Menu

- Minimum Spanning Tree in a graph
- Technique: greedy approach
Minimum spanning trees

**Input:** A connected, undirected graph $G = (V, E)$ with weight function $w : E \rightarrow \mathbb{R}$
(for simplicity, assume that all edge weights are distinct)

**Output:** A *spanning tree* $T$
(a tree that connects all vertices) of minimum weight $\sum_{e \in T} w(e)$
Example of MST
Example of MST
Greedy approach
Greedy approach

- **Greedy vs Dynamic Programming**
  - **DP:** Recursive formula + overlapping sub-problems
    - I.e., there are many possible choices, and we prepare for all of them
    - E.g., “what is the optimal cost if this interval is/is not in the solution?”
  - **Greedy:** A locally optimal choice is globally optimal
    - E.g., “without loss of generality the interval that ends first must be in the solution”
    - Requires proving some property of the problem
Theorem. Let $T$ be the MST of $G = (V, E)$, and let $A \subseteq V$. Suppose that $(u, v) \in E$ is the least-weight edge connecting $A$ to $V - A$. Then, $(u, v) \in T$. 
Proof of theorem

Proof. Suppose \((u, v) \not\in T\)

\[ T: \]

\[ \in A \]

\[ \in V - A \]

\((u, v) = \text{least-weight edge connecting } A \text{ to } V - A\)
Proof of theorem

Proof. Suppose \((u, v) \notin T\)

Consider the unique simple path from \(u\) to \(v\) in \(T\).
Proof of theorem

**Proof.** Suppose \((u, v) \notin T\)

Consider the unique simple path from \(u\) to \(v\) in \(T\).
Swap \((u, v)\) with the first edge on this path that connects a vertex in \(A\) to a vertex in \(V - A\).
Proof of theorem

Proof. Suppose \((u, v) \notin T\)

Consider the unique simple path from \(u\) to \(v\) in \(T\).

Swap \((u, v)\) with the first edge on this path that connects a vertex in \(A\) to a vertex in \(V - A\).

A lighter-weight spanning tree than \(T\) results.
Prim’s Algorithm
Example of Prim’s algorithm

\[ \in A \quad \in V - A \]
Example of Prim’s algorithm

\[ \in A \]
\[ \in V - A \]
Example of Prim’s algorithm
Example of Prim’s algorithm

\[ \in A \quad \in V - A \]
Example of Prim’s algorithm
Example of Prim’s algorithm
Example of Prim’s algorithm

\[ \in A \]

\[ \in V - A \]
Example of Prim’s algorithm
Example of Prim’s algorithm

\[ \in A \]

\[ \in V - A \]
Example of Prim’s algorithm

\[ \in A \]

\[ \in V - A \]
Example of Prim’s algorithm

∈ \( A \)

∈ \( V − A \)

\[
\begin{align*}
5 & \quad \longrightarrow \quad 6 \quad \longrightarrow \quad 7 \\
3 & \quad \longrightarrow \quad 8 \\
0 & \quad \longrightarrow \quad 9
\end{align*}
\]
Example of Prim’s algorithm

\[ \in A \]

\[ \in V - A \]
Recap 1: Adjacency lists

An **adjacency list** of a vertex $v \in V$ is the list $\text{Adj}[v]$ of vertices adjacent to $v$.

\[
\text{Adj}[1] = \{2, 3\} \\
\text{Adj}[2] = \{3\} \\
\text{Adj}[3] = \{\} \\
\text{Adj}[4] = \{3\}
\]

For undirected graphs, $|\text{Adj}[v]| = \text{degree}(v)$. For digraphs, $|\text{Adj}[v]| = \text{out-degree}(v)$.

**Handshaking Lemma:** $\sum_{v \in V} \text{degree}(v) = 2|E|$ for undirected graphs

$\Rightarrow$ adjacency lists use $\Theta(V + E)$ storage
Recap 2: Priority queue

• A data structure maintaining $n$ elements $u$
• Each element $u$ has an associated value $key[u]$
  – Initially $key[u]=\infty$
• (At least) two operations:
  – EXTRACT-MIN: returns the smallest element of the queue, and removes it from the queue
  – DECREASE-KEY (or $key[u] \leftarrow w$): decreases the value of $key[u]$ to $w$
• Heap implementation:
  – Initialization: $O(n)$
  – EXTRACT-MIN + DECREASE-KEY: $O(\log n)$
Prim’s algorithm

**Idea:** Maintain $V - A$ as a priority queue $Q$. Key each vertex in $Q$ with the weight of the least-weight edge connecting it to a vertex in $A$.

$Q \leftarrow V$

$\text{key}[v] \leftarrow \infty$ for all $v \in V$

$\text{key}[s] \leftarrow 0$ for some arbitrary $s \in V$

while $Q \neq \emptyset$

    do $u \leftarrow \text{EXTRACT-MIN}(Q)$

    for each $v \in \text{Adj}[u]$

        do if $v \in Q$ and $w(u, v) < \text{key}[v]$

        then $\text{key}[v] \leftarrow w(u, v)$  ▷ **Decrease-Key**

            $\pi[v] \leftarrow u$

At the end, $\{(v, \pi[v])\}$ forms the MST.
Analysis of Prim

\[ Q \leftarrow V \]
\[ \text{key}[v] \leftarrow \infty \text{ for all } v \in V \]
\[ \text{key}[s] \leftarrow 0 \text{ for some arbitrary } s \in V \]

while \( Q \neq \emptyset \)

\[ \text{do } u \leftarrow \text{EXTRACT-MIN}(Q) \]

\[ \text{for each } v \in \text{Adj}[u] \]

\[ \text{do if } v \in Q \text{ and } w(u, v) < \text{key}[v] \]

\[ \text{then } \text{key}[v] \leftarrow w(u, v) \]
\[ \pi[v] \leftarrow u \]

Handshaking Lemma \( \Rightarrow \Theta(E) \) implicit \text{DECREASE-KEY}'s.

Time = \( \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}} \)
Analysis of Prim (continued)

\[
Time = \Theta(V) \cdot T_{\text{Extract-Min}} + \Theta(E) \cdot T_{\text{Decrease-Key}}
\]

<table>
<thead>
<tr>
<th></th>
<th>( Q )</th>
<th>( T_{\text{Extract-Min}} )</th>
<th>( T_{\text{Decrease-Key}} )</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>( O(V) )</td>
<td>( O(1) )</td>
<td></td>
<td>( O(V^2) )</td>
</tr>
<tr>
<td>binary heap</td>
<td>( O(\lg V) )</td>
<td>( O(\lg V) )</td>
<td></td>
<td>( O(E \lg V) )</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>( O(\lg V) )</td>
<td>amortized</td>
<td>( O(1) )</td>
<td>( O(E + V \lg V) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>amortized</td>
<td>worst case</td>
</tr>
</tbody>
</table>
MST algorithms

Kruskal’s algorithm:
• Uses the *disjoint-set data structure*
• Running time $= O(E \log V)$.

Best to date:
• Karger, Klein, and Tarjan [1993].
• Randomized algorithm.
• $O(V + E)$ expected time.