Objectives

- Wrap up: Weighted, directed graph shortest path
- Minimum Spanning Tree

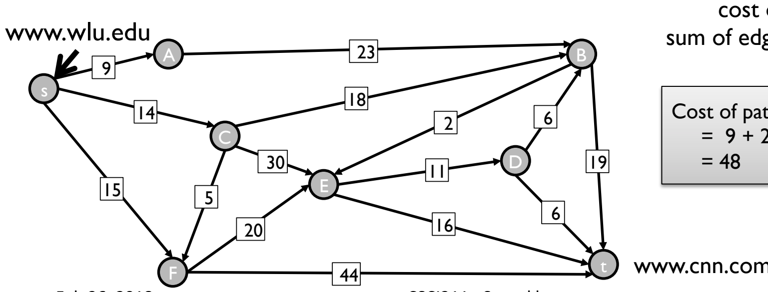
Review

- What are greedy algorithms?

Review: Shortest Path Problem

What was our strategy?

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



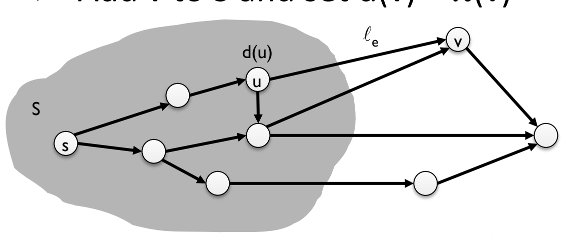
cost of path = sum of edge costs in path

Cost of path s-A-B-E-t
= 9 + 23 + 2 + 16
= 48

Review: Dijkstra's Algorithm

1. Maintain a set of **explored nodes** S
   - ➢ Keep the shortest path distance d(u) from *s* to *u*
2. Initialize S={s}, d(s)=0, ∀ u≠s, d(u)=∞
3. Repeatedly choose unexplored node *v* which minimizes $\pi(v) = \min_{e=(u,v)\,:\,u \in S} d(u) + \ell_e,$
   - ➢ Add v to S and set d(v) = π(v)

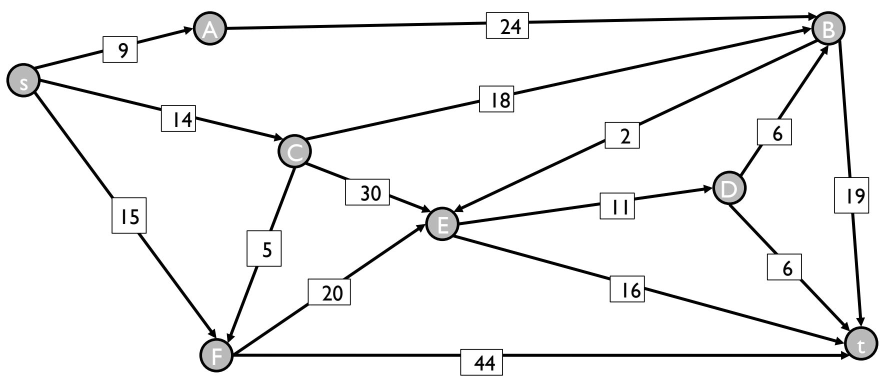
shortest path to (some u in explored part followed by a single edge (u, v))



Implementation Ideas
- What to represent?
- How to represent?

Dijkstra’s Shortest Path Algorithm

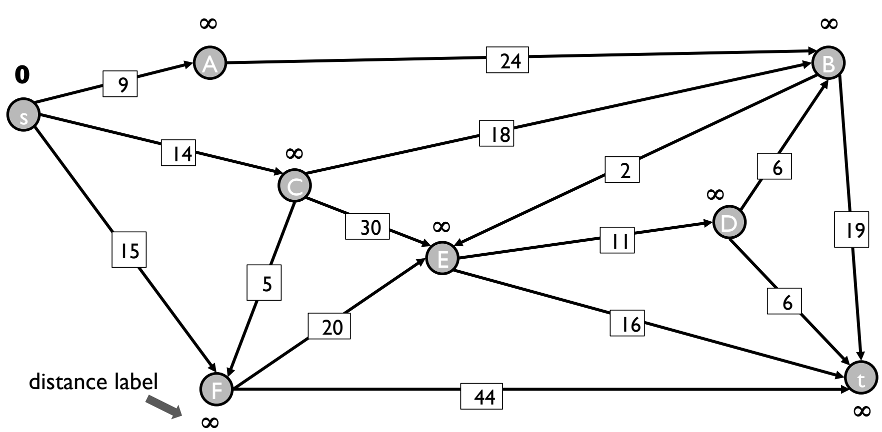
- Find shortest path from s to t



Dijkstra’s Shortest Path Algorithm

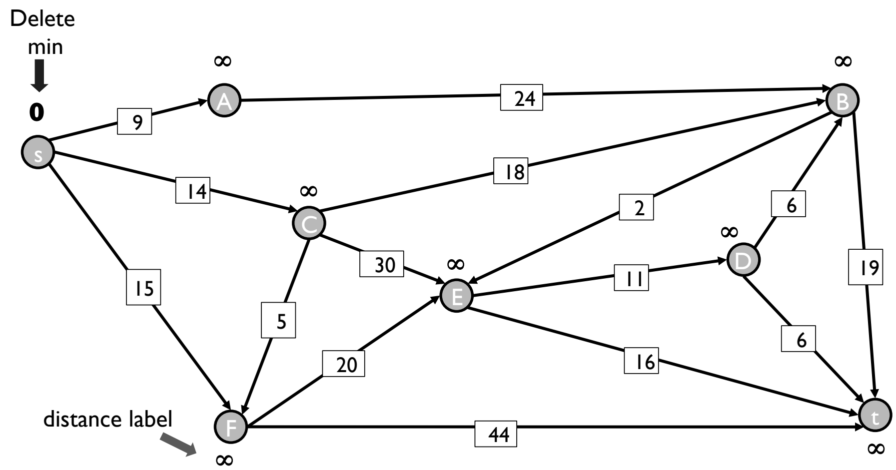
S = { }
PQ = { s, A, B, C, D, E, F, t }

Initialize distances to all nodes to infinity



Dijkstra’s Shortest Path Algorithm

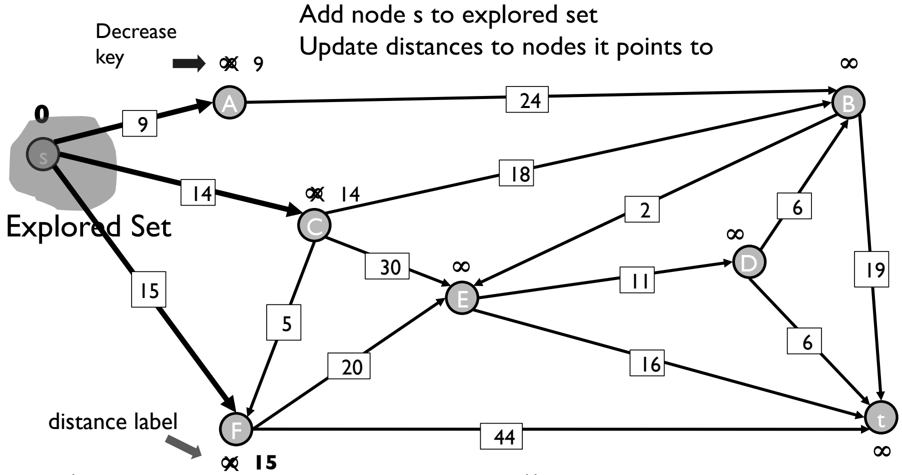
S = { }
PQ = { s, A, B, C, D, E, F, t }



Dijkstra’s Shortest Path Algorithm

S = { s }
PQ = { A, B, C, D, E, F, t }

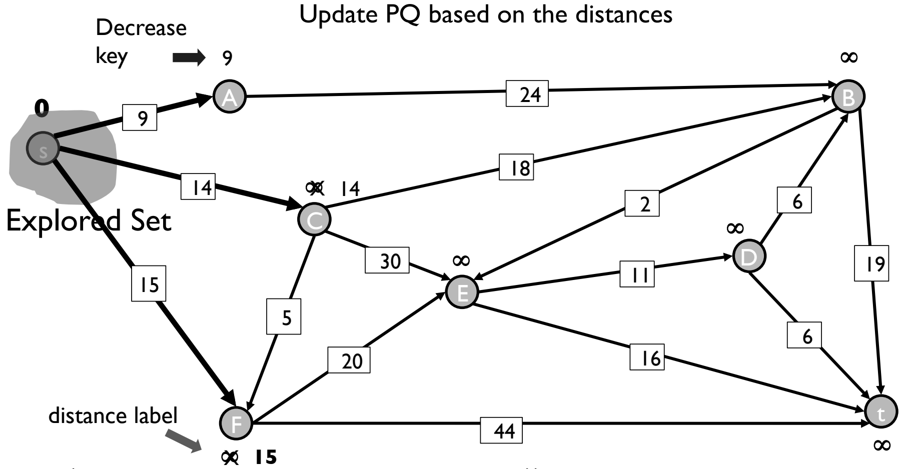
Add node s to explored set
Update distances to nodes it points to



Dijkstra’s Shortest Path Algorithm

S = { s }
PQ = { A, C, F, B, D, E, t }

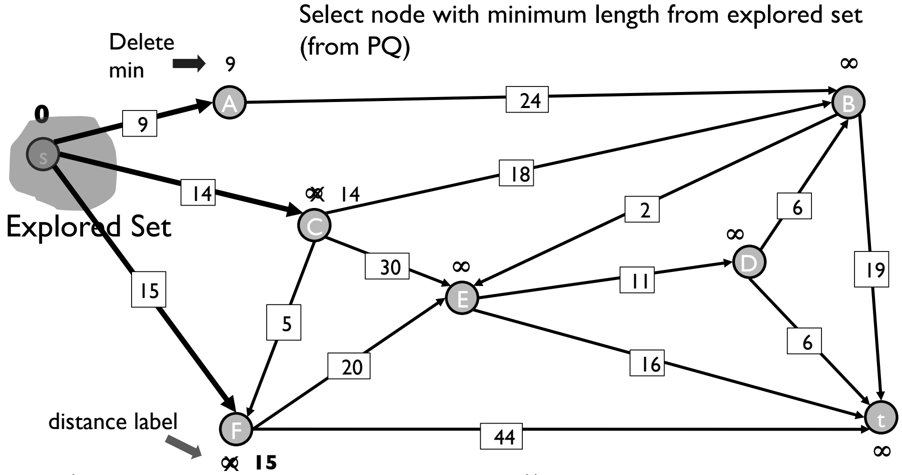
Update PQ based on the distances



Dijkstra’s Shortest Path Algorithm

S = { s }
PQ = { A, C, F, B, D, E, t }

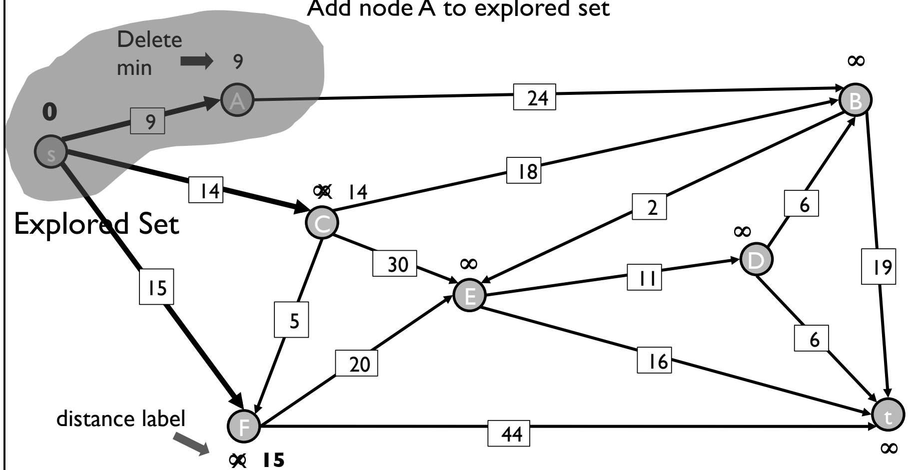
Select node with minimum length from explored set (from PQ)



Dijkstra's Shortest Path Algorithm

S = { s, A }
PQ = {C, F, B, D, E, t }

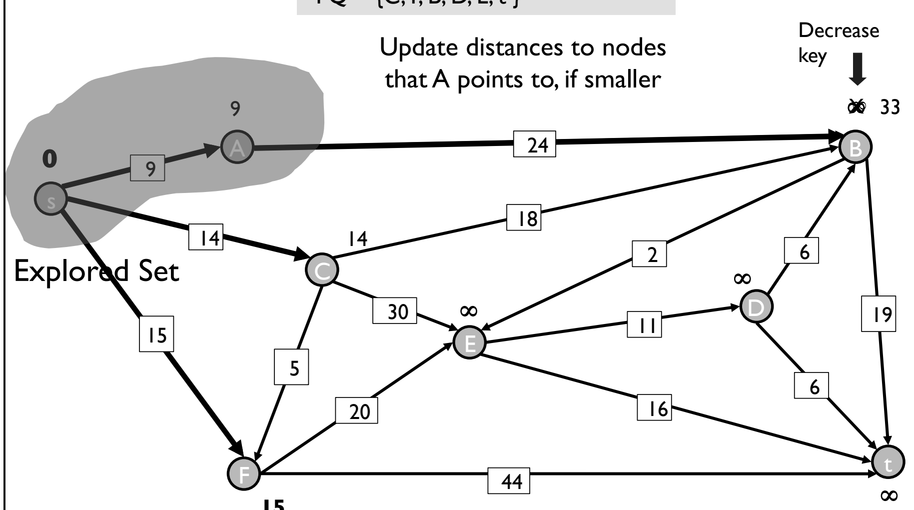
Add node A to explored set



Dijkstra's Shortest Path Algorithm

S = { s, A }
PQ = {C, F, B, D, E, t }

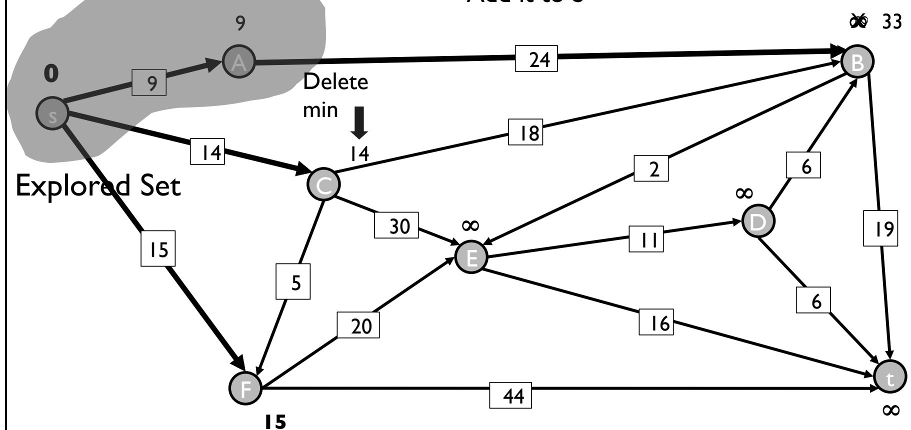
Update distances to nodes that A points to, if smaller



Dijkstra's Shortest Path Algorithm

S = { s, A }
PQ = {C, F, B, D, E, t }

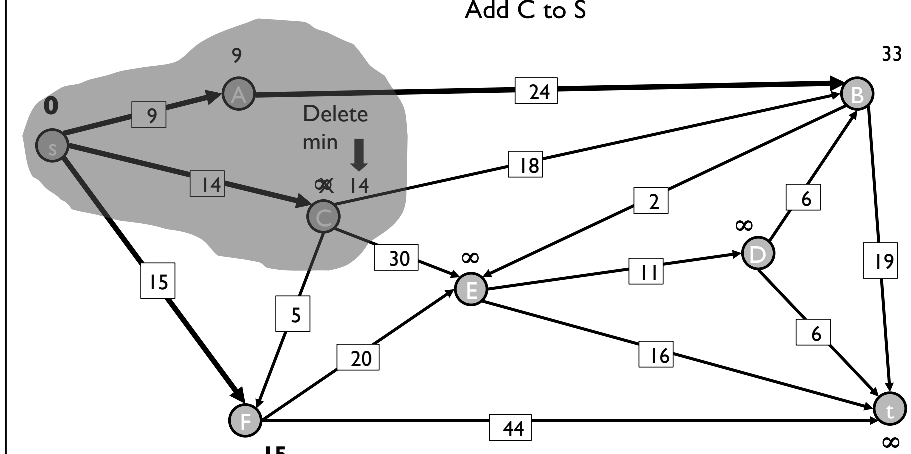
Delete the minimum node from PQ,
Add it to S



Dijkstra's Shortest Path Algorithm

S = { s, A, C }
PQ = { F, B, D, E, t }

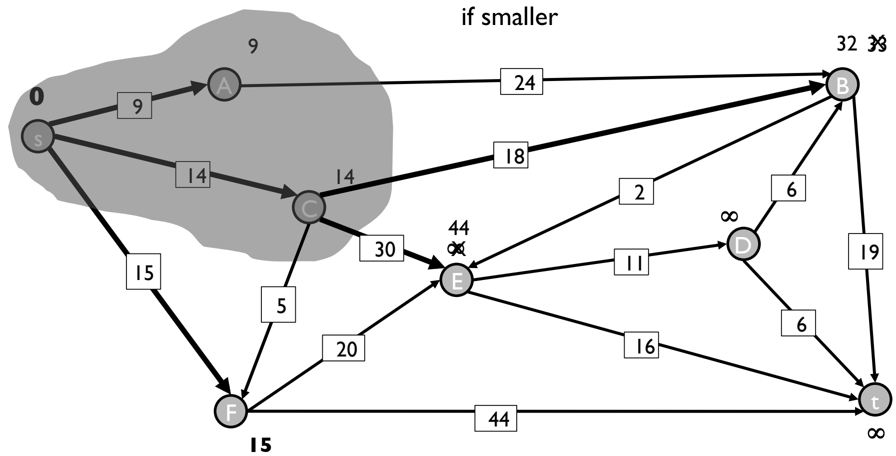
Add C to S



Dijkstra's Shortest Path Algorithm

S = { s, A, C }
PQ = { F, B, **E**, D, t }

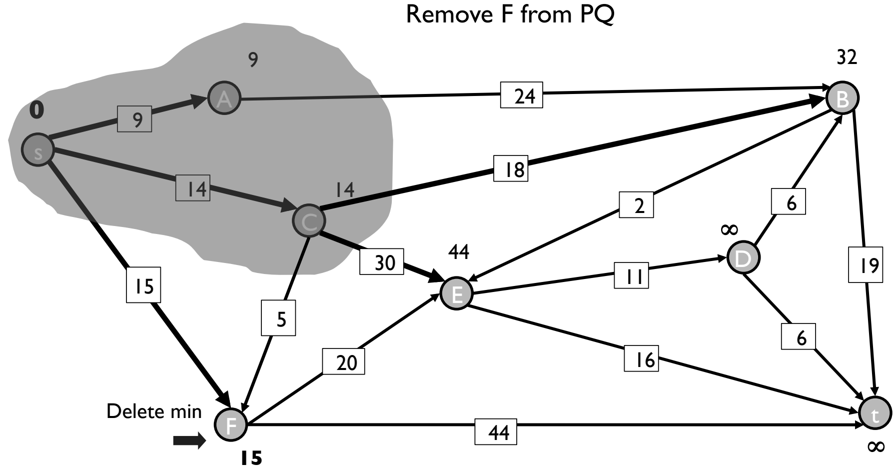
Update distances to nodes C points to, if smaller



Dijkstra's Shortest Path Algorithm

S = { s, A, C }
PQ = { F, B, E, D, t }

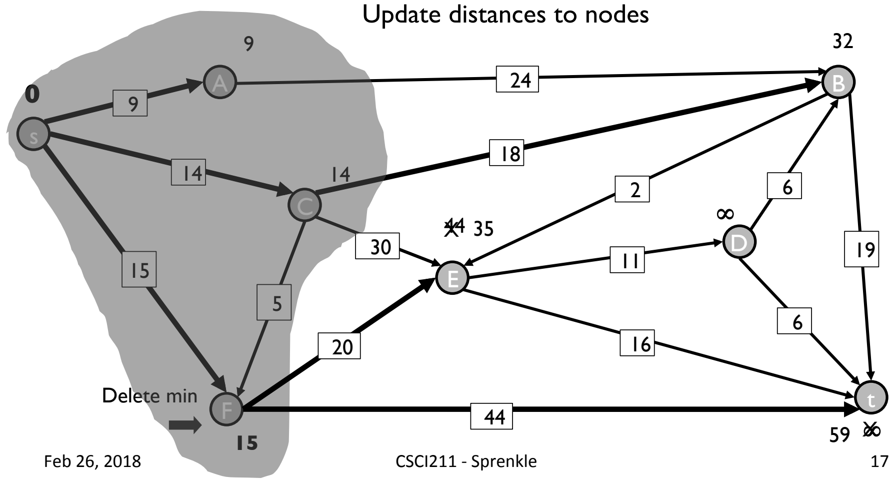
Remove F from PQ



Dijkstra’s Shortest Path Algorithm

S = { s, A, C, F }
PQ = { B, E, **t**, D }

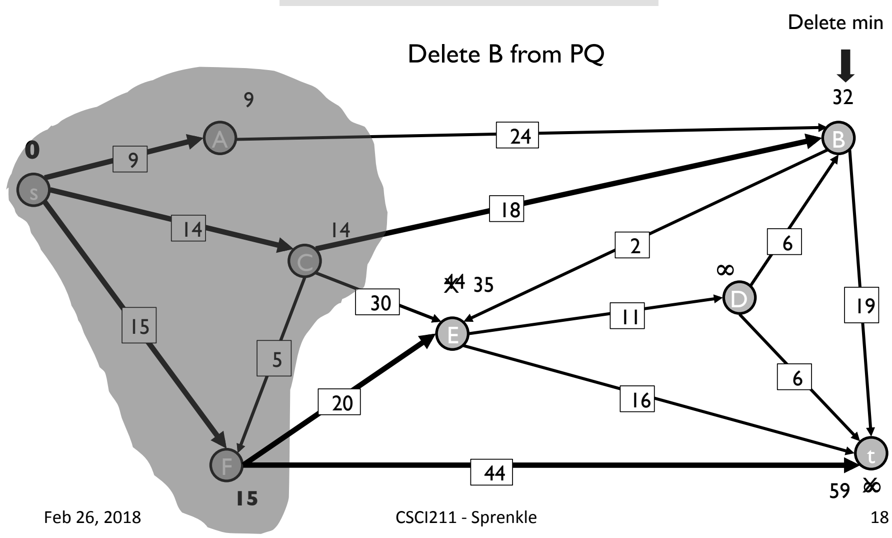
Add F to S,
Update distances to nodes



Dijkstra’s Shortest Path Algorithm

S = { s, A, C, F }
PQ = { B, E, t, D }

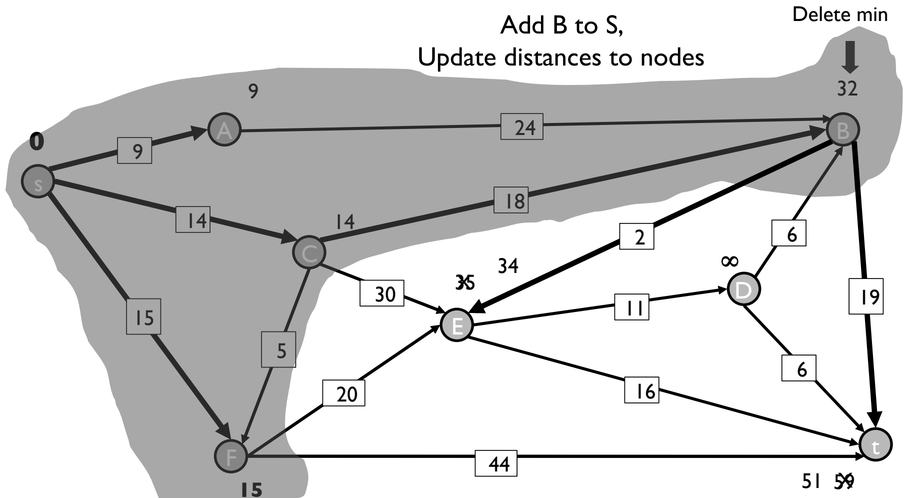
Delete B from PQ



Dijkstra's Shortest Path Algorithm

S = { s, A, C, F, B }
PQ = { E, t, D }

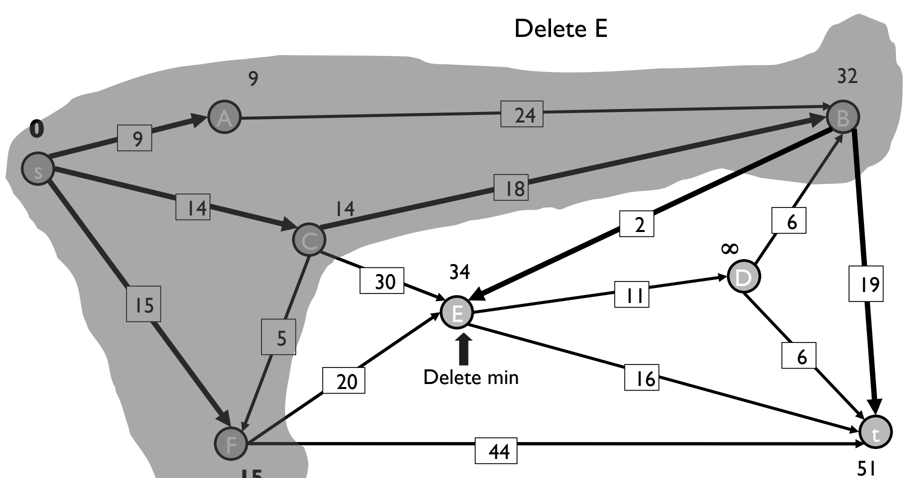
Add B to S,
Update distances to nodes



Dijkstra's Shortest Path Algorithm

S = { s, A, C, F, B }
PQ = { E, t, D }

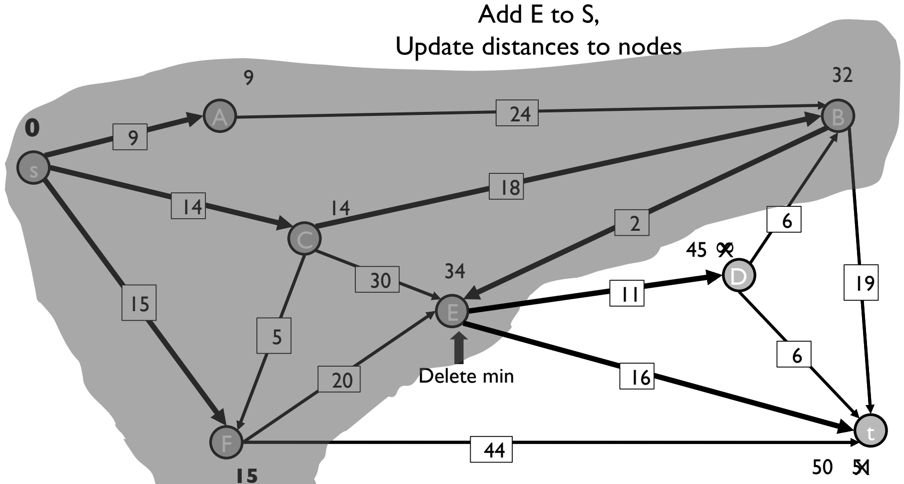
Delete E



Dijkstra’s Shortest Path Algorithm

S = { s, A, C, F, B, E }
PQ = { **D**, t }

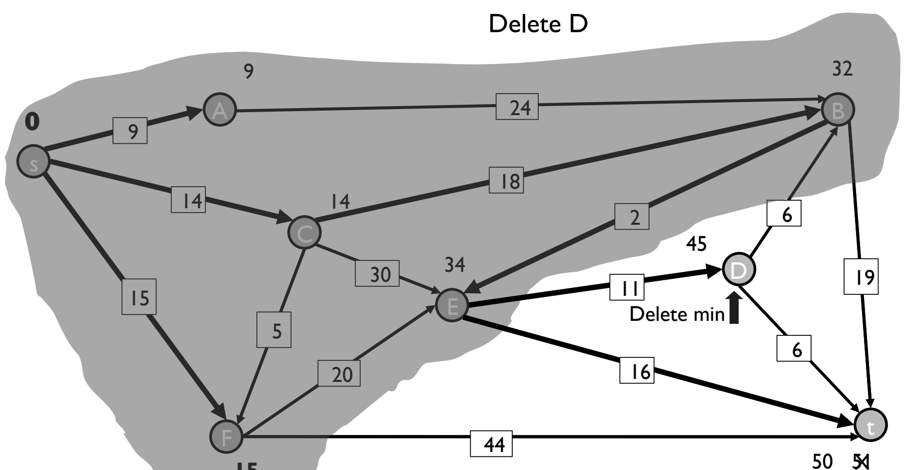
Add E to S,
Update distances to nodes



Dijkstra’s Shortest Path Algorithm

S = { s, A, C, F, B, E }
PQ = { D, t }

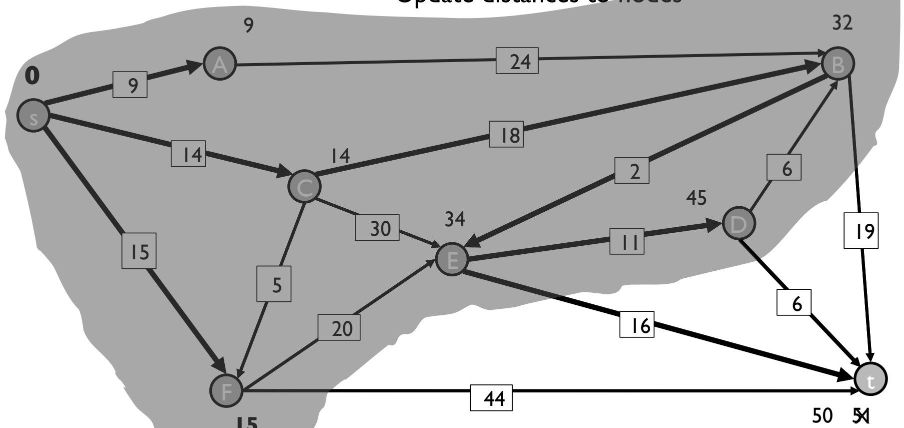
Delete D



Dijkstra's Shortest Path Algorithm

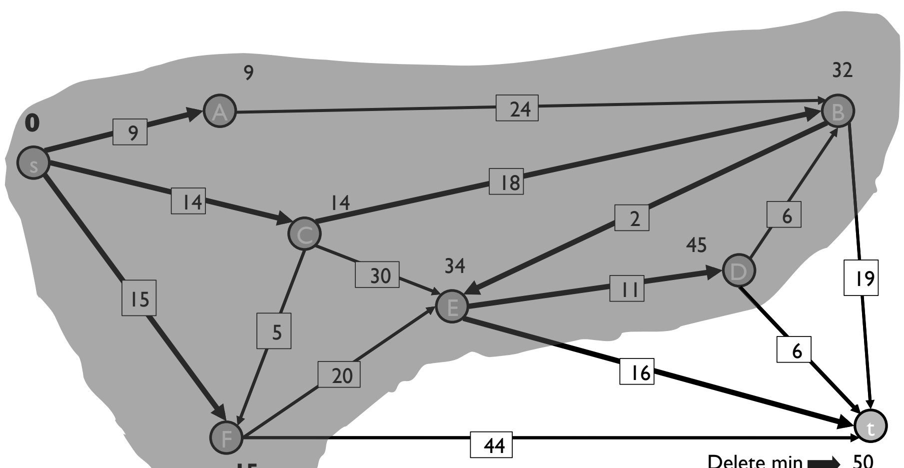
S = { s, A, C, F, B, E, D }
PQ = { t }

Add D to S,
Update distances to nodes



Dijkstra's Shortest Path Algorithm

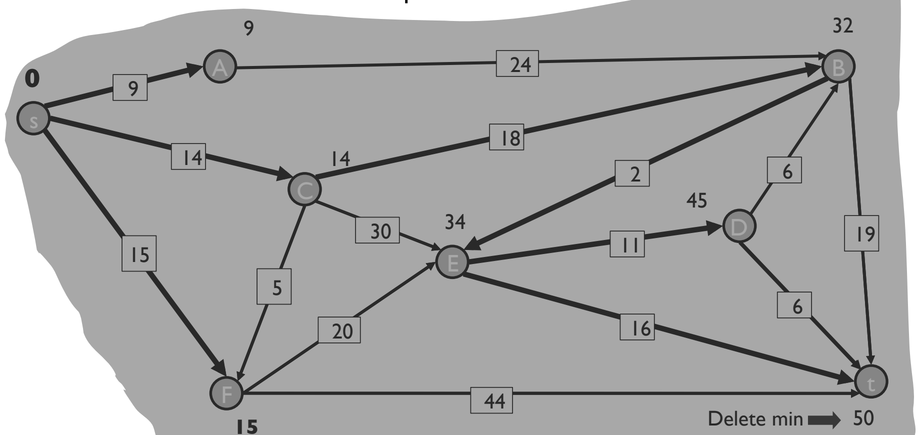
S = { s, A, C, F, B, E, D }
PQ = { t }



Dijkstra’s Shortest Path Algorithm

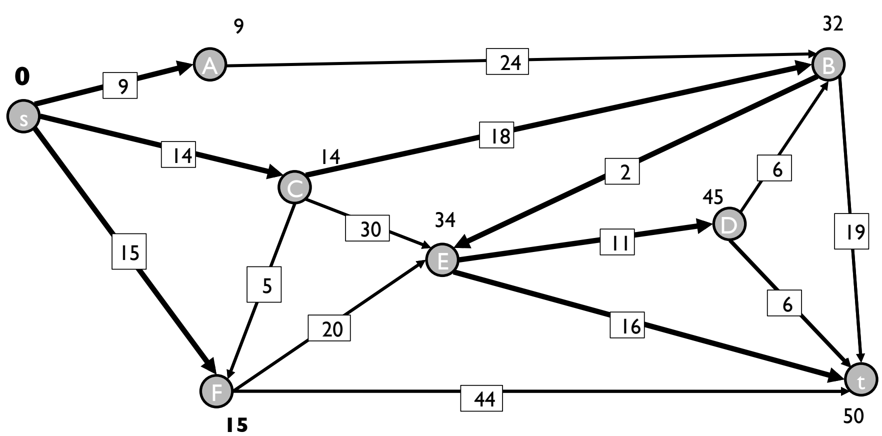
S = { s, A, C, F, B, E, D, t }
PQ = { }

Add t to S,
Update distances to nodes



Dijkstra’s Shortest Path Algorithm

S = { s, A, C, F, B, E, D, t }
PQ = { }

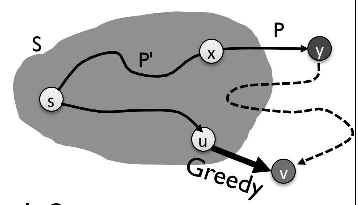


Dijkstra's Algorithm: Proof of Correctness

- Invariant. For each node u ∈ S, d(u) is the length of the shortest s-u path
- Pf. (by induction on |S|)
- Base case: |S|=1 …
- Inductive hypothesis?
- Next step?

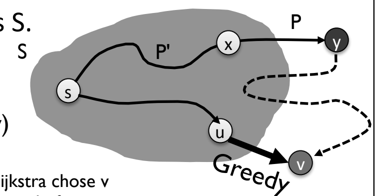
Dijkstra's Algorithm: Proof of Correctness

- Prove: For each node u ∈ S, d(u) is the length of the shortest s-u path
- Pf. (by induction on |S|)
- Base case: For |S| = 1, S={s}; d(s) = 0 ✓
- Inductive hypothesis:
  Assume true for |S| = k, k ≥ 1
- Proof:
  - Grow |S| to k+1
  - Greedy: Add node *v* by *u*→*v*
  - What do we know about *s*→*u*?
  - Why didn't we pick *y* as the next node?
  - What can we say about other *s*→*v* paths?



Dijkstra's Algorithm: Proof of Correctness

- Prove: For each node $u \in S$, $d(u)$ is the length of the shortest s-u path
- Pf. (by induction on $|S|$)
- Inductive hypothesis: Assume true for $|S| = k$, $k \geq 1$
- Proof:
  - Let *v* be the next node added to *S* by Greedy, 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. It's no shorter than $\pi(v)$.
  - Let *x*→*y* be the first edge in P that leaves S, and let P' be the subpath to *x*.
  - P is already too long as soon as it leaves S.



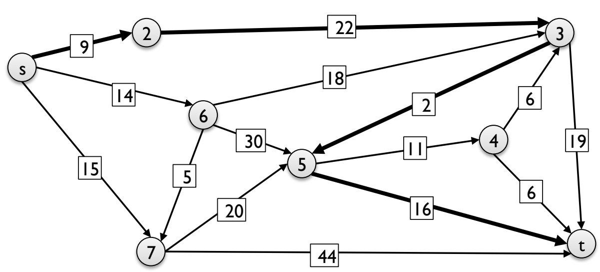
In terms of inequalities:

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

- first ≥: nonnegative weights
- =: inductive hypothesis
- second ≥: defn of $\pi(y)$
- third ≥: Dijkstra chose v instead of y

Discussion: Dijstra's Algorithm

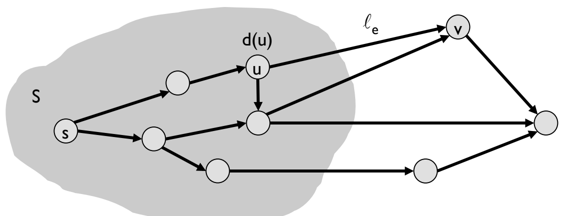
- Why does the algorithm break down if we allow negative weights/costs on edges?



Dijkstra's Algorithm: Analysis

1. Maintain a set of explored nodes S
   - Know the shortest path distance d(u) from *s* to *u*
2. Initialize S={s}, d(s)=0, ∀ u≠s, d(u)=∞
3. Repeatedly choose unexplored node *v* which minimizes $\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e ,$
   - Add v to S and set d(v) = π(v)

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



Running time?
Implementation?
Data structures?

Dijkstra's Algorithm: Analysis

1. Maintain a set of explored nodes S
   - Keep the shortest path distance d(u) from *s* to *u*
2. Initialize S={s}, d(s)=0, ∀ u≠s, d(u)=∞
3. Repeatedly choose unexplored node *v* which minimizes $\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e ,$
   - Add v to S and set d(v) = π(v)

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

PQ Operation	RT of Op	# in Dijkstra
Insert		
ExtractMin		
ChangeKey		
IsEmpty		
Total		

- How long does each operation take?
- How many of each operation?

Dijkstra's Algorithm: Implementation

- For each unexplored node, explicitly maintain

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

  - ➢ Next node to explore = node with minimum π(v).
  - ➢ When exploring v, for each incident edge e = (v, w), update $\pi(w) = \min \{ \pi(w), \pi(v) + \ell_e \}$.
- Efficient implementation. Maintain a priority queue of unexplored nodes, prioritized by π(v)

PQ Operation	RT of Op	# in Dijkstra
Insert	log n	n
ExtractMin	log n	n
ChangeKey	log n	m
IsEmpty	1	n
Total		

Dijkstra's Algorithm: Implementation

- For each unexplored node, explicitly maintain

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

  - ➢ Next node to explore = node with minimum π(v).
  - ➢ When exploring v, for each incident edge e = (v, w), update $\pi(w) = \min \{ \pi(w), \pi(v) + \ell_e \}$.
- Efficient implementation. Maintain a priority queue of unexplored nodes, prioritized by π(v)

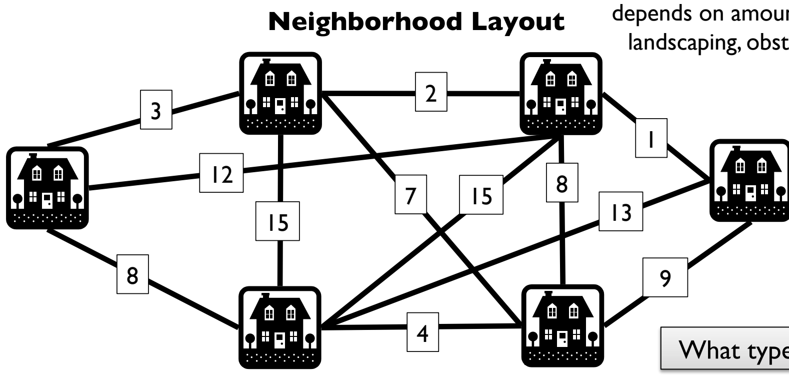
PQ Operation	RT of Op	# in Dijkstra
Insert	log n	n
ExtractMin	log n	n
ChangeKey	log n	m
IsEmpty	1	n
Total		**m log n**

O(m log n)

Laying Cable

- Comcast wants to lay cable in a neighborhood
  - Reach all houses
  - Least cost

Cost of laying cable btw houses depends on amount of cable, landscaping, obstacles, etc.



What type of graph?

Looking ahead

- Wiki today: Chapter 4 (front matter), 4.1, 4.2, 4.4
- PS5 due Friday