Shortest paths and MSTs

- What's a shortest path?
- What's a MST?
- How are they related?
- How are they different?
Draw next to each node the cost of the shortest path from A to that node.
Shortest path
Shortest path
Shortest path
Minimum spanning tree

Draw the minimum spanning tree of this graph
Minimum spanning tree

Distance from A to B in MST?
Distance from A to D in MST?
function **dijkstra**(G, s):

// Input: graph G with vertices V, and source s
// Output: Nothing
// Purpose: Decorate nodes with shortest distance from s

for v in V:
    v.dist = infinity  // Initialize distance decorations
    v.prev = null  // Initialize previous pointers to null
s.dist = 0  // Set distance to start to 0

PQ = PriorityQueue(V)  // Use v.dist as priorities

while PQ not empty:
    u = PQ.removeMin()
    for all edges (u, v):  // each edge coming out of u
        if u.dist + cost(u, v) < v.dist:  // cost() is weight
            v.dist = u.dist + cost(u,v)  // Replace as necessary
            v.prev = u  // Maintain pointers for path
            PQ.decreaseKey(v, v.dist)
function `prim(G)`:
  // Input: weighted, undirected graph G with vertices V
  // Output: list of edges in MST
  for all v in V:
    v.cost = ∞
    v.prev = null
  s = a random v in V // pick a random source s
  s.cost = 0
  MST = []
  PQ = PriorityQueue(V) // priorities will be v.cost values
  while PQ is not empty:
    v = PQ.removeMin()
    if v.prev != null:
      MST.append((v, v.prev))
    for all incident edges (v,u) of v such that u is in PQ:
      if u.cost > (v,u).weight:
        u.cost = (v,u).weight
        u.prev = v
        PQ.decreaseKey(u, u.cost)
  return MST
For the final...

- To study: look over homeworks, notes
- Rewrite definitions *in your own words*
- In answering questions:
  - Be explicit and clear
  - Convince us you understand!
What we’ve done this semester

- Analysis
  - Big-O
  - Worst-case analysis
  - Amortized analysis
  - Average-case analysis
- Social responsibility
What we’ve done this semester

- Data structures
  - Dynamic stacks, queues, lists
  - Hash tables
  - Trees
    - BSTs
    - Heaps
  - Graphs
What we’ve done this semester

- Algorithms
  - Recursive
  - Dynamic programming
  - Searching trees and graphs
  - Sorting
  - Shortest paths
  - MSTs
  - Topological sort
What we’ve done this semester

- Other stuff
  - Basics of machine learning
  - Functional programming
  - Hardness
  - Program verification
Some advice

- Sometimes performance doesn’t matter
  - Programs that will run once on small data
  - Cases where \( n \) is always small
- When it does, focus on big-O first
- Then on smaller things (constant factors, language choice, etc.)
Some advice

‣ Social responsibility: be prepared

‣ If you go on in CS (but really, regardless of what you do) at some point you’ll have to make a choice
  ‣ Your boss asks you to implement something ethically questionable
  ‣ You get a job offer from a company whose work conflicts with your values

‣ Worth spending some time thinking about what you’ll do
Some advice

- One reason to learn data structures and algorithms: try not to reinvent the wheel
- You're looking at a problem (for an independent class project, for work, for research, etc.)
  - Can this problem be represented as a graph?
  - Would a priority queue be useful?
  - Is this problem amenable to dynamic programming?
  - Is this problem NP-complete?
- You might not remember the details of Dijkstra's algorithm after this semester
  - But you'll know it's there when you need it!