4.4 Shortest Paths in a Graph

shortest path from computer science department to Einstein’s house
Shortest Path Problem

**Shortest path network.**
- Directed graph $G = (V, E)$.
- Source $s$, destination $t$.
- Length $l_e = \text{length of edge } e \ (l_e \geq 0)$.

**Shortest path problem:** find shortest directed path from $s$ to $t$.

Cost of path $s-2-3-5-t = 9 + 23 + 6 + 6 = 44$. 
Dijkstra’s Algorithm

Dijkstra’s algorithm.

- Maintain a set of explored nodes $S$ for which we have determined the shortest path distance $d(u)$ from $s$ to $u$.
- Initialize $S = \{s\}$, $d(s) = 0$.
- Repeatedly choose unexplored node $v$ which minimizes $\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e$.

add $v$ to $S$, and set $d(v) = \pi(v)$. shortest path to some $u$ in explored part, followed by a single edge $(u, v)$
Dijkstra's Algorithm

Dijkstra's algorithm.

- Maintain a set of explored nodes $S$ for which we have determined the shortest path distance $d(u)$ from $s$ to $u$.
- Initialize $S = \{ s \}$, $d(s) = 0$.
- Repeatedly choose unexplored node $v$ which minimizes

\[ \pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e, \]

add $v$ to $S$, and set $d(v) = \pi(v)$.

![Diagram of Dijkstra's Algorithm](image-url)
Dijkstra's Algorithm: Proof of Correctness

**Invariant.** For each node \( u \in S \), \( d(u) \) is the length of the shortest \( s-u \) path.

**Pf.** (by induction on \( |S| \))

**Base case:** \( |S| = 1 \) is trivial.

**Inductive hypothesis:** Assume true for \( |S| = k \geq 1 \).

- Let \( v \) be next node added to \( S \), and let \( u-v \) be the chosen edge.
- The shortest \( s-u \) path plus \( (u, v) \) is an \( s-v \) path of length \( \pi(v) \).
- Consider any \( s-v \) path \( P \). We’ll see that it’s no shorter than \( \pi(v) \).
- Let \( x-y \) be the first edge in \( P \) that leaves \( S \), and let \( P' \) be the subpath to \( x \).
- \( P \) is already too long as soon as it leaves \( S \).

\[
\ell(P) \geq \ell(P') + \ell(x,y) \geq d(x) + \ell(x, y) \geq \pi(y) \geq \pi(v)
\]

- nonnegative weights
- inductive hypothesis
- defn of \( \pi(y) \)
- Dijkstra chose \( v \) instead of \( y \)
Dijkstra's Algorithm: Implementation

For each unexplored node, explicitly maintain \( \pi(v) = \min_{e=(u,v): u \in S} d(u) + \ell_e \).

- Next node to explore = node with minimum \( \pi(v) \).
- When exploring \( v \), for each incident edge \( e = (v, w) \), update \( \pi(w) = \min \{ \pi(w), \pi(v) + \ell_e \} \).

**Efficient implementation.** Maintain a priority queue of unexplored nodes, prioritized by \( \pi(v) \).

<table>
<thead>
<tr>
<th>PQ Operation</th>
<th>Dijkstra</th>
<th>Array</th>
<th>Binary heap</th>
<th>d-way Heap</th>
<th>Fib heap †</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>( n )</td>
<td>( n )</td>
<td>( \log n )</td>
<td>( d \log_d n )</td>
<td>1</td>
</tr>
<tr>
<td>ExtractMin</td>
<td>( n )</td>
<td>( n )</td>
<td>( \log n )</td>
<td>( d \log_d n )</td>
<td>( \log n )</td>
</tr>
<tr>
<td>ChangeKey</td>
<td>( m )</td>
<td>1</td>
<td>( \log n )</td>
<td>( \log_d n )</td>
<td>1</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>( n )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>( n^2 )</td>
<td>( m \log n )</td>
<td>( m \log_{m/n} n )</td>
<td>( m + n \log n )</td>
<td></td>
</tr>
</tbody>
</table>

† Individual ops are amortized bounds
The question of whether computers can think is like the question of whether submarines can swim.

Do only what only you can do.

In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, they are without precedent in the cultural history of mankind.

The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence.

APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.
4.5 Minimum 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
  - learning salient features for real-time face verification
  - reducing data storage in sequencing amino acids in a protein
  - 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 $T = \phi$. 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 the MST.
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 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 $f$. 

\[ S \]
\[ e \text{ is in the MST} \]

\[ f \text{ is not in the MST} \]
Cycles and Cuts

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

![Diagram of a cycle](image)

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

![Diagram of a cut](image)

**Example:**
- **Cycle $C$:** 1-2, 2-3, 3-4, 4-5, 5-6, 6-1
- **Cut $S$:** $\{4, 5, 8\}$
- **Cutset $D$:** 5-6, 5-7, 3-4, 3-5, 7-8
**Claim.** A cycle and a cutset intersect in an even number of edges.

**Pf.** (by picture)

---

**Cycle-Cut Intersection**

**Cycle** $C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1$

**Cutset** $D = 3-4, 3-5, 5-6, 5-7, 7-8$

**Intersection** = 3-4, 5-6
**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 $e$.

**Pf.** (exchange argument)
- Suppose $e$ does not belong to $T^*$, and let's see what happens.
- Adding $e$ to $T^*$ creates a cycle $C$ in $T^*$.
- Edge $e$ is both in the cycle $C$ and in the cutset $D$ corresponding to $S$ \( \Rightarrow \) there exists another edge, say $f$, that is in both $C$ and $D$.
- $T' = T^* \cup \{ e \} - \{ f \}$ is also a spanning tree.
- Since $c_e < c_f$, cost($T'$) < cost($T^*$).
- This is a contradiction. ■
Greedy Algorithms

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

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

Pf. (exchange argument)
- Suppose $f$ belongs to $T^*$, and let's see what happens.
- Deleting $f$ from $T^*$ creates a cut $S$ in $T^*$.
- Edge $f$ is both in the cycle $C$ and in the cutset $D$ corresponding to $S$  
  $\Rightarrow$ there exists another edge, say $e$, that is in both $C$ and $D$.
- $T' = T^* \cup \{e\} - \{f\}$ is also a spanning tree.
- Since $c_e < c_f$, $\text{cost}(T') < \text{cost}(T^*)$.
- This is a contradiction. □
Prim's Algorithm: Proof of Correctness

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

- Initialize $S = \text{any node}$.
- Apply cut property to $S$.
- Add min cost edge in cutset corresponding to $S$ to $T$, and add one new explored node $u$ to $S$. 

![Diagram of Prim's algorithm](image)
Implementation: Prim's Algorithm

Implementation. Use a priority queue ala Dijkstra.

- Maintain set of explored nodes \( S \).
- For each unexplored node \( v \), maintain attachment cost \( a[v] = \text{cost of cheapest edge} \ v \text{ to a node in} \ S \).
- \( O(n^2) \) with an array; \( O(m \log n) \) with a binary heap.

```
Prim(G, c) {
    foreach (v \in V) a[v] \leftarrow \infty
    Initialize an empty priority queue Q
    foreach (v \in V) insert v onto Q
    Initialize set of explored nodes S \leftarrow \emptyset

    while (Q is not empty) {
        u \leftarrow \text{delete min element from Q}
        S \leftarrow S \cup \{ u \}
        foreach (edge e = (u, v) incident to u)
            if ((v \not\in S) and (c_e < a[v]))
                decrease priority a[v] to c_e
    }
```
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.
Implementation: Kruskal’s Algorithm

Implementation. Use the union-find data structure.
- Build set $T$ of edges in the MST.
- Maintain set for each connected component.
- $O(m \log n)$ for sorting and $O(m \alpha (m, n))$ for union-find.

```plaintext
Kruskal(G, c) {
    Sort edges weights so that $c_1 \leq c_2 \leq \ldots \leq c_m$.
    $T \leftarrow \emptyset$
    
    foreach $(u \in V)$ make a set containing singleton $u$
    
    for $i = 1$ to $m$
    
    (u,v) = $e_i$
    
    if (u and v are in different connected components) {
        $T \leftarrow T \cup \{e_i\}$
        merge the sets containing u and v
    }
    
    return $T$
}
```

are $u$ and $v$ in different connected components?
merge two components
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. e.g., if all edge costs are integers, perturbing cost of edge $e_i$ by $i / n^2$

Use implementation detail. Can handle arbitrarily small perturbations implicitly by breaking ties lexicographically, according to index.

```java
boolean less(i, j) {
    if (cost(e_i) < cost(e_j)) return true
    else if (cost(e_i) > cost(e_j)) return false
    else if (i < j) return true
    else return false
}
```
MST Algorithms: Theory

Deterministic comparison based algorithms.

- $O(m \log n)$  
  Jarník, Prim, Dijkstra, Kruskal, Boruvka
- $O(m \log \log n)$. Cheriton-Tarjan (1976), Yao (1975)
- $O(m \beta(m, n))$. Fredman-Tarjan (1987)
- $O(m \log \beta(m, n))$. Gabow-Galil-Spencer-Tarjan (1986)
- $O(m \alpha(m, n))$. Chazelle (2000)

Holy grail. $O(m)$.

Notable.

- $O(m)$ verification. 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
4.7 Clustering

Outbreak of cholera deaths in London in 1850s.
Reference: Nina Mishra, HP Labs
Clustering

Clustering. Given a set $U$ of $n$ objects labeled $p_1, \ldots, p_n$, 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 $10^9$ 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.
**Dendrogram.** Scientific visualization of hypothetical sequence of evolutionary events.

- Leaves = genes.
- Internal nodes = hypothetical ancestors.

Tumors in similar tissues cluster together.

Reference: Botstein & Brown group
**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.

**Pf.** Let $C$ denote some other clustering $C_1, \ldots, C_k$.
- The spacing of $C^*$ is the length $d^*$ of the $(k-1)^{st}$ most expensive edge.
- Let $p_i, p_j$ 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 $p_i$-$p_j$ path in $C^*_r$ spans two different clusters in $C$.
- All edges on $p_i$-$p_j$ path have length $\leq d^*$ since Kruskal chose them.
- Spacing of $C$ is $\leq d^*$ since $p$ and $q$ are in different clusters. □