Chapter 4
Greedy Algorithms

Motivating application

- A communication network
  - Set of locations
  - Also know the cost if we build a direct link between two locations
- Goal: build a communication network, such that there is a path between any pair of nodes, with minimum cost

- Assuming all costs are positive, the network with minimal cost has no cycles
- Connected, and no cycles—it’s a tree, called a spanning tree

Minimum Spanning Tree

**Minimum spanning tree.** Given a connected graph $G = (V, E)$ with real-valued edge weights $c_e$, an MST is a subset of the edges $T \subseteq E$ such that $T$ is a spanning tree whose sum of edge weights is minimized.

$G = (V, E)$

$T, \sum_{e \in T} c_e = 50$

Cayley’s Theorem. There are $n^{n-2}$ spanning trees of $K_n$. can’t solve by brute force
**Applications**

MST is fundamental problem with diverse applications.

- Network design.
  - telephone, electrical, hydraulic, TV cable, computer, road
- Approximation algorithms for NP-hard problems.
  - traveling salesperson problem, Steiner tree
- Indirect applications.
  - max bottleneck paths
  - LDPC codes for error correction
  - image registration with Renyi entropy
  - model locality of particle interactions in turbulent fluid flows
  - autoconfig protocol for Ethernet bridging to avoid cycles in a network
- Cluster analysis.

**Greedy Algorithms**

Kruskal’s algorithm. Start with $T = \emptyset$. Consider edges in ascending order of cost. Insert edge $e$ in $T$ unless doing so would create a cycle.

Reverse-Delete algorithm. Start with $T = E$. Consider edges in descending order of cost. Delete edge $e$ from $T$ unless doing so would disconnect $T$.

Prim’s algorithm. Start with some root node $s$ and greedily grow a tree $T$ from $s$ outward. At each step, add the cheapest edge $e$ to $T$ that has exactly one endpoint in $T$.

Remark. All three algorithms produce an MST.

**Simplifying assumption.** All edge costs $c_{e}$ are distinct.

**Cut property.** Let $S$ be any subset of nodes, and let $e$ be the min cost edge with exactly one endpoint in $S$. Then the MST contains $e$.

**Cycle property.** Let $C$ be any cycle, and let $f$ be the max cost edge belonging to $C$. Then the MST does not contain $f$.

**Cycles and Cuts**

**Cycle.** Set of edges the form $a-b$, $b-c$, $c-d$, ..., $y-z$, $z-a$.

**Cutset.** A cut is a subset of nodes $S$. The corresponding cutset $D$ is the subset of edges with exactly one endpoint in $S$.
Greedy Algorithms

Simplifying assumption. All edge costs $c_e$ are distinct.

Cut property. Let $S$ be any subset of nodes, and let $e$ be the min cost edge with exactly one endpoint in $S$. Then the MST $T^*$ contains $e$.

Proof. (exchange argument)

Suppose $e = (v, w)$ does not belong to $T^*$, and let's see what happens.

There already a path from $v$ to $w$ in $T^*$

Let $f$ be the edge in the path that crosses from $S$ to $V - S$

$T' = T^* \cup \{ e \} - \{ f \}$ is also a spanning tree.
- Still $n-1$ edges
- Connected
- Since $c_e < c_f$, cost($T'$) < cost($T^*$).
- This is a contradiction. □

Prim's Algorithm: Proof of Correctness

Prim's algorithm. [Jarník 1930, Dijkstra 1957, Prim 1959]

- Initialize $S$ = any node.
- Gradually adding nodes to $S$
- Always add to $T$ the minimum cost edge in cutset corresponding to $S$, and add one new explored node $u$ to $S$.
- According to cut property, this is optimal

Kruskal's Algorithm: Proof of Correctness

Kruskal's algorithm. [Kruskal, 1956]

- Consider edges in ascending order of weight.
- Case 1: If adding $e$ to $T$ creates a cycle, discard $e$ according to cycle property.
- Case 2: Otherwise, insert $e = (u, v)$ into $T$ according to cut property where $S$ = set of nodes in u’s connected component.
Lexicographic Tiebreaking

To remove the assumption that all edge costs are distinct: perturb all edge costs by tiny amounts to break any ties.

**Impact.** Kruskal and Prim only interact with costs via pairwise comparisons. If perturbations are sufficiently small, MST with perturbed costs is MST with original costs.

**Implementation.** Can handle arbitrarily small perturbations implicitly by breaking ties lexicographically, according to index.

```java
boolean less(i, j) {
    if      (cost(ei) < cost(ej)) return true
    else if (cost(ei) > cost(ej)) return false
    else if (i < j)               return true
    else            return false
}
```

Clustering

**Clustering.** Given a set U of n objects labeled p1, ..., pn, classify into coherent groups.

**Distance function.** Numeric value specifying "closeness" of two objects.

**Fundamental problem.** Divide into clusters so that points in different clusters are far apart.

Routing in mobile ad hoc networks.
- Identify patterns in gene expression.
- Document categorization for web search.
- Similarity searching in medical image databases
- Skycat: cluster 10^6 sky objects into stars, quasars, galaxies.

Clustering of Maximum Spacing

**k-clustering.** Divide objects into k non-empty groups.

**Distance function.** Assume it satisfies several natural properties.

- $d(p_i, p_j) = 0$ iff $p_i = p_j$ (identity of indiscernibles)
- $d(p_i, p_j) \geq 0$ (nonnegativity)
- $d(p_i, p_j) = d(p_j, p_i)$ (symmetry)

**Spacing.** Min distance between any pair of points in different clusters.

**Clustering of maximum spacing.** Given an integer k, find a k-clustering of maximum spacing.
**Greedy Clustering Algorithm**

Single-link k-clustering algorithm.
- Form a graph on the vertex set \( U \), corresponding to \( n \) clusters.
- Find the closest pair of objects such that each object is in a different cluster, and add an edge between them.
- Repeat \( n-k \) times until there are exactly \( k \) clusters.

**Key observation.** This procedure is precisely Kruskal’s algorithm (except we stop when there are \( k \) connected components).

**Remark.** Equivalent to finding an MST and deleting the \( k-1 \) most expensive edges.

---

**Greedy Clustering Algorithm: Analysis**

**Theorem.** Let \( C^* \) denote the clustering \( C^*_1, \ldots, C^*_k \) formed by deleting the \( k-1 \) most expensive edges of a MST. \( C^* \) is a \( k \)-clustering of max spacing.

**Proof.** Let \( C \) denote some other clustering \( C_1, \ldots, C_k \).
- The spacing of \( C^* \) is the length \( d^* \) of the \( (k-1) \)th most expensive edge.
- Let \( p, q \) be in the same cluster in \( C^* \), say \( C^*_r \), but different clusters in \( C \), say \( C_s \) and \( C_t \).
- Some edge \((p, q)\) on \( p-q \) path in \( C^*_r \) spans two different clusters in \( C \).
- All edges on \( p-q \) path have length \( \leq d^* \) since Kruskal chose them.
- Spacing of \( C \) is \( \leq d^* \) since \( p \) and \( q \) are in different clusters.

---

**MST Algorithms: Theory**

**Deterministic comparison based algorithms.**

- \( O(m \log n) \) \([\text{Tamnık, Prim, Dijkstra, Kruskal, Boruvka}]\)
- \( O(m \log n) \) \([\text{Cheriton-Tarjan 1976, Yao 1975}]\)
- \( O(m \beta(m, n)) \) \([\text{Fredman-Tarjan 1987}]\)
- \( O(m \log \beta(m, n)) \) \([\text{Gabow-Gall- Spencer-Tarjan 1986}]\)
- \( O(m \alpha(m, n)) \) \([\text{Chazelle 2000}]\)

**Holy grail.** \( O(m) \).

**Notable.**

- \( O(m) \) randomized. \([\text{Karger-Klein-Tarjan 1995}]\)
- \( O(m) \) verification. \([\text{Dixon-Rauch-Tarjan 1992}]\)

**Euclidean.**

- 2-d: \( O(n \log n) \). compute MST of edges in Delaunay
- k-d: \( O(k n^2) \). dense Prim
Dendrogram. Scientific visualization of hypothetical sequence of evolutionary events.
- Leaves = genes.
- Internal nodes = hypothetical ancestors.


Dendrogram of Cancers in Human

Tumors in similar tissues cluster together.

Reference: Botstein & Brown group