CSE 326: Data Structures
Minimum Spanning Trees

Hal Perkins

Spring 2007

Lectures 26

Today's Outline

Minimum Spanning Tree

1. Prim's
2. Kruskal's

Reading: Weiss, Ch. 9

Minimum Spanning Trees

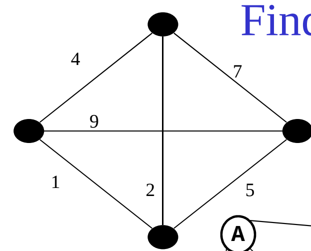
Given an undirected graph **G**=(**V**,**E**), find a graph **G'**=(**V**, **E'**) such that:

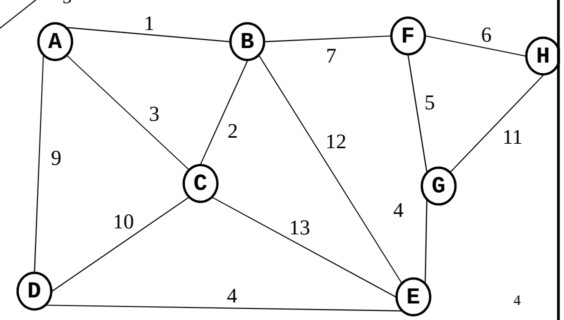
- E' is a subset of E
- |E'| = |V| - 1
- G' is connected
- $\sum_{(u,v)\in E'} c_{uv}$ is minimal

G' is a **minimum spanning tree**.

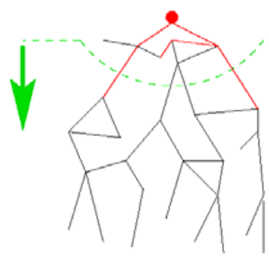
Applications: wiring a house, power grids, Internet connections

Find the MST





Two Different Approaches



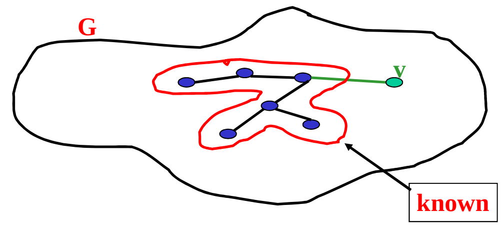
Prim's Algorithm
Almost identical to Dijkstra's

Kruskals's Algorithm
Completely different!

5

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.



6

Prim's Algorithm for MST

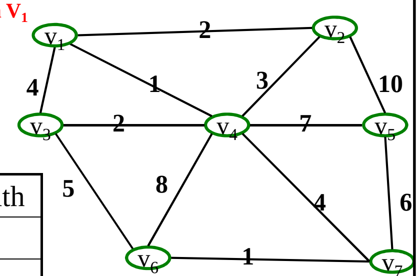
A *node-based* greedy algorithm
Builds MST by greedily adding nodes

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*

7

Find MST using Prim's

Start with V_1



V	Kwn	Distance	path
v1			
v2			
v3			
v4			
v5			
v6			
v7			

Order Declared Known:
V_1

8

Prim's Algorithm Analysis

Running time:

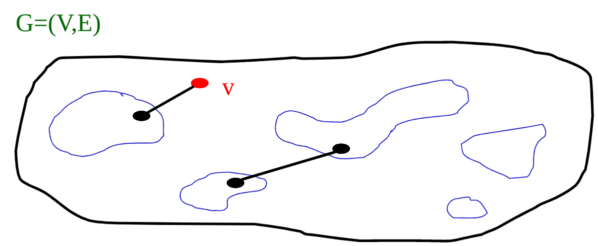
Same as Dijkstra's: O(|E| log |V|)

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.



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

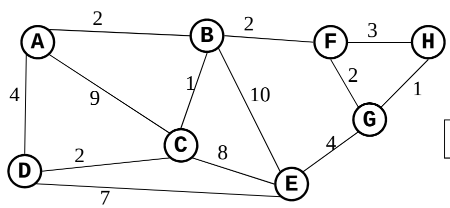
```
void Graph::kruskal(){
  int edgesAccepted = 0;
  DisjSet s(NUM_VERTICES);

  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);
    }
  }
}
```

|E| heap ops

2|E| finds

|V| unions

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?

13

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!

14