CSE 373  
Data Structures & Algorithms  

Lectures 21-22  
Minimum Spanning Tree
A Hidden Tree
Spanning Tree in a Graph

Vertex = router
Edge = link between routers

Spanning tree
- Connects all the vertices
- No cycles
Undirected Graph

- **G = (V,E)**
  - V is a set of vertices (or nodes)
  - E is a set of unordered pairs of vertices

V = \{1,2,3,4,5,6,7\}
E = \{\{1,2\},\{1,6\},\{1,5\},\{2,7\},\{2,3\},\{3,4\},\{4,7\},\{4,5\},\{5,6\}\}

2 and 3 are adjacent
2 is incident to edge \{2,3\}
Spanning Tree Problem

• Input: An undirected graph $G = (V,E)$. $G$ is connected.

• Output: $T$ contained in $E$ such that
  – $(V,T)$ is a connected graph
  – $(V,T)$ has no cycles
Spanning Tree Algorithm

ST(i: vertex)
mark i;
for each j adjacent to i do
  if j is unmarked then
    Add {i,j} to T;
    ST(j);
  end{ST}
end{ST}

Main
T := empty set;
ST(1);
end{Main}
Example of Depth First Search

ST(1)
Example Step 2

\{1,2\}
Example Step 3

\{1,2\} \{2,7\}
Example Step 4

\(\{1,2\} \quad \{2,7\} \quad \{7,5\}\)
Example Step 5

{1,2}  {2,7}  {7,5}  {5,4}

ST(1)  
ST(2)  
ST(7)  
ST(5)  
ST(4)
Example Step 6

{1,2} {2,7} {7,5} {5,4} {4,3}
Example Step 7

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\}

ST(1)  
ST(2)  
ST(7)  
ST(5)  
ST(4)  
ST(3)
Example Step 8

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\}

ST(1) ST(2) ST(7) ST(5) ST(4)
Example Step 9

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\}

ST(1) ST(2) ST(7) ST(5) ST(4)
Example Step 10

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\}

ST(1)  
ST(2)  
ST(7)  
ST(5)
Example Step 11

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}

ST(1) ST(2) ST(7) ST(5) ST(6)
Example Step 12

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
Example Step 13

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}

ST(1)  
ST(2)  
ST(7)  
ST(5)
Example Step 14

\{1,2\} \ {2,7} \ {7,5} \ {5,4} \ {4,3} \ {5,6}\n
ST(1)  
ST(2)  
ST(7)  

11/30/2009 CSE 373 Fall 2009 -- Dan Suciu
Example Step 15

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
Example Step 16

\{1,2\} \{2,7\} \{7,5\} \{5,4\} \{4,3\} \{5,6\}
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
Minimum Spanning Tree Problem

• Input: Undirected Graph \( G = (V,E) \) and a cost function \( C \) from \( E \) to the reals. \( C(e) \) is the cost of edge \( e \).

• Output: A spanning tree \( T \) with minimum total cost. That is: \( T \) that minimizes

\[
C(T) = \sum_{e \in T} C(e)
\]
Best Spanning Tree

• Each edge has the probability that it won’t fail
• Find the spanning tree that is least likely to fail
Example of a Spanning Tree

Probability of success = \(0.85 \times 0.95 \times 0.89 \times 0.95 \times 1.0 \times 0.84\)

\[= 0.5735\]
Minimum Spanning Tree Problem

• Input: Undirected Graph $G = (V,E)$ and a cost function $C$ from $E$ to the reals. $C(e)$ is the cost of edge $e$.

• Output: A spanning tree $T$ with minimum total cost. That is: $T$ that minimizes

$$C(T) = \sum_{e \in T} C(e)$$
Reducing Best to Minimum

Let $P(e)$ be the probability that an edge doesn’t fail. Define:

$$C(e) = -\log_{10} (P(e))$$

Minimizing $\sum_{e \in T} C(e)$

is equivalent to maximizing $\prod_{e \in T} P(e)$

because $\prod_{e \in T} P(e) = \prod_{e \in T} 10^{-C(e)} = 10^{-\sum_{e \in T} C(e)}$
Example of Reduction

Best Spanning Tree Problem

Minimum Spanning Tree Problem
Find the MST
Find the MST
Two Different Approaches

Prim’s Algorithm
Looks familiar!

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 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 $b$
Find MST using Prim’s

Start with $V_1$

Your Turn

<table>
<thead>
<tr>
<th>V</th>
<th>Kwn</th>
<th>Distance</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_4$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_5$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_6$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_7$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Order Declared Known:

$V_1$
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?*
Example of Kruskal 1

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 2

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 2

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 3

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 4

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

1 1 2 2 3 3 3 3 4

11/30/2009 CSE 373 Fall 2009 -- Dan Suciu
Example of Kruskal 5

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 3 \quad 4

1/30/2009 CSE 373 Fall 2009 -- Dan Suciu
Example of Kruskal 6
Example of Kruskal 7

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 3 4
Example of Kruskal 7
Example of Kruskal 8,9

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 4
Data Structures for Kruskal

• Sorted edge list

\{7,4\} \{2,1\} \{7,5\} \{5,6\} \{5,4\} \{1,6\} \{2,7\} \{2,3\} \{3,4\} \{1,5\}

0 1 1 2 2 3 3 3 3 4

• Disjoint Union / Find
  – Union(a,b) - union the disjoint sets named by a and b
  – Find(a) returns the name of the set containing a
Example of DU/F 1

\[
\begin{align*}
\text{Find}(5) &= 7 \\
\text{Find}(4) &= 7
\end{align*}
\]
Example of DU/F 2

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}
0 1 1 2 2 3 3 3 3 4

Find(1) = 1
Find(6) = 7
Example of DU/F 3

Union(1,7)

{7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}
0 1 1 2 2 3 3 3 4
Kruskal’s Algorithm with DU / F

Sort the edges by increasing cost;
Initialize A to be empty;
for each edge \{i,j\} chosen in increasing order do
  u := Find(i);
  v := Find(j);
  if not(u = v) then
    add \{i,j\} to A;
    Union(u,v);
Kraskal 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);
        }
    }
}
```
Find MST using Kruskal’s

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