Graphs III
Chapter 9 in Weiss

CSE 326
Data Structures
Ruth Anderson

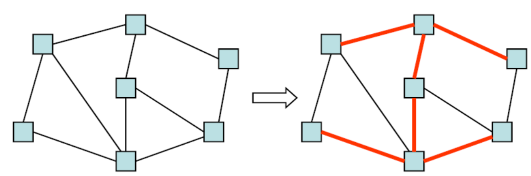
Today's Outline

- **Announcements**
  - **Written Homework #7 due Fri March 5**
  - **Project 3 Benchmarking & Written (and Above & Beyond) due Fri March 5 by 11pm**
  - **Last Homework! Written Homework #8 due Fri March 1**
- **Today's Topics:**
  - **Graphs**
    - **Minimum Spanning Trees**
      - Prim
      - Kruskal

Graphs so far

- Representations
- Topological Sort
- Finding paths
  - DFS
  - BFS
  - Dijkstra
- Minimum Spanning Trees
  - Prim
  - Kruskal

Spanning Tree in a Graph



Vertex = router
Edge = link between routers

Spanning tree
- Connects all the vertices
- No cycles

Spanning Tree problem

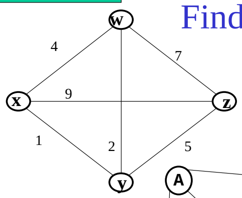
- Input: An undirected graph G = (V, E) such that G is connected
- Output: E' contained in E such that
  - (V, E') is a connected graph
  - (V, E') is acyclic

Minimum Spanning Tree problem

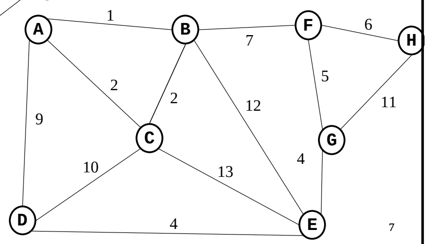
- Input: An undirected graph G = (V, E) such that G is connected
- Output: E' contained in E such that
  - (V, E') is a connected graph
  - (V, E') is acyclic
  - $\sum_{(u,v)\in E'} c_{uv}$ is minimal
  - The graph G' = (V, E') is a Minimum Spanning Tree

Student Activity

Find the MST

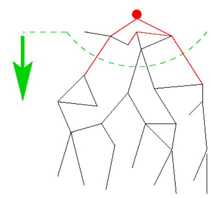


How many edges in a MST?

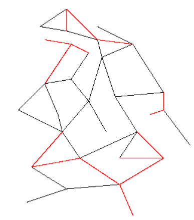


What is the total cost of each MST?

Two Different Approaches



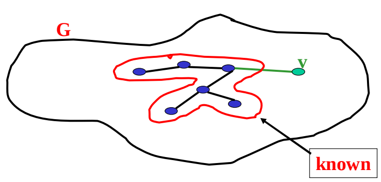
Prim's Algorithm



Kruskals's Algorithm

Prim's algorithm

Idea: Grow a tree by adding an edge from the "known" vertices to the "unknown" vertices. Pick the ***edge with the smallest weight.***



Prim's Algorithm for MST

A *node-based* greedy algorithm

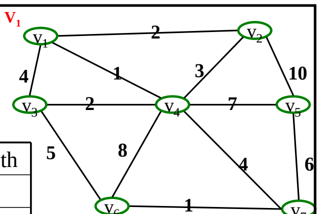
Builds MST by greedily adding nodes

1. Select a node to be the "root"
   - mark it as known
   - Update cost of all its neighbors
2. While there are unknown nodes left in the graph
   a. Select an unknown node *b* with the smallest cost from some *known* node *a*
   b. Mark *b* as known
   c. Add (*a*, *b*) to MST
   d. Update cost of all nodes adjacent to *b*

Student Activity

Start with V_1

Find MST using Prim's



V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

Order Declared Known:

V_1

Total Cost:

Prim's Algorithm Analysis

Running time:

Same as Dijkstra's: O(|E| log |V|)

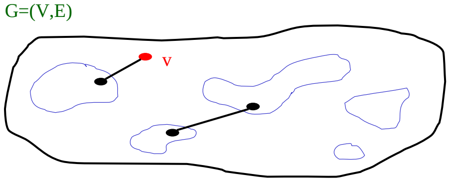
Correctness:

Proof is similar to Dijkstra's

Kruskal's MST Algorithm

Idea: Grow a forest out of edges that do not create a cycle. Pick an ***edge with the smallest weight.***

G=(V,E)



Kruskal's Algorithm for MST

An *edge-based* greedy algorithm

Builds MST by greedily adding edges

1. Initialize with
   - empty MST
   - all vertices marked unconnected
   - all edges unmarked
2. While there are still unmarked edges
   a. Pick the lowest cost edge `(u,v)` and mark it
   b. If **u** and **v** are not already connected, add `(u,v)` to the MST and mark **u** and **v** as connected to each other

Doesn't it sound familiar?

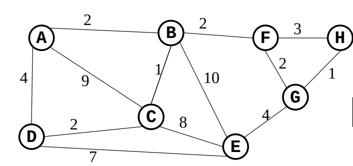
Kruskal code

```
void Graph::kruskal(){
  int edgesAccepted = 0;
  DisjSet s(NUM_VERTICES);

  while (edgesAccepted < NUM_VERTICES – 1){
    e = smallest weight edge not deleted yet;
    // edge e = (u, v)
    uset = s.find(u);
    vset = s.find(v);
    if (uset != vset){
      edgesAccepted++;
      s.unionSets(uset, vset);
    }
  }
}
```


Student Activity

Find MST using Kruskal's



Total Cost:

- Now find the MST using Prim's method.
- Under what conditions will these methods give the same result?

Kruskal's Algorithm: Correctness

It clearly generates a spanning tree. Call it T_K.

Suppose T_K is *not* minimum:

Pick another spanning tree T_{min} with *lower cost* than T_K

Pick the smallest edge $e_1=(u,v)$ in T_K that is **not** in T_{min}

T_{min} already has a path p in T_{min} from u to v

$\Rightarrow$ Adding e_1 to T_{min} will create a cycle in T_{min}

Pick an edge e_2 in p that Kruskal's algorithm considered *after* adding e_1 (must exist: u and v unconnected when e_1 considered)

$\Rightarrow$ cost(e_2) $\geq$ cost(e_1)

$\Rightarrow$ can replace e_2 with e_1 in T_{min} without increasing cost!

Keep doing this until T_{min} is identical to T_K

$\Rightarrow$ T_K must also be minimal – contradiction!