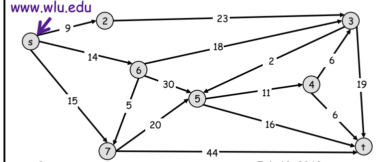
Objectives

- Problem: Shortest Path

Shortest Path Problem

- 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

www.wlu.edu



cost of path = sum of edge costs in path

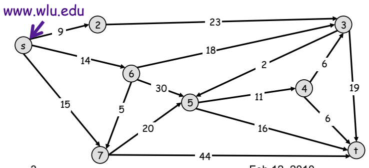
Cost of path s-2-3-5-t
= 9 + 23 + 2 + 16
= 48

www.cnn.com

Shortest Path Problem

- Shortest path problem: find shortest directed path from s to t
- Towards algorithm ideas:
  - What is shortest path from s→2? s→6?
  - What is the shortest path from s→3? 5? 7?

www.wlu.edu



cost of path = sum of edge costs in path

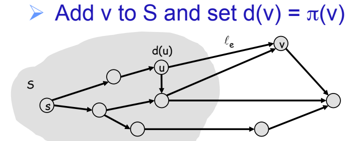
Cost of path s-2-3-5-t
= 9 + 23 + 2 + 16
= 48

www.cnn.com

Dijkstra’s Algorithm

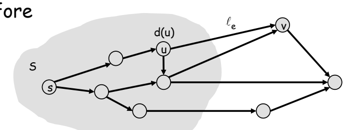
1. Maintain a set of **explored nodes** S
   - Keep the shortest path distance d(u) from *s* to *u*
2. Initialize S={s}, d(s)=0, ∀u≠s, d(u)=∞
3. Repeatedly choose unexplored node *v* which minimizes $\pi(v) = \min_{e = (u,v)\,:\, u \in S} d(u) + \ell_e$,
   - Add v to S and set d(v) = π(v)

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

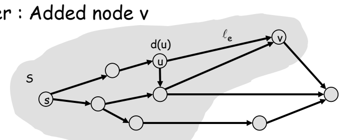


Dijkstra's Algorithm

Before

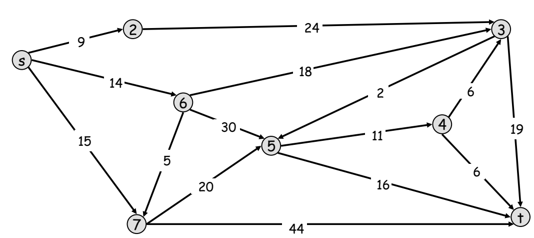


After : Added node v



Dijkstra's Shortest Path Algorithm

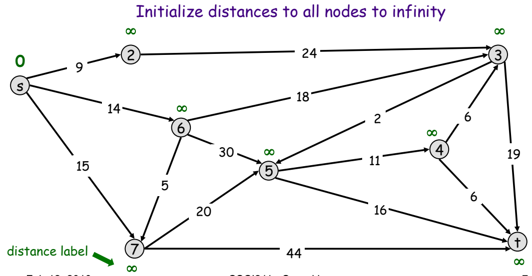
- Find shortest path from s to t.



Dijkstra's Shortest Path Algorithm

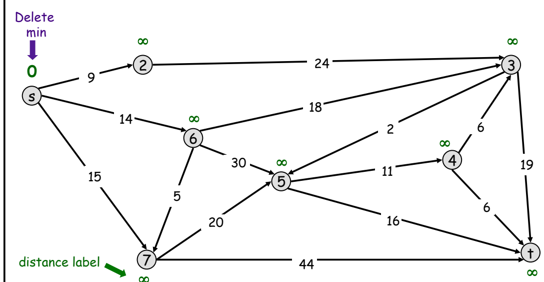
S = { }
PQ = { s, 2, 3, 4, 5, 6, 7, t }

Initialize distances to all nodes to infinity



Dijkstra's Shortest Path Algorithm

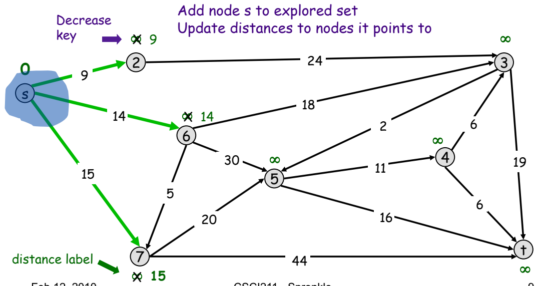
S = { }
PQ = { s, 2, 3, 4, 5, 6, 7, t }



Dijkstra's Shortest Path Algorithm

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

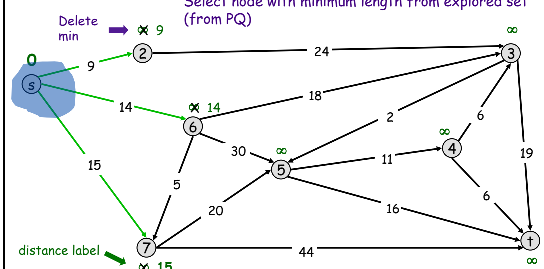
Add node s to explored set
Update distances to nodes it points to



Dijkstra's Shortest Path Algorithm

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

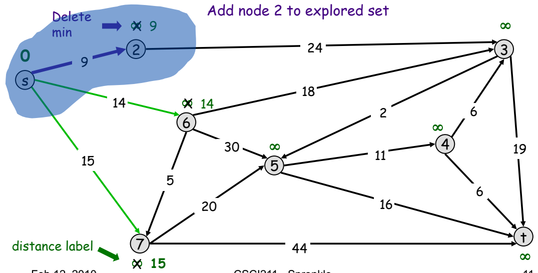
Select node with minimum length from explored set (from PQ)



Dijkstra's Shortest Path Algorithm

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

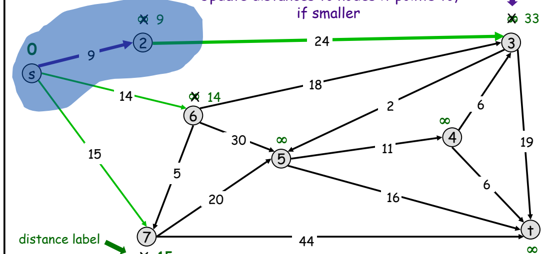
Add node 2 to explored set



Dijkstra's Shortest Path Algorithm

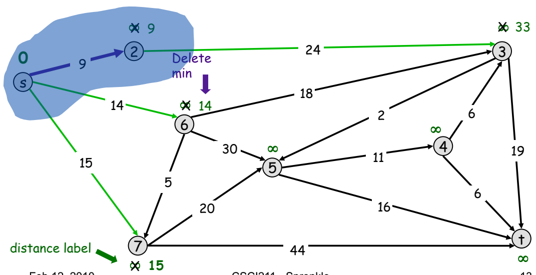
S = { s, 2 }
PQ = { 6, 7, 3, 4, 5, t }

Update distances to nodes it points to, if smaller



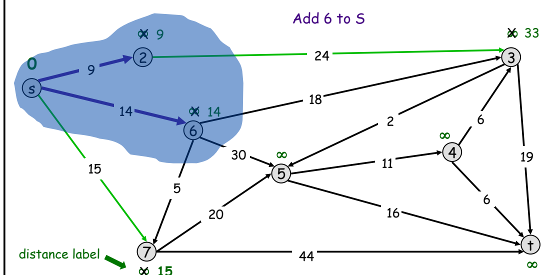
Dijkstra's Shortest Path Algorithm

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



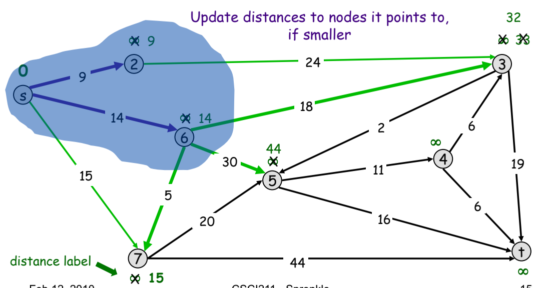
Dijkstra's Shortest Path Algorithm

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



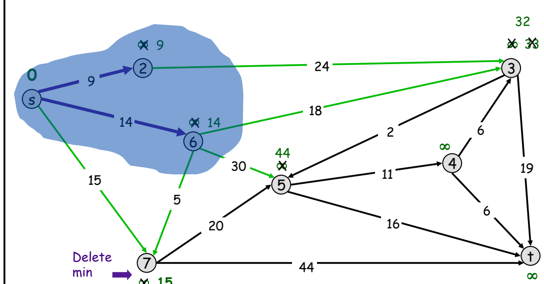
Dijkstra's Shortest Path Algorithm

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



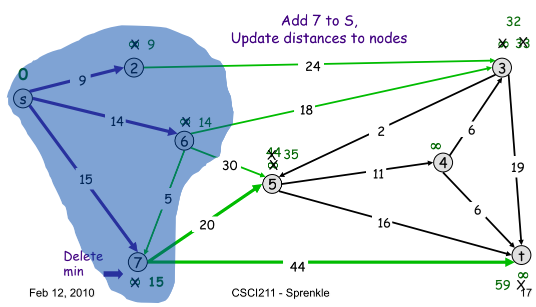
Dijkstra's Shortest Path Algorithm

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



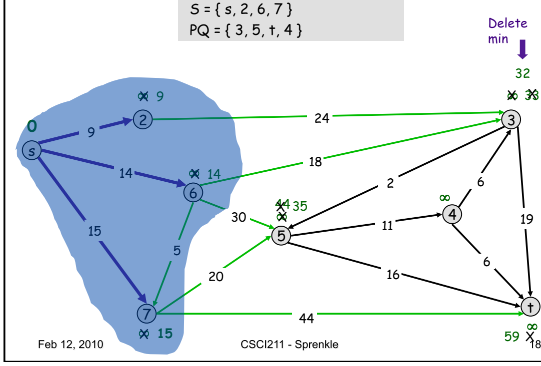
Dijkstra's Shortest Path Algorithm

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



Dijkstra's Shortest Path Algorithm

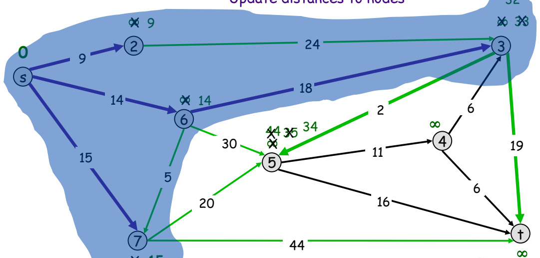
S = { s, 2, 6, 7 }
PQ = { 3, 5, t, 4 }



Dijkstra's Shortest Path Algorithm

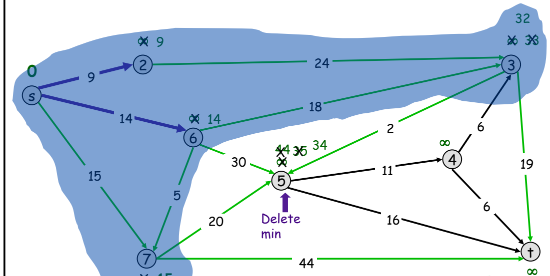
S = { s, 2, 6, 7, 3 }
PQ = { 5, t, 4 }

Add 3 to S,
Update distances to nodes



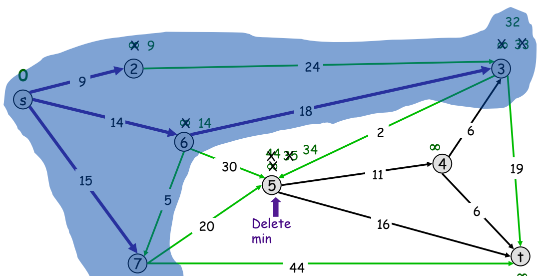
Dijkstra's Shortest Path Algorithm

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



Dijkstra's Shortest Path Algorithm

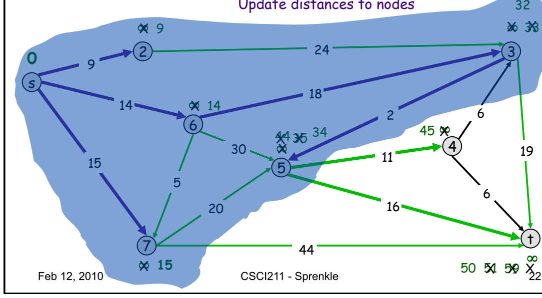
S = { s, 2, 6, 7, 3 }
PQ = { 5, t, 4 }



Dijkstra's Shortest Path Algorithm

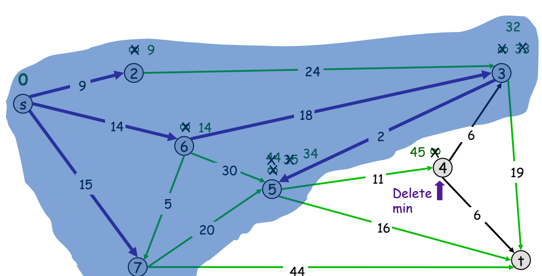
S = { s, 2, 6, 7, 3, 5 }
PQ = { 4, t }

Add 5 to S,
Update distances to nodes



Dijkstra's Shortest Path Algorithm

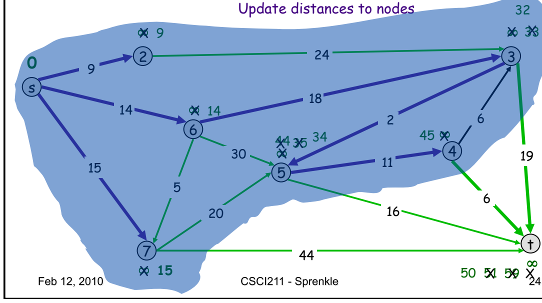
S = { s, 2, 6, 7, 3, 5 }
PQ = { 4, t }



Dijkstra's Shortest Path Algorithm

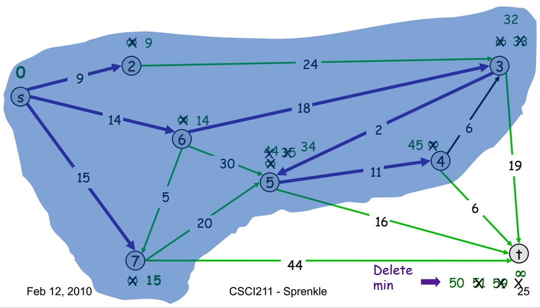
S = { s, 2, 6, 7, 3, 5, 4 }
PQ = { t }

Add 4 to S,
Update distances to nodes



Dijkstra's Shortest Path Algorithm

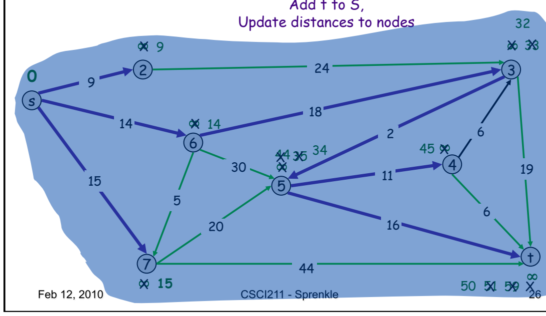
S = { s, 2, 6, 7, 3, 5, 4 }
PQ = { t }



Dijkstra's Shortest Path Algorithm

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

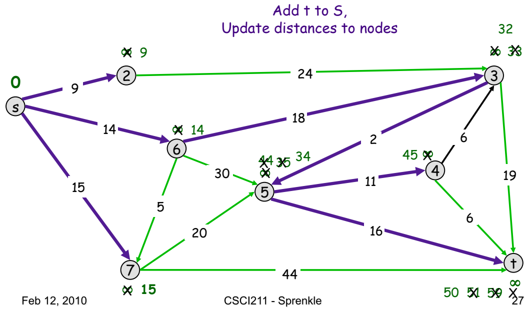
Add t to S,
Update distances to nodes



Dijkstra's Shortest Path Algorithm

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

Add t to S,
Update distances to nodes



How Greedy?

How Greedy?

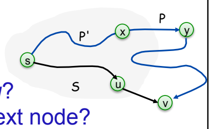
- We always form **shortest new *s-v* path** from a path in S followed by a *single* edge
- Proof of optimality: *Stays ahead* of all other solutions
  - Each time selects a path to a node *v*, that path is shorter than every other possible path to *v*

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?

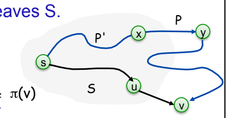
Dijkstra's Algorithm: Proof of Correctness

- Invariant. 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
  - 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

- Invariant. For each node u ∈ 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 ≥ 1
  - 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 π(v)
  - Consider any *s*→*v* path P. It's no shorter than π(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 (in)equalities:

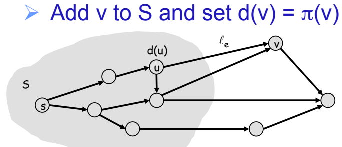
$\ell(P) \geq \ell(P') + \ell(x,y) = d(x) + \ell(x,y) \geq \pi(y) \geq \pi(v)$

nonnegative weights; inductive hypothesis; defn of π(y); Dijkstra chose v instead of y

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
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)

Using a priority queue, how many
- Inserts?
- Finding minimum?
- Deletions?
- Updating the key?
- Determining if empty?

How long does each operation take?

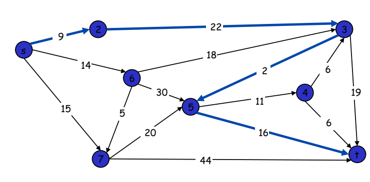
Dijkstra's Algorithm: Implementation

- 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	Dijkstra	Binary heap
Insert	n	log n
ExtractMin	n	log n
ChangeKey	m	log n
IsEmpty	n	1
Total		m log n

Discussion: Dijstra's Algorithm

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



Assignments

- Read Chapter 4
  - Wiki due next Wednesday
- Problem Set 4 due next Friday