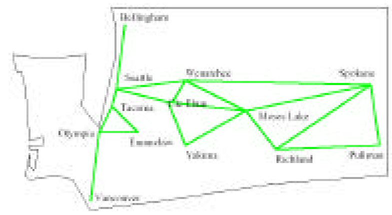
CSE 326: Data Structures
Minimum Spanning Trees
and
The Dynamic Duo (Prim and Kruskal)

Hannah Tang and Brian Tjaden

Summer Quarter 2002

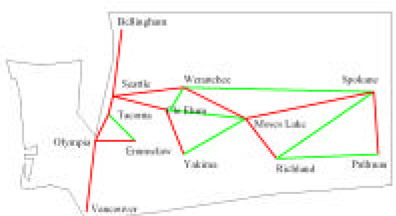
Some Applications: Moving Around Washington



What's the fastest way from Seattle to Spokane?

Use Dijkstra's Algorithm!

Some Applications: Communication in Washington

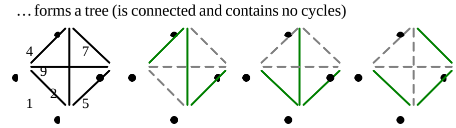


What's the cheapest inter-city network?

Spanning Tree

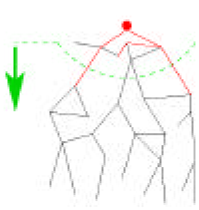
Spanning tree: a subset of the edges from a connected graph that…

- … touches all vertices in the graph (*spans* the graph)
- … forms a tree (is connected and contains no cycles)



Minimum spanning tree: the spanning tree with the least total edge cost.

Two Different Algorithms



Prim's Algorithm

Almost identical to Dijkstra's

Kruskals's Algorithm

Completely different!

Prim's Algorithm for Minimum Spanning Trees

A node-oriented greedy algorithm (builds an MST by greedily adding nodes)

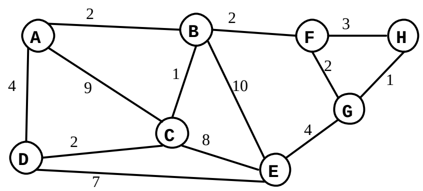
Select a node to be the "root" and mark it as known

While there are unknown nodes left in the graph

- Select the unknown node n with the smallest cost from some known node m
- Mark n as known
- Add (m, n) to our MST

Runtime:

Prim's Algorithm In Action



Kruskal's Algorithm for Minimum Spanning Trees

An edge-oriented greedy algorithm (builds an MST by greedily adding edges)

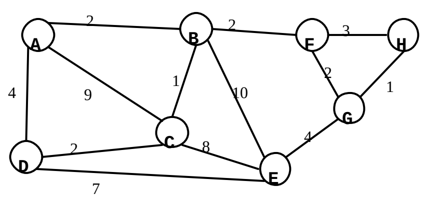
Initialize all vertices to unconnected

While there are still unmarked edges

- Pick the lowest cost edge **`e = (u, v)`** and mark it
- If **`u`** and **`v`** are not already connected, add **`e`** to the minimum spanning tree and connect **`u`** and **`v`**

Sound familiar?
(Think maze generation.)

Kruskal's Algorithm In Action



Proof of Correctness

We already showed this finds a spanning tree:

- That was part of our definition of a good maze.

Proof by contradiction that Kruskal's finds the minimum:

- Assume another spanning tree has *lower cost* than Kruskal's
- Pick an edge $\mathbf{e_1}$ **`= (u, v)`** in that tree that's *not* in Kruskal's
- Kruskal's tree connects **`u`**'s and **`v`**'s sets with another edge $\mathbf{e_2}$
- But, $\mathbf{e_2}$ must have at most the same cost as $\mathbf{e_1}$!
- So, swap $\mathbf{e_2}$ for $\mathbf{e_1}$ (at worst keeping the cost the same)
- Repeat until the tree is identical to Kruskal's: **contradiction**!

QED: Kruskal's algorithm finds a minimum spanning tree.

Data Structures for Kruskal's Algorithm

`|E|` times:
Pick the lowest cost edge…

- findMin/deleteMin

`|E|` times:
If **`u`** and **`v`** are not already connected…
…connect **`u`** and **`v`**.

- find representative
- union

Runtime: