Greedy Algorithms for Minimum Spanning Tree

- [Prim] Extend a tree by including the cheapest outgoing edge
- [Kruskal] Add the cheapest edge that joins disjoint components
- [ReverseDelete] Delete the most expensive edge that does not disconnect the graph

Why do the greedy algorithms work?

- For simplicity, assume all edge costs are distinct

Proof

- Suppose $T$ is a spanning tree that does not contain $e$
- Add $e$ to $T$, this creates a cycle
- The cycle must have some edge $e_i = (u_i, v_i)$ with $u_i$ in $S$ and $v_i$ in $V-S$
- $T_i = T - \{e_i\} + \{e\}$ is a spanning tree with lower cost
- Hence, $T$ is not a minimum spanning tree

Edge inclusion lemma

- Let $S$ be a subset of $V$, and suppose $e = (u, v)$ is the minimum cost edge of $E$, with $u$ in $S$ and $v$ in $V-S$
- $e$ is in every minimum spanning tree of $G$
  - Or equivalently, if $e$ is not in $T$, then $T$ is not a minimum spanning tree

Proof

- Suppose $T$ is a spanning tree that does not contain $e$
- Add $e$ to $T$, this creates a cycle
- The cycle must have some edge $e_i = (u_i, v_i)$ with $u_i$ in $S$ and $v_i$ in $V-S$
- $T_i = T - \{e_i\} + \{e\}$ is a spanning tree with lower cost
- Hence, $T$ is not a minimum spanning tree
Optimality Proofs

• Prim’s Algorithm computes a MST
• Kruskal’s Algorithm computes a MST

• Show that when an edge is added to the MST by Prim or Kruskal, the edge is the minimum cost edge between S and V-S for some set S.

Prim’s Algorithm

\[ S = \{\}; \quad T = \{\}; \]
\[ \text{while } S \neq V \]
\[ \text{choose the minimum cost edge } e = (u,v), \text{ with } u \text{ in } S, \text{ and } v \text{ in } V-S \]
\[ \text{add } e \text{ to } T \]
\[ \text{add } v \text{ to } S \]

Prove Prim’s algorithm computes an MST

• Show an edge e is in the MST when it is added to T

Kruskal’s Algorithm

Let \( C = \{(v_1), (v_2), \ldots, (v_n)\}; \quad T = \{\} \)
\[ \text{while } |C| > 1 \]
\[ \text{Let } e = (u, v) \text{ with } u \text{ in } C_i \text{ and } v \text{ in } C_j \text{ be the minimum cost edge joining distinct sets in } C \]
\[ \text{Replace } C_i \text{ and } C_j \text{ by } C_i \cup C_j \]
\[ \text{Add } e \text{ to } T \]

Prove Kruskal’s algorithm computes an MST

• Show an edge e is in the MST when it is added to T

Reverse-Delete Algorithm

• Lemma: The most expensive edge on a cycle is never in a minimum spanning tree
Dealing with the assumption of no equal weight edges
- Force the edge weights to be distinct
  - Add small quantities to the weights
  - Give a tie breaking rule for equal weight edges

Application: Clustering
- Given a collection of points in an r-dimensional space, and an integer K, divide the points into K sets that are closest together

Distance clustering
- Divide the data set into K subsets to maximize the distance between any pair of sets
  - \( \text{dist} (S_1, S_2) = \min \{ \text{dist}(x, y) \mid x \in S_1, y \in S_2 \} \)

Divide into 2 clusters

Divide into 3 clusters

Divide into 4 clusters
Distance Clustering Algorithm

Let $C = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}; \ T = \{\}$
while $|C| > K$
  Let $e = (u, v)$ with $u$ in $C_i$ and $v$ in $C_j$ be the minimum cost edge joining distinct sets in $C$
  Replace $C_i$ and $C_j$ by $C_i U C_j$