Minimum-Cost Spanning Tree

- weighted connected undirected graph
- spanning tree
- cost of spanning tree is sum of edge costs
- find spanning tree that has minimum cost

Example

- Network has 10 edges.
- Spanning tree has only $n - 1 = 7$ edges.
- Need to either select 7 edges or discard 3.
Edge Selection Greedy Strategies

- Start with an \( n \)-vertex 0-edge forest. Consider edges in ascending order of cost. Select edge if it does not form a cycle together with already selected edges.
  - Kruskal’s method.

- Start with a 1-vertex tree and grow it into an \( n \)-vertex tree by repeatedly adding a vertex and an edge. When there is a choice, add a least cost edge.
  - Prim’s method.

Edge Selection Greedy Strategies

- Start with an \( n \)-vertex forest. Each component/tree selects a least cost edge to connect to another component/tree. Eliminate duplicate selections and possible cycles. Repeat until only 1 component/tree is left.
  - Sollin’s method.
Edge Rejection Greedy Strategies

- Start with the connected graph. Repeatedly find a cycle and eliminate the highest cost edge on this cycle. Stop when no cycles remain.
- Consider edges in descending order of cost. Eliminate an edge provided this leaves behind a connected graph.

Kruskal’s Method

- Start with a forest that has no edges.
- Consider edges in ascending order of cost.
- Edge (1,2) is considered first and added to the forest.
Kruskal’s Method

- Edge (7,8) is considered next and added.
- Edge (3,4) is considered next and added.
- Edge (5,6) is considered next and added.
- Edge (2,3) is considered next and added.
- Edge (1,3) is considered next and rejected because it creates a cycle.

Kruskal’s Method

- Edge (2,4) is considered next and rejected because it creates a cycle.
- Edge (3,5) is considered next and added.
- Edge (3,6) is considered next and rejected.
- Edge (5,7) is considered next and added.
Kruskal’s Method

- n - 1 edges have been selected and no cycle formed.
- So we must have a spanning tree.
- Cost is 46.
- Min-cost spanning tree is unique when all edge costs are different.

Prim’s Method

- Start with any single vertex tree.
- Get a 2-vertex tree by adding a cheapest edge.
- Get a 3-vertex tree by adding a cheapest edge.
- Grow the tree one edge at a time until the tree has n - 1 edges (and hence has all n vertices).
Sollin’s Method

- Start with a forest that has no edges.
- Each component selects a least cost edge with which to connect to another component.
- Duplicate selections are eliminated.
- Cycles are possible when the graph has some edges that have the same cost.

Sollin’s Method

- Each component that remains selects a least cost edge with which to connect to another component.
- Beware of duplicate selections and cycles.
Greedy Minimum-Cost Spanning Tree Methods

• Can prove that all result in a minimum-cost spanning tree.
• Prim’s method is fastest.
  ▪ $O(n^2)$ using an implementation similar to that of Dijkstra’s shortest-path algorithm.
  ▪ $O(e + n \log n)$ using a Fibonacci heap.
• Kruskal’s uses union-find trees to run in $O(n + e \log e)$ time.

Pseudocode For Kruskal’s Method

Start with an empty set $T$ of edges.
while (E is not empty & & |T| != n-1)
{
  Let (u,v) be a least-cost edge in E.
  E = E - {(u,v)}. // delete edge from E
  if ((u,v) does not create a cycle in T)
    Add edge (u,v) to T.
}
if (|T| == n-1) T is a min-cost spanning tree.
else Network has no spanning tree.
Data Structures For Kruskal’s Method

Edge set $E$.
Operations are:
- Is $E$ empty?
- Select and remove a least-cost edge.
Use a min heap of edges.
- Initialize. $O(e)$ time.
- Remove and return least-cost edge. $O(\log e)$ time.

Data Structures For Kruskal’s Method

Set of selected edges $T$.
Operations are:
- Does $T$ have $n-1$ edges?
- Does the addition of an edge $(u, v)$ to $T$ result in a cycle?
- Add an edge to $T$. 
Use an array linear list for the edges of T.

- Does T have \( n - 1 \) edges?
  - Check size of linear list. \( O(1) \) time.
- Does the addition of an edge \((u, v)\) to T result in a cycle?
  - Not easy.
- Add an edge to T.
  - Add at right end of linear list. \( O(1) \) time.

Just use an array rather than ArrayLinearList.

---

Does the addition of an edge \((u, v)\) to T result in a cycle?

- Each component of T is a tree.
- When \( u \) and \( v \) are in the same component, the addition of the edge \((u,v)\) creates a cycle.
- When \( u \) and \( v \) are in the different components, the addition of the edge \((u,v)\) does not create a cycle.
Data Structures For Kruskal’s Method

- Each component of $T$ is defined by the vertices in the component.
- Represent each component as a set of vertices.
  - $\{1, 2, 3, 4\}, \{5, 6\}, \{7, 8\}$
- Two vertices are in the same component iff they are in the same set of vertices.

Data Structures For Kruskal’s Method

- When an edge $(u, v)$ is added to $T$, the two components that have vertices $u$ and $v$ combine to become a single component.
- In our set representation of components, the set that has vertex $u$ and the set that has vertex $v$ are united.
  - $\{1, 2, 3, 4\} + \{5, 6\} \Rightarrow \{1, 2, 3, 4, 5, 6\}$
Data Structures For Kruskal’s Method

• Initially, $T$ is empty.

• Initial sets are:
   - $\{1\} \{2\} \{3\} \{4\} \{5\} \{6\} \{7\} \{8\}$

• Does the addition of an edge $(u, v)$ to $T$ result in a cycle? If not, add edge to $T$.

  $s1 = \text{find}(u); s2 = \text{find}(v);

  \text{if} (s1 != s2) \text{union}(s1, s2);

Data Structures For Kruskal’s Method

• Use FastUnionFind.

• Initialize.
  - $O(n)$ time.

• At most $2e$ finds and $n-1$ unions.
  - Very close to $O(n + e)$.

• Min heap operations to get edges in increasing order of cost take $O(e \log e)$.

• Overall complexity of Kruskal’s method is $O(n + e \log e)$. 