CSE 332: Data Structures & Parallelism
Lecture 22: Minimum Spanning Trees

Ruth Anderson
Autumn 2018
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

G’ is a minimum spanning tree.

Applications:

- Example: Electrical wiring for a house or clock wires on a chip
- Example: A road network if you cared about asphalt cost rather than travel time

$$\sum_{(u,v) \in E'} c_{uv}$$ is minimal
Student Activity

Find the MST
Two Different Approaches

Prim’s Algorithm
Almost identical to Dijkstra’s

Kruskals’s Algorithm
Completely different!
**Prim’s algorithm**

**Idea:** Grow a tree by picking a vertex from the unknown set that has the smallest cost. Here cost = cost of the edge that connects that vertex to the known set. *Pick the vertex with the smallest cost that connects “known” to “unknown.”*

**A node-based greedy algorithm**

Builds MST by greedily adding nodes
Prim’s Algorithm vs. Dijkstra’s

Recall:

**Dijkstra** picked the unknown vertex with smallest cost where
\[ \text{cost} = \text{distance to the source}. \]

**Prim’s** pick the unknown vertex with smallest cost where
\[ \text{cost} = \text{distance from this vertex to the known set} \] (in other words, the cost of the smallest edge connecting this vertex to the known set)

- Otherwise identical
- Compare to slides in Dijkstra lecture!
**Prim’s Algorithm for MST**

1. For each node $v$, set $v\text{.cost} = \infty$ and $v\text{.known} = \text{false}$
2. Choose any node $v$. (this is like your “start” vertex in Dijkstra)
   a) Mark $v$ as known
   b) For each edge $(v, u)$ with weight $w$:
      set $u\text{.cost} = w$ and $u\text{.prev} = v$
3. While there are unknown nodes in the graph
   a) Select the unknown node $v$ with lowest cost
   b) Mark $v$ as known and add $(v, v\text{.prev})$ to output (the MST)
   c) For each edge $(v, u)$ with weight $w$,
      
      \[ \text{if}(w < u\text{.cost}) \{
          u\text{.cost} = w;
          u\text{.prev} = v;
      \} \]
Example: Find MST using Prim’s

Order added to known set:

<table>
<thead>
<tr>
<th>vertex</th>
<th>known?</th>
<th>cost</th>
<th>prev</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Find MST using Prim’s

Start with $V_1$

<table>
<thead>
<tr>
<th>V</th>
<th>Kwn</th>
<th>Distance</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Order Declared Known:

$V_1$

**Total Cost:**
Prim’s Analysis

• Correctness ??
  – A bit tricky
  – Intuitively similar to Dijkstra
  – Might return to this time permitting (unlikely)

• Run-time
  – Same as Dijkstra
  – $O(|E| \log |V|)$ using a priority queue
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 all vertices are not connected
   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
Aside: Union-Find aka Disjoint Set ADT

- **Union(x,y)** – take the union of two sets named x and y
  – Given sets: {3,5,7}, {4,2,8}, {9}, {1,6}
  – **Union(5,1)**
    Result: {3,5,7,1,6}, {4,2,8}, {9},
    To perform the union operation, we replace sets x and y by \((x \cup y)\)

- **Find(x)** – return the name of the set containing x.
  – Given sets: {3,5,7,1,6}, {4,2,8}, {9},
  – **Find(1)** returns 5
  – **Find(4)** returns 8

- We can do Union in constant time.
- We can get Find to be *amortized* constant time
  (worst case \(O(\log n)\) for an individual Find operation).
Kruskal’s pseudo 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);
        }
    }
}
```
**Example: Find MST using Kruskal’s**

Edges in sorted order:
1:  (A,D), (C,D), (B,E), (D,E)
2:  (A,B), (C,F), (A,C)
3:  (E,G)
5:  (D,G), (B,D)
6:  (D,F)
10: (F,G)

Output:

Note: At each step, the union/find sets are the trees in the forest
Find MST using Kruskal’s

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

Kruskal’s algorithm is clever, simple, and efficient
  – But does it generate a minimum spanning tree?
  – How can we prove it?

First: it generates a spanning tree
  – Intuition: Graph started connected and we added every edge that did not create a cycle
  – Proof by contradiction: Suppose $u$ and $v$ are disconnected in Kruskal’s result. Then there’s a path from $u$ to $v$ in the initial graph with an edge we could add without creating a cycle. But Kruskal would have added that edge. Contradiction.

Second: There is no spanning tree with lower total cost…
The inductive proof set-up

Let $F$ (stands for “forest”) be the set of edges Kruskal has added at some point during its execution.

Claim: $F$ is a subset of one or more MSTs for the graph (Therefore, once $|F|=|V|-1$, we have an MST.)

Proof: By induction on $|F|

Base case: $|F|=0$: The empty set is a subset of all MSTs

Inductive case: $|F|=k+1$: By induction, before adding the $(k+1)^{th}$ edge (call it $e$), there was some MST $T$ such that $F\setminus\{e\} \subseteq T$ …
**Staying a subset of some MST**

Claim: $F$ is a subset of *one or more* MSTs for the graph

So far:  $F\setminus\{e\} \subseteq T$:

Two disjoint cases:
- If $\{e\} \subseteq T$: Then $F \subseteq T$ and we’re done
- Else $e$ forms a cycle with some simple path (call it $p$) in $T$
  - Must be since $T$ is a spanning tree
Staying a subset of some MST

Claim: F is a subset of one or more MSTs for the graph

So far: \( F-\{e\} \subseteq T \) and \( e \) forms a cycle with \( p \subseteq T \)

- There must be an edge \( e_2 \) on \( p \) such that \( e_2 \) is not in \( F \)
  - Else Kruskal would not have added \( e \)

- Claim: \( e_2\.weight == e\.weight \)
Staying a subset of some MST

Claim: \( F \) is a subset of one or more MSTs for the graph

So far: \( F-\{e\} \subseteq T \)
- \( e \) forms a cycle with \( p \subseteq T \)
- \( e2 \) on \( p \) is not in \( F \)

• Claim: \( e2\text{.weight} == e\text{.weight} \)
  - If \( e2\text{.weight} > e\text{.weight} \), then \( T \) is not an MST because \( T-\{e2\}+\{e\} \) is a spanning tree with lower cost: contradiction
  - If \( e2\text{.weight} < e\text{.weight} \), then Kruskal would have already considered \( e2 \). It would have added it since \( T \) has no cycles and \( F-\{e\} \subseteq T \). But \( e2 \) is not in \( F \): contradiction
Staying a subset of some MST

Claim: \( F \) is a subset of one or more MSTs for the graph

So far: \( F-\{e\} \subseteq T \)
- \( e \) forms a cycle with \( p \subseteq T \)
- \( e_2 \) on \( p \) is not in \( F \)
- \( e_2.\text{weight} == e.\text{weight} \)

- Claim: \( T-\{e_2\}+\{e\} \) is an MST
  - It’s a spanning tree because \( p-\{e_2\}+\{e\} \) connects the same nodes as \( p \)
  - It’s minimal because its cost equals cost of \( T \), an MST

- Since \( F \subseteq T-\{e_2\}+\{e\} \), \( F \) is a subset of one or more MSTs
Done.