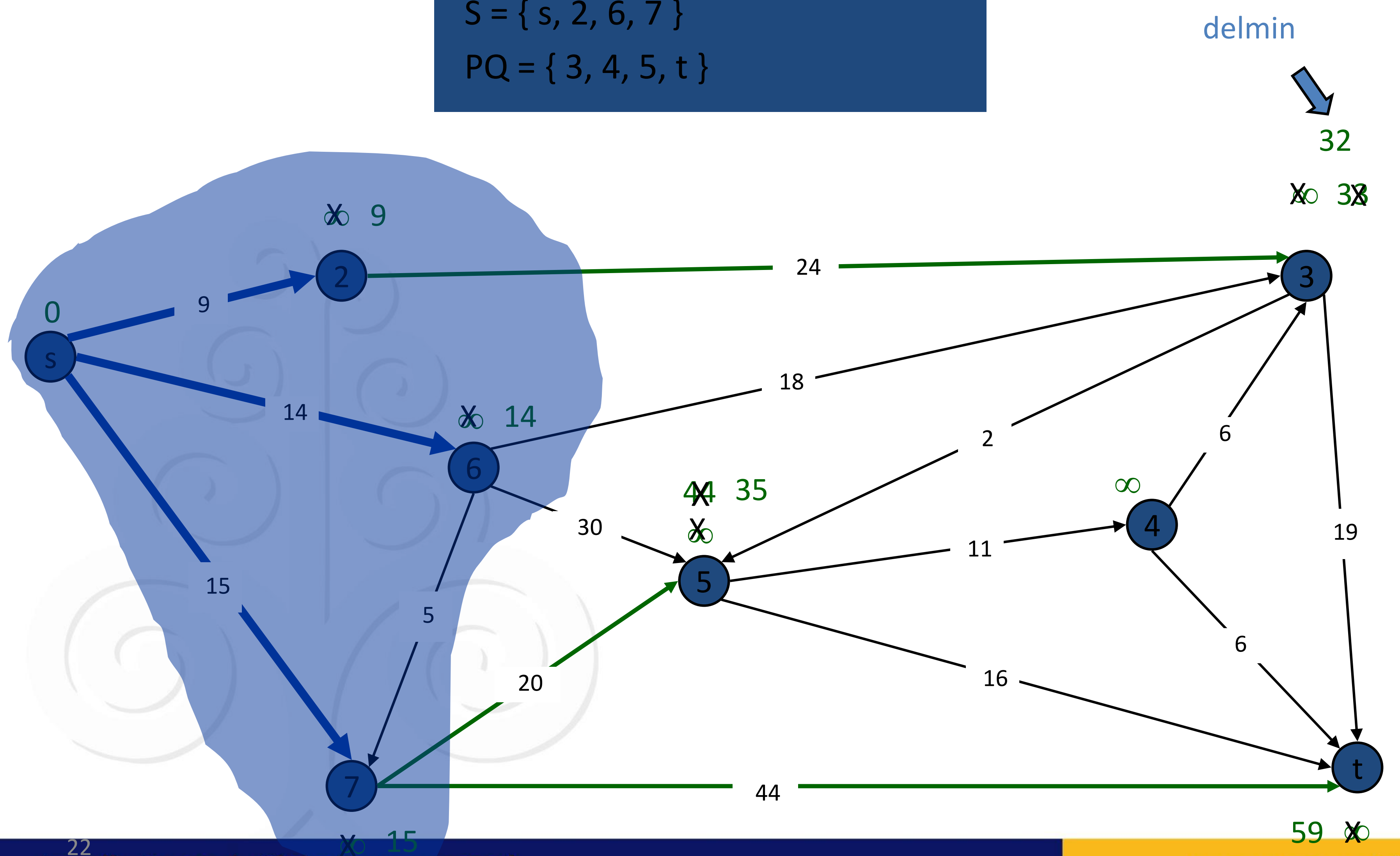
PQ = { 3, 4, 5, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 6, 7 }

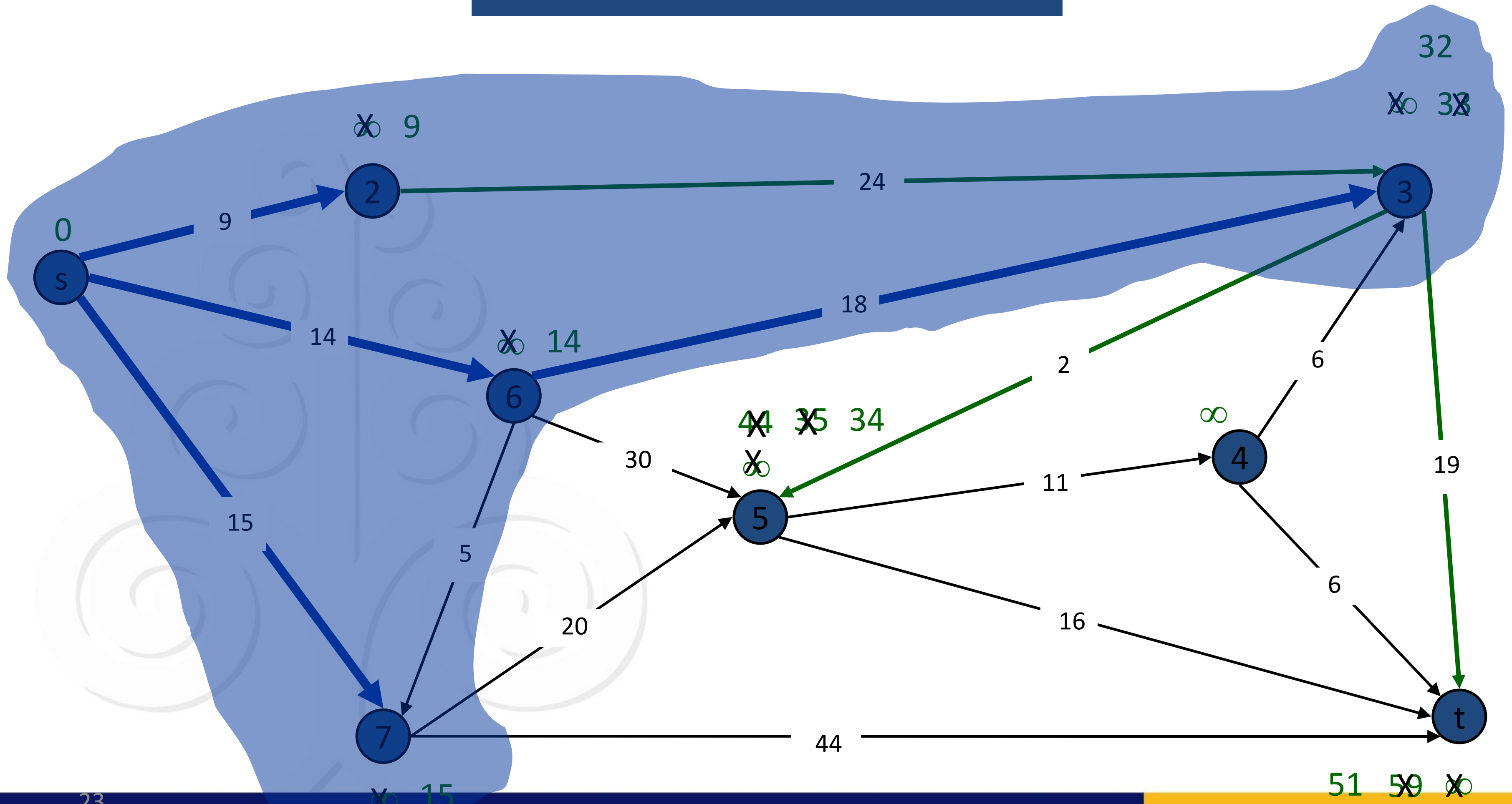
PQ = { 3, 4, 5, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 6, 7 }

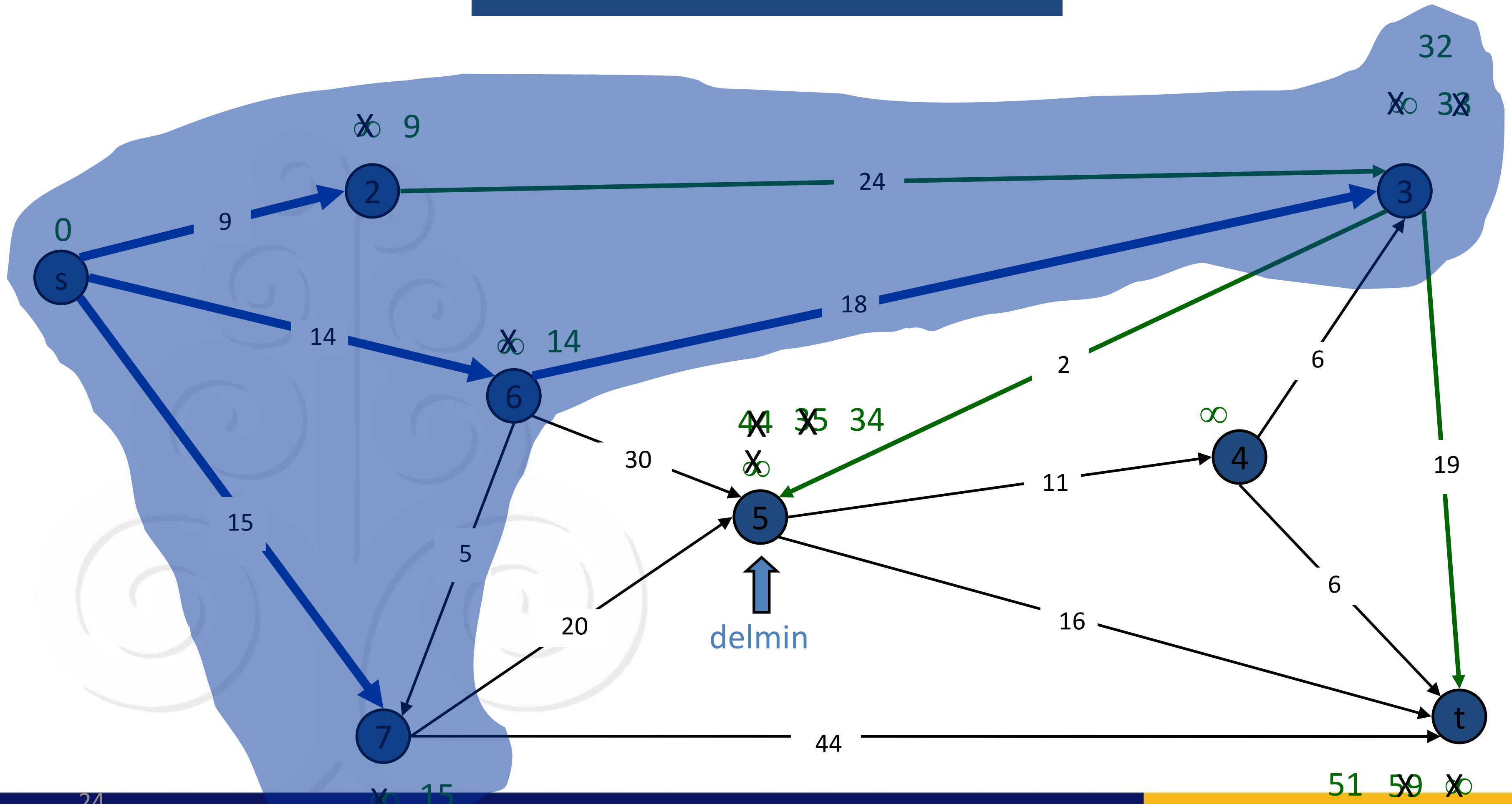
PQ = { 4, 5, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 6, 7 }

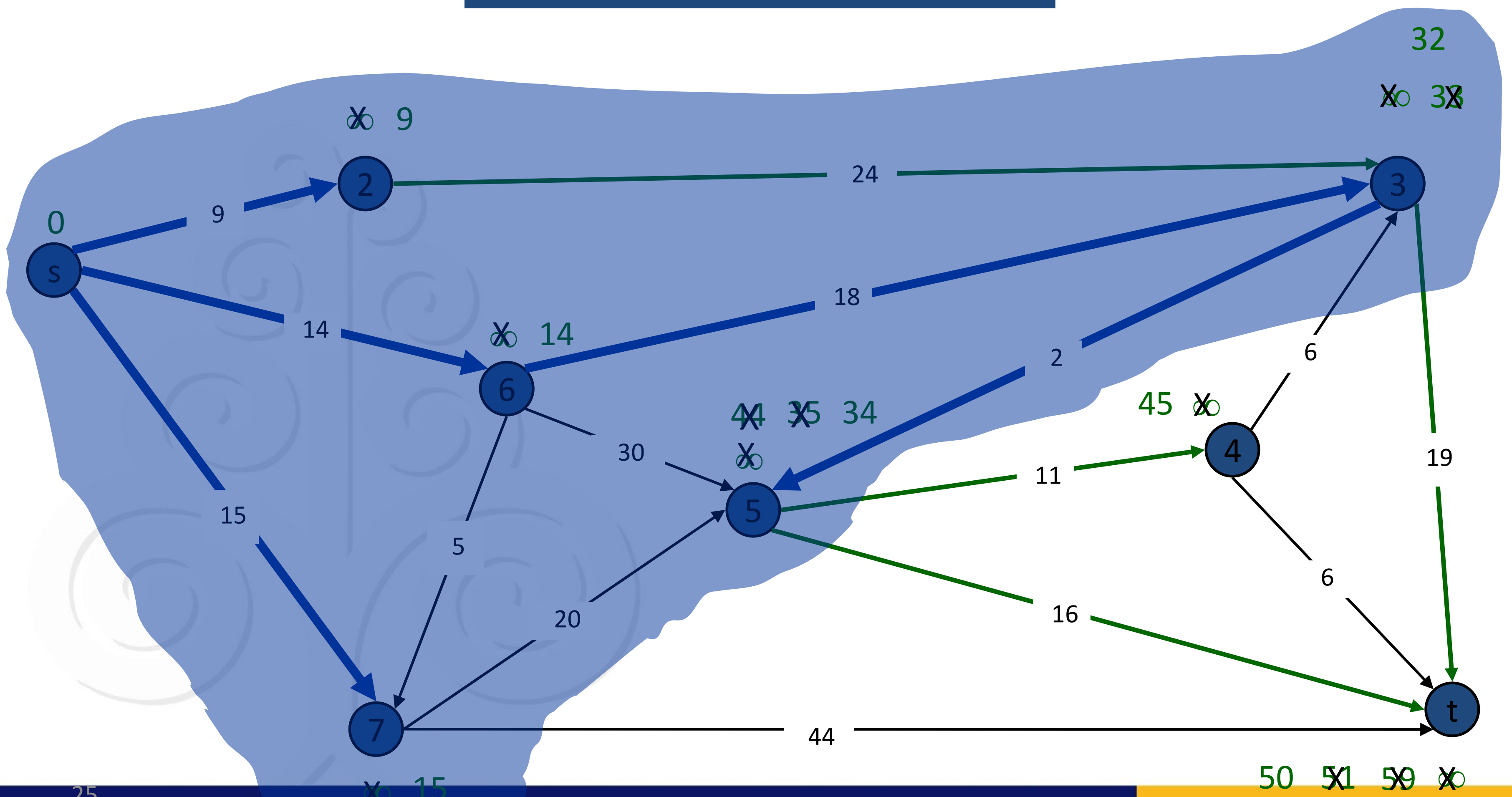
PQ = { 4, 5, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 5, 6, 7 }

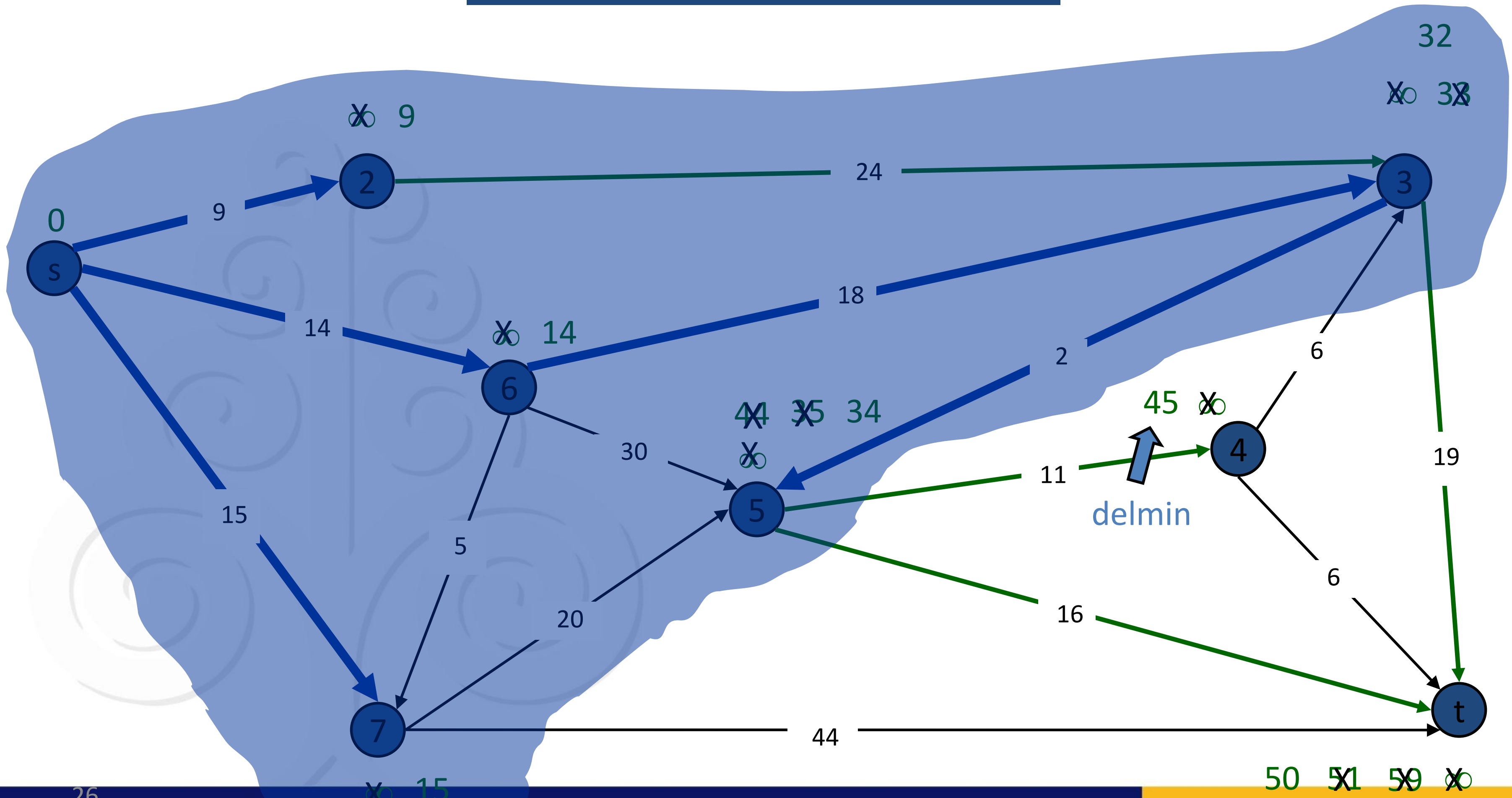
PQ = { 4, t }



Dijkstra's Shortest Path Algorithm

S = { s, 2, 3, 5, 6, 7 }

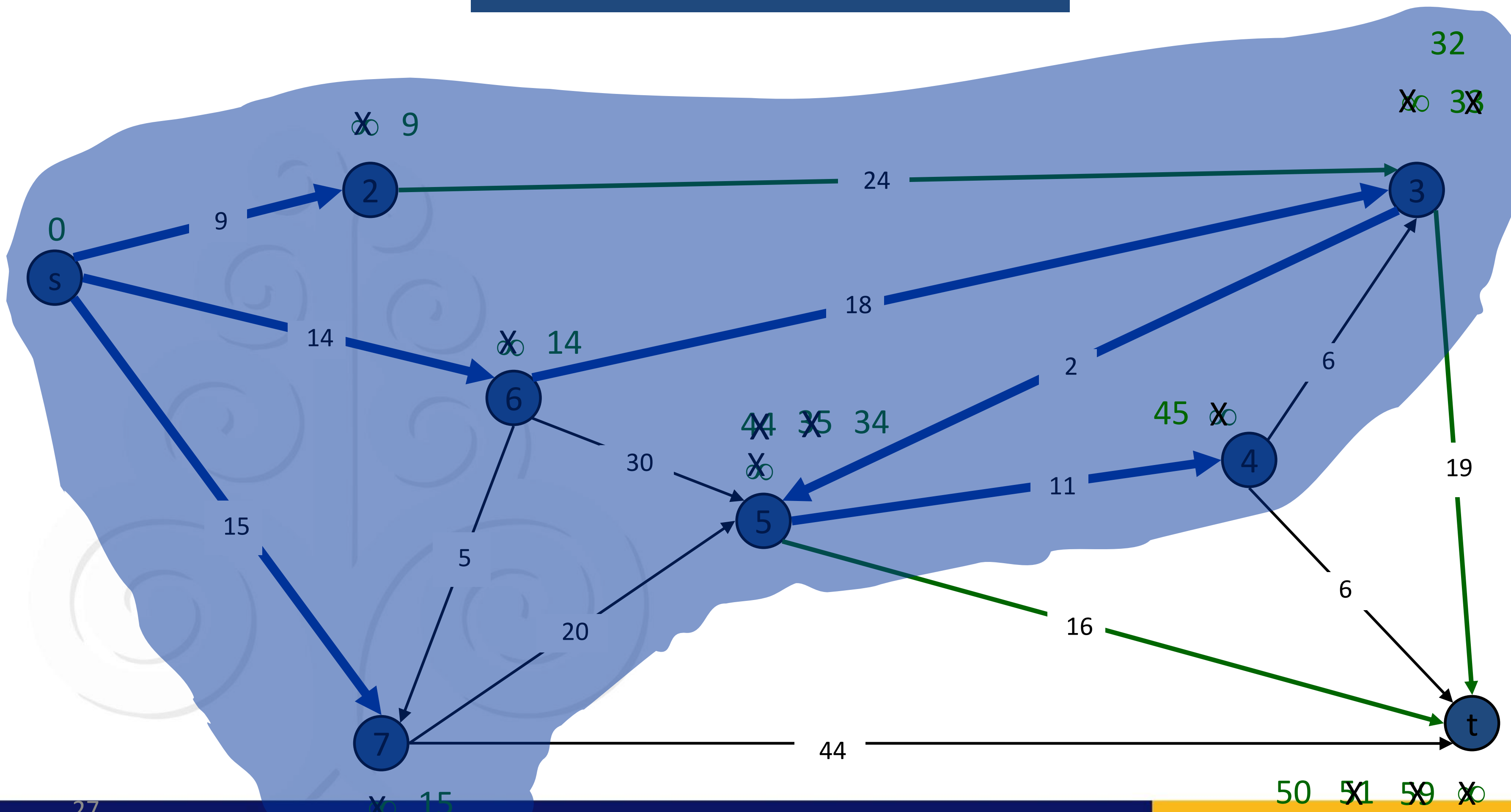
PQ = { 4, t }



Dijkstra's Shortest Path Algorithm

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

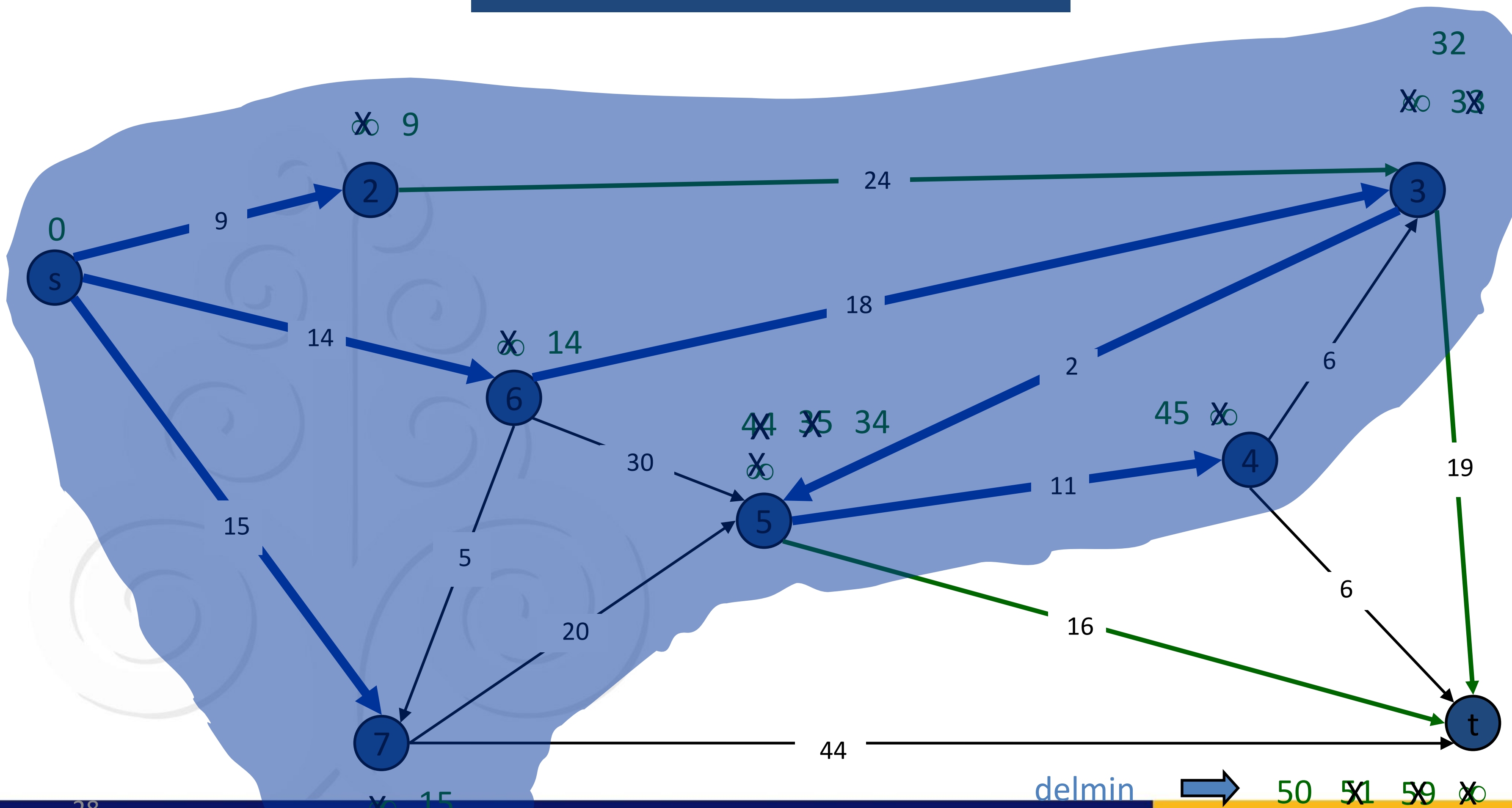
PQ = { t }



Dijkstra's Shortest Path Algorithm

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

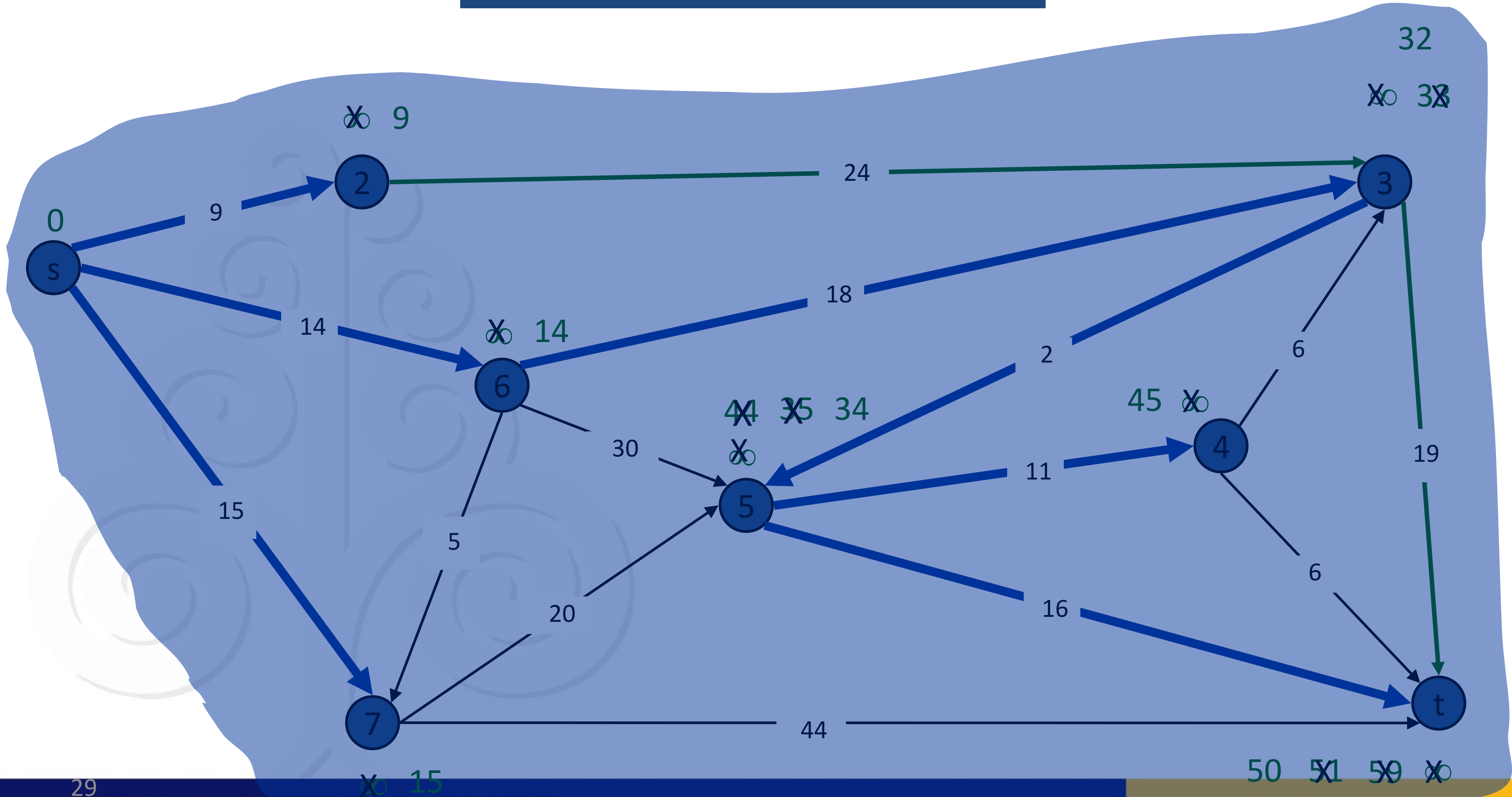
PQ = { t }



Dijkstra's Shortest Path Algorithm

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

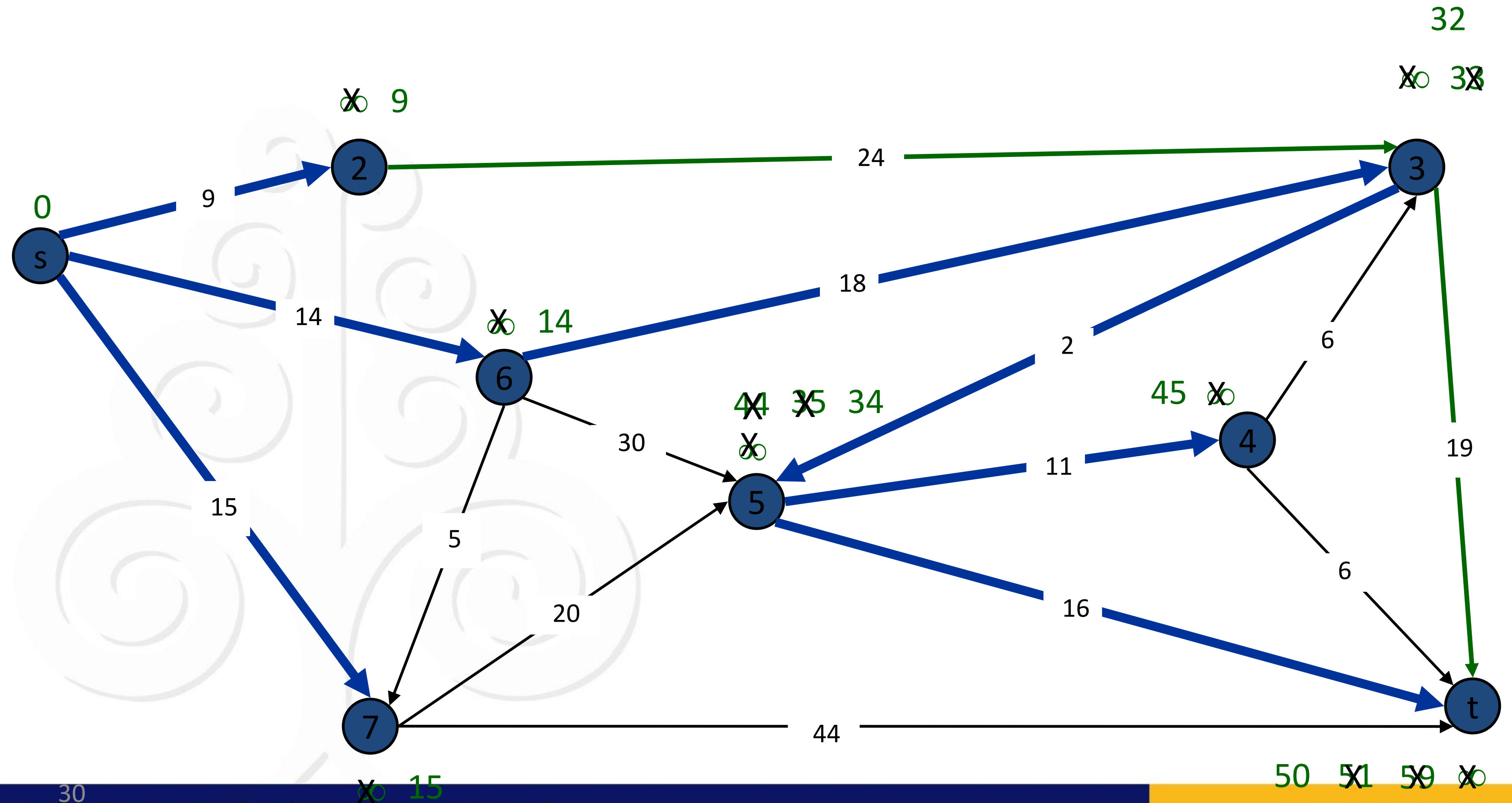
PQ = { }



Dijkstra's Shortest Path Algorithm

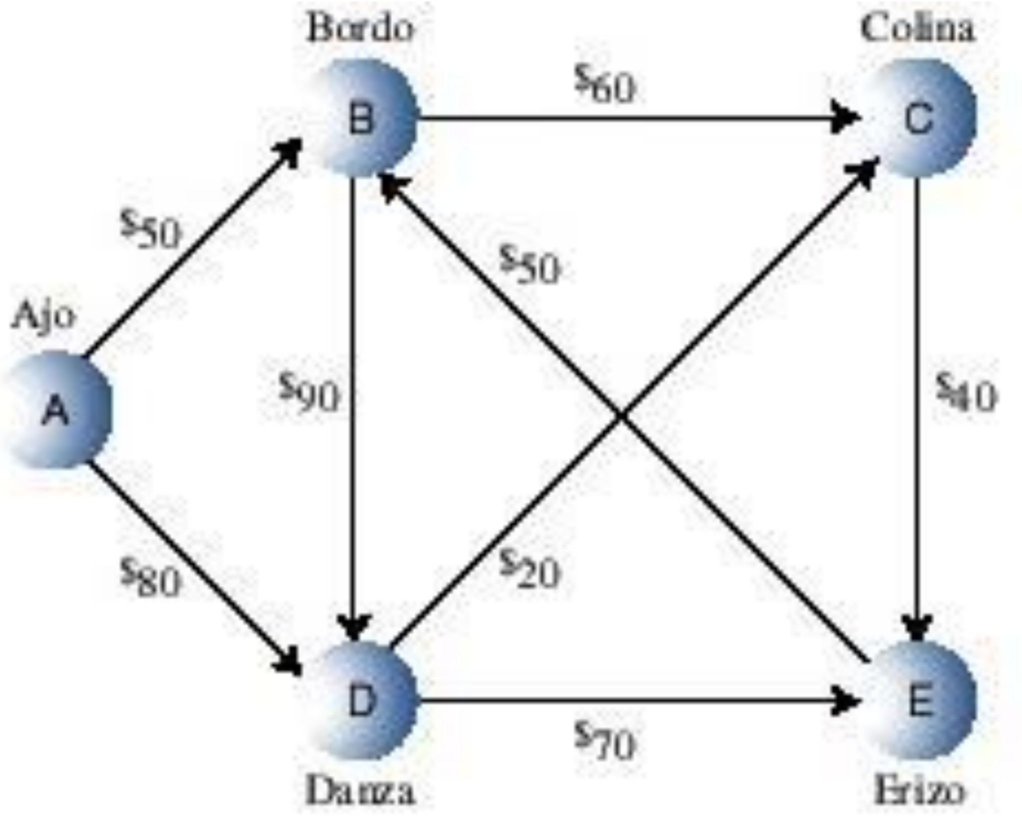
S = { s, 2, 3, 4, 5, 6, 7, t }

PQ = { }

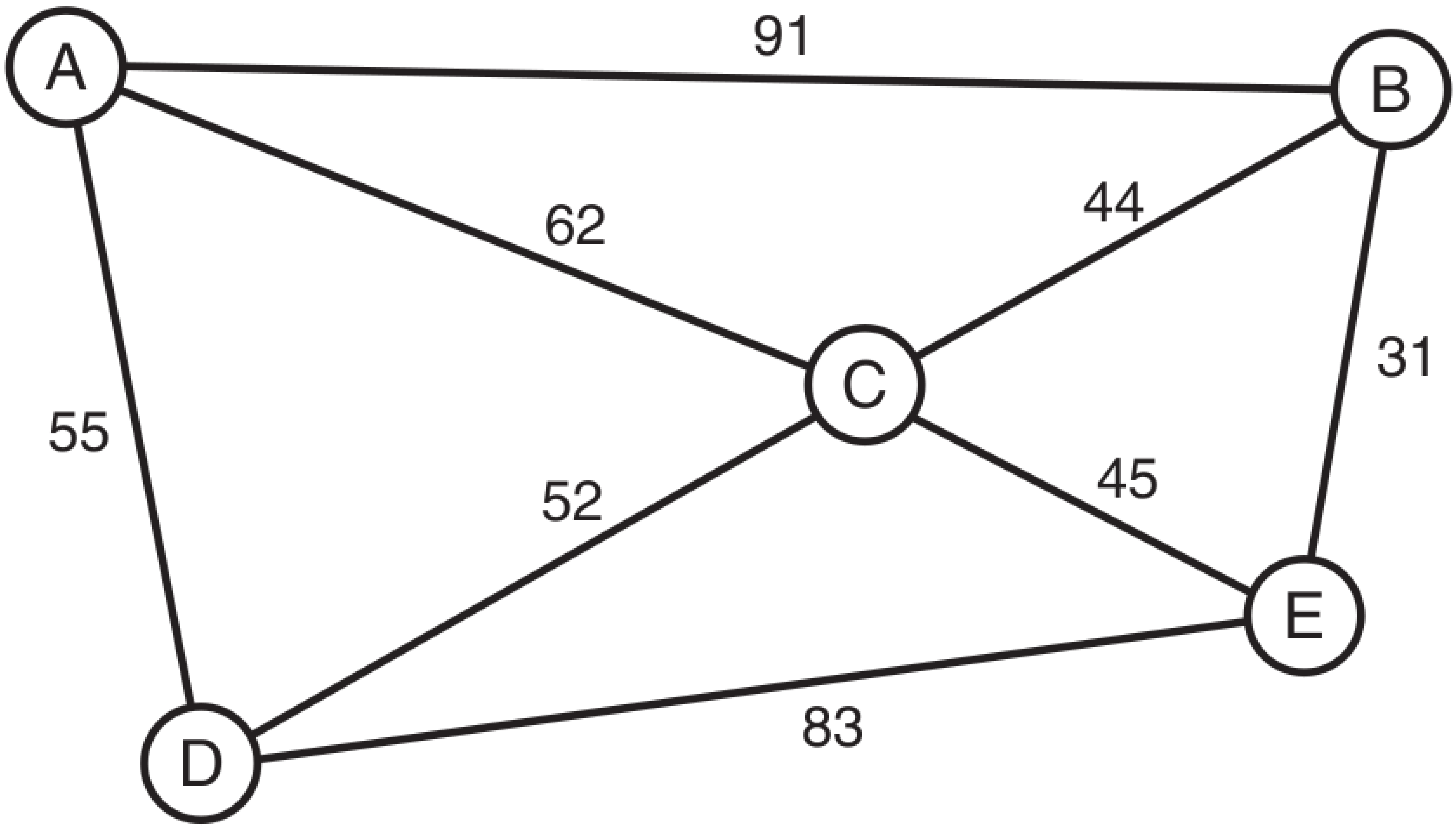


Example

- Let's try this example with train costs:



Find the shortest path from D to all other vertices!



THANK YOU

Mengakar Kuat
Menjulang
Tinggi