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 \( T = \emptyset \). 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.
4.4 Shortest Paths in a Graph

shortest path from Princeton CS department to Einstein's house
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.
Shortest Path Problem

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

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

Cost of path $s$-$2$-$3$-$5$-$t$

$= 9 + 23 + 2 + 16$

$= 48$. 
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 $d(v) = \pi(v)$.

\[\text{shortest path to some } u \text{ in explored part, followed by a single edge } (u, v)\]
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 $d(v) = \pi(v)$.

The diagram illustrates the process, showing the shortest path to some $u$ in the explored part, followed by a single edge $(u, v)$.
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 ( ^{\dagger} )</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>
</tbody>
</table>
| **Total**    | \( n^2 \)| \( m \log n \)| \( m \log_{m/n} n \)| \( m + n \log n \)| \n
\( ^{\dagger} \) Individual ops are amortized bounds
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 $f$. 

![Diagram showing e is in the MST and f is not in the MST]
Cycles and Cuts

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

![Graph of a cycle](image)

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

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

![Graph of a cut](image)

*Cut S = \{4, 5, 8\}*

*Cutset D = 5-6, 5-7, 3-4, 3-5, 7-8*
Cycle-Cut Intersection

Claim. A cycle and a cutset intersect in an even number of edges.

Pf. (by picture)