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 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
**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 with the smallest cost from some **known** node
   b. Mark **b** as **known**
   c. Add \((a, b)\) to MST
   d. Update cost of all nodes adjacent to **b**

---

**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

\[ \text{Doesn’t it sound familiar?} \]

Kruskal code

```c
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);
    }
  }
}
```

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 \(T_{\text{mn}}\) with lower cost than \(T_k\)

Pick the smallest edge \(e_1 = (u,v)\) in \(T_k\) that is not in \(T_{\text{mn}}\)

\(T_{\text{mn}}\) already has a path \(p\) in \(T_{\text{mn}}\) from \(u\) to \(v\)

\(\Rightarrow\) Adding \(e_1\) to \(T_{\text{mn}}\) will create a cycle in \(T_{\text{mn}}\)

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)

\(\Rightarrow\) \(\text{cost}(e_2) \geq \text{cost}(e_1)\)

\(\Rightarrow\) can replace \(e_2\) with \(e_1\) in \(T_{\text{mn}}\) without increasing cost!

Keep doing this until \(T_{\text{mn}}\) is identical to \(T_k\)

\(\Rightarrow\) \(T_k\) must also be minimal – contradiction!