LEcTure 14. Optimization

Graphs, MST, Greedy, Prim

• Graph representation
• Minimum spanning trees
• Greedy algorithms hallmarks
  1. Optimal substructure
  2. Overlapping subproblems
  3. Greedy choice property
• Prim’s algorithm
• Kruskall’s algorithm

Prof. Manolis Kellis

Design and Analysis of Algorithms
6.046J/18.401J

Graphs (review)

Definition. A directed graph (digraph) $G = (V, E)$ is an ordered pair consisting of
• a set $V$ of vertices (singular: vertex),
• a set $E \subseteq V \times V$ of edges.

In an undirected graph $G = (V, E)$, the edge set $E$ consists of unordered pairs of vertices.

In either case, we have $|E| = O(V^2)$. Moreover, if $G$ is connected, then $|E| \geq |V| - 1$, which implies that $\lg |E| = \Theta(\lg V)$.

(Review CLRS, Appendix B.)

Adjacency-matrix representation

The adjacency matrix of a graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$, is the matrix $A[1 \ldots n, 1 \ldots n]$ given by

$$A[i, j] = \begin{cases} 1 & \text{if } (i, j) \in E, \\ 0 & \text{if } (i, j) \not\in E. \end{cases}$$

$$
\begin{array}{c|cccc}
  & 1 & 2 & 3 & 4 \\
\hline
1 & 1 & 0 & 1 & 1 \\
2 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 \\
\end{array}
$$

$\Theta(V^2)$ storage

$\Rightarrow$ dense representation.

Adjacency-list representation

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

$$
\begin{array}{c|c}
  & 1 & 2 \\
\hline
\text{Adj[1]} & \{2, 3\} & \{\} \\
\text{Adj[2]} & \{3\} & \{\} \\
\text{Adj[3]} & \{\} & \{\} \\
\text{Adj[4]} & \{\} & \{\} \\
\end{array}
$$

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

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

Handshaking Lemma: $\sum_{v \in V} \text{degree}(v) = 2|E|$ for undirected graphs $\Rightarrow$ adjacency lists use $\Theta(V + E)$ storage — a sparse representation (exercise B.4.1).

Minimum spanning trees

Our first problem on graphs

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. (CLRS covers the general case.)

Output: A spanning tree $T$ — a tree that connects all vertices — of minimum weight:

$$w(T) = \sum_{(u, v) \in T} w(u, v).$$
Example of MST

Two key optimization algorithm techniques:
Greedy algorithms & Dynamic Programming

Interval scheduling

<table>
<thead>
<tr>
<th>1:</th>
<th>2:</th>
<th>3:</th>
<th>4:</th>
<th>5:</th>
<th>6:</th>
</tr>
</thead>
</table>

Greedy choice: (1) Always pick interval that ends first.
(2) Never look back!

Dynamic programming:
(1) Compute all subproblem scores
(2) Pick max score
(3) Trace back to construct solution

Requires that greedy choice applies
Requires small number of subproblems

Both require optimal substructure: optimal solution to larger problem instance contains within it optimal solutions to small (sub)problem instances

Hallmarks of optimization problems:
Greedy vs. Dynamic Programming

1. Optimal substructure? (Both)
An optimal solution to a problem (instance) contains optimal solutions to subproblems.

2. Few overlapping subproblems? (DP)
A recursive solution contains a “small” number of distinct subproblems repeated many times.

3. Greedy choice? (Greedy)
Locally optimal choices lead to globally optimal solution

Do these apply to MST? Can we exploit them to solve MST efficiently?

Example of MST

Example of MST

• The Challenge: There are exponentially many paths through the graph. Finding the minimum weight using exhaustive search (brute force): $\Theta(2^{|V|})$
### 1. Optimal substructure

**MST** $T$: 
(Other edges of $G$ are not shown.)

Remove any edge $(u, v) \in T$.

Then, $T$ is partitioned into two subtrees $T_1$ and $T_2$.

**Theorem** The subtree $T_1$ is a MST of $G_1 = (V_1, E_1)$, ($G_1$ = subgraph of $G$ induced by vertices of $T_1$):
- $V_1$ = vertices of $T_1$,
- $E_1 = \{ (x, y) \in E : x, y \in V_1 \}$.

Similarly for $T_2$.

### Proof of optimal substructure

**Proof technique:** By contradiction using cut and paste argument. Show that if a better solution existed for a subpart of the optimal solution, then we could get a better optimal solution $\rightarrow$ contradiction.

**Theorem.** Subtree $T_1$ is an MST of $G_1 = (V_1, E_1)$.

**Proof:** If not MST, some $T'_1$ exists w/ lower weight

$$w(T) = w(u, v) + w(T_1) + w(T_2).$$

If $T'_1$ were a lower-weight spanning tree than $T_1$ for $G_1$, then $T' = \{(u, v)\} \cup T'_1 \cup T_2$ would be a lower-weight spanning tree than $T$ for $G$.

### Some graphs contain multiple MSTs

**Graphs with non-repeated edge weights have unique MSTs**

**Theorem:**
- If every edge weight is unique
- Then $\exists$ unique MST

**Proof:** (by contradiction)
- Suppose $\exists$ two different MSTs $A$ and $B$.
- Let $e_A$ be the edge of least weight that is in one of the MSTs and not the other
- As $B$ is a MST, $\{e_A\} \cup B$ must contain a cycle $C$.
- Then in cycle $C$ has an edge $e_B$ with $w(e_B) > w(e_A)$, by the choice of $e_A$ since all edges in $B$ with less weight are in $A$, by the choice of $e_A$.
- Thus, replacing $e_B$ with $e_A$ in $B$ yields a spanning tree with a smaller weight.
- This contradicts the assumption that $B$ is a MST.

### Unique weights vs. general weights

**MST** $T$: 
(Other edges of $G$ are not shown.)

Note: $(u, v)$ did not need to be the only min-weight edge spanning $(S, V-S)$ in proof (there may be multiple). It only needed to be some min-weight edge spanning that cut.

Proof is simpler in unique case:
- Let $A$ be a subset of $E$ that is included in some MST for $G$, let $(S, V-S)$ be any cut of $G$ that respects $A$, and let $(u, v)$ be the unique light edge crossing $(S, V-S)$.
- Then, edge $(u, v)$ MUST be included in $A$.

**Proof by contradiction (cut and paste argument):** If I remove $(u, v)$, any other edge spanning cut must be larger
2. Few overlapping subproblems?

- Space of subproblems is highly overlapping, but it’s actually exponential!
- Dynamic programming approach won’t work, as we can’t exhaustively calculate all subproblem solutions for re-use

2. Few overlapping subproblems (no \( \rightarrow \) no DP)
A recursive solution contains a “small” number of distinct subproblems repeated many times.

3. Greedy choice property

**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.** Suppose \((u, v) \notin T\). Cut and paste.

Consider the unique simple path from \( u \) to \( v \) in \( T \). Let \((u', v')\) be the first edge on this path that connects a vertex in \( A \) to a vertex in \( V - A \). By construction, \( w(u', v') > w(u, v) \), since \((u, v)\) is the least-weight edge. Thus, replacing \((u', v')\) with \((u, v)\) in \( T \) leads to \( T' \), also a spanning tree with lower weight. Hence contradiction, as \( T \) is a MST.

Note on confusing notation: \( T \) contains \((u', v')\) and \( T' \) contains \((u, v)\).

3. Greedy choice property (yes: Greedy, no: DP)
Locally optimal choices lead to globally optimal solution

Solve MST problem using greedy algorithm

Several algorithms are possible, depending on the greedy choice made

**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 \( A \).

\[
\begin{align*}
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 \\
\text{while } Q \neq \emptyset & \\
\text{do } u &\leftarrow \text{EXTRACT-MIN}(Q) \\
\text{for each } v \in \text{Adj}[u] & \\
\text{do } & \\
\text{if } v \in Q \text{ and } w(u, v) < \text{key}[v] & \\
\text{then } & \text{key}[v] \leftarrow w(u, v) \quad \text{\( \triangleright \) DECREASE-KEY} \\
\pi[v] &\leftarrow u \\
\end{align*}
\]

At the end, \( \{(v, \pi[v])\} \) forms the MST.
Example of Prim’s algorithm
Example of Prim’s algorithm

Example of Prim’s algorithm

Example of Prim’s algorithm

Example of Prim’s algorithm

Example of Prim’s algorithm

Example of Prim’s algorithm
Example of Prim’s algorithm

Analysis of Prim

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

while \( Q \neq \emptyset \)

\[
\begin{align*}
\text{degree}(u) & \text{ times} \\
\text{for each } v \in Adj[u] & \text{ do if } v \in Q \text{ and } w(u, v) < \text{key}[v] \\
& \text{ then } \text{key}[v] \leftarrow w(u, v) \quad \pi[v] \leftarrow u
\end{align*}
\]

Handshaking Lemma \( \Rightarrow \Theta(E) \) implicit DECREASE-KEY’s.

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

Analysis of Prim (continued)

MST algorithms

Prim’s algorithm:
- Single tree extended. Least-weight edge A:V-A.
- Running time = \( O(V + E \lg V) \).

Kruskal’s algorithm:
- Forests merged. Least-weight non-cycle edge.
- Running time = \( O(E \lg V) \).

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

Summary

- Graphs, MST, Greedy, Prim
- Graph representation
- Minimum spanning trees
- Greedy algorithms hallmarks
  - Optimal substructure
  - Overlapping subproblems
  - Greedy choice property
- Prim’s algorithm / Kruskal’s algorithm