Minimum Spanning Trees

The minimum-spanning-tree problem
- Given a weighted undirected graph, compute a spanning tree of minimum weight

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.

Minimum Spanning Tree Algorithms

- Kruskal’s Algorithm for Minimum Spanning Tree construction
  - A greedy algorithm.
  - Uses a priority queue.
  - Uses the UNION-FIND technique.

- Prim’s Algorithm for Minimum Spanning Tree
  - Related to Dijkstra’s Algorithm for shortest paths.
  - Both based on expanding cloud of known vertices (basically using a priority queue instead of a DFS stack)

Kruskal’s Algorithm

Idea: Grow a forest out of edges that do not create a cycle. Pick an edge with the smallest weight.

An edge-based greedy algorithm Builds MST by greedily adding edges

Kruskal’s Algorithm Pseudocode

1. Sort edges by weight (better: put in min-heap)
2. Put each node in its own subset (of a UNION-FIND instance).
3. While output size < |V| - 1
   - Consider next smallest edge $(u,v)$
   - If $\text{find}(u)$ and $\text{find}(v)$ indicate $u$ and $v$ are in different sets
     - Output $(u,v)$
     - Perform $\text{union}\left(\text{find}(u),\text{find}(v)\right)$

Recall invariant:
- $u$ and $v$ in same set if and only if connected in output-so-far

Kruskal’s Example

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 subsets correspond to the trees in a forest.
Kruskal’s Example

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

Output: (A,D)

Note: At each step, the union/find sets are the trees in the forest

Kruskal’s Example

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: (A,D), (C,D)

Note: At each step, the union/find sets are the trees in the forest

Kruskal’s Example

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: (A,D), (C,D), (B,E)

Note: At each step, the union/find sets are the trees in the forest

Kruskal’s Example

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: (A,D), (C,D), (B,E), (D,E)

Note: At each step, the union/find sets are the trees in the forest
Kruskal's Example

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: (A,D), (C,D), (B,E), (D,E), (C,F), (E,G)

Note: At each step, the union/find sets are the trees in the forest

Kruskal's Algorithm Analysis

Idea: Grow a forest out of edges that do not grow a cycle. (This is similar to the maze-construction problem: knocking down a wall was essentially adding an edge that connected adjacent cells.)

– But now consider the edges in order by weight

So:
– Sort edges: $O(|E|\log |E|)$
– Iterate through edges using union-find for cycle detection almost $O(|E|)$

Somewhat better:
– Floyd's algorithm to build min-heap with edges $O(|E|)$
– Iterate through edges using UNION-FIND for cycle prevention and deleteMin to get next edge $O(|E|\log |E|)$
– Not better worst-case asymptotically, but often stops long before considering all edges.

Kruskal's Algorithm

List the edges in order of size:

<table>
<thead>
<tr>
<th>Edge</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED</td>
<td>2</td>
</tr>
<tr>
<td>AB</td>
<td>3</td>
</tr>
<tr>
<td>AE</td>
<td>4</td>
</tr>
<tr>
<td>CD</td>
<td>4</td>
</tr>
<tr>
<td>BC</td>
<td>5</td>
</tr>
<tr>
<td>EF</td>
<td>5</td>
</tr>
<tr>
<td>CF</td>
<td>6</td>
</tr>
<tr>
<td>AF</td>
<td>7</td>
</tr>
<tr>
<td>BF</td>
<td>8</td>
</tr>
<tr>
<td>CF</td>
<td>8</td>
</tr>
</tbody>
</table>

Kruskal's Algorithm

Select the edge with min cost

ED 2

Kruskal's Algorithm

Select the next minimum cost edge that does not create a cycle

ED 2
AB 3
Select the next minimum cost edge that does not create a cycle:

- ED 2
- AB 3
- CD 4 (or AE 4)

Next step:

- BC 5 – forms a cycle
- EF 5

All vertices have been connected. The solution is:

- ED 2
- AB 3
- CD 4
- AE 4
- EF 5

Total weight of tree: 18