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

Applications: wiring a house, power grids, Internet connections

Today’s Outline

Minimum Spanning Tree
1. Prim’s
2. Kruskal’s

Reading: Weiss, Ch. 9
Two Different Approaches

Prim’s Algorithm
Almost identical to Dijkstra’s

Kruskals’s Algorithm
Completely different!

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

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

Find MST using Prim’s

Start with V1

Order Declared Known: V1

Start with V1
Prim’s Algorithm Analysis

Running time:
Same as Dijkstra’s: \( O(|E| \log |V|) \)

Correctness:
Proof is similar to Dijkstra’s

Kruskal’s 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

Kruskal code

```cpp
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);
        }
    }
}
```

Doesn’t it sound familiar?
Your Turn

Find MST using Kruskal’s

- 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 $T_{\text{min}}$ with lower cost than $T_K$
- Pick the smallest edge $e_1 = (u, v)$ in $T_K$ that is not in $T_{\text{min}}$
- $T_{\text{min}}$ already has a path $p$ in $T_{\text{min}}$ from $u$ to $v$
  - Adding $e_1$ to $T_{\text{min}}$ will create a cycle in $T_{\text{min}}$
- 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)
  - $\text{cost}(e_2) \geq \text{cost}(e_1)$
  - can replace $e_2$ with $e_1$ in $T_{\text{min}}$ without increasing cost!
- Keep doing this until $T_{\text{min}}$ is identical to $T_K$
  - $T_K$ must also be minimal – contradiction!