Lecture 14: April 5, 2001

Today:
- Graph Representations & Algorithms
- Greedy Algorithms
  - Minimum Spanning Tree
    * Kruskal’s Algorithm
    * Prim’s Algorithm

Graph Representation

Graph $G = (V, E)$

$V$ = set of vertices
$E$ = set of edges = subset of $V \times V$

If $G$ is connected, then
$|E| \geq |V| - 1 \Rightarrow \log |E| = \Theta(\log V)$

(Note: drop $|$ inside asymptotic notation)

Graph can be directed or undirected

Assume vertices $V = \{1, 2, \ldots, n\}$
Define “Adjacency matrix” $A[1..n, 1..n]$

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

Adjacency matrix: $\Theta(V^2)$ storage $\Rightarrow$ dense representation

Sparse Graph Representation

Adjacency list: $\Theta(V + E)$ storage
$\Rightarrow$ sparse representation

For each vertex $v$, keep a list $\text{Adj}[v]$ of vertices adjacent to $v$

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

$\text{Adj}[v] = \text{degree}(v)$ [for undirected graphs]
$\text{Adj}[v] = \text{out-degree}(v)$ [for digraphs]
Handshaking Lemma

For undirected graphs,
\[ \sum \text{deg}(v) = 2|E| \]

Thus adjacency list uses \( \Theta(V + E) \) storage

Minimum Spanning Tree

Undirected, connected graph \( G = (V, E) \)
Weight function \( W : E \rightarrow \mathbb{R} \)
(Here, assume edge weights distinct)

Spanning Tree: Tree that connects all vertices
What is ST of above graph?

Minimum Spanning Tree:
Tree that connects all vertices, and minimizes
\[ w(T) = \sum_{(u,v) \in T} w(u, v) \]
What is MST of above graph?

Optimal Substructure

\( T \) (Note: some other edges omitted):

Removing \((u, v)\) partitions \( T \) into \( T_1 \) and \( T_2 \)
Claim: \( T_1 \) is MST of \( G_1 = (V_1, E_1) \), the subgraph of \( G \) induced by vertices in \( T_1 \).
\[ V_1 = \text{vertices in } T_1 \]
\[ E_1 = \{(x, y) \in E : x, y \in V_1\} \]
\( T_2 \) is MST of \( G_2 \).
\[ w(T) = w(u, v) + w(T_1) + w(T_2) \]
Can’t be a tree better than \( T_1 \) or \( T_2 \), or \( T \) would be suboptimal!
(Overlapping subproblems? Dynamic prog? Yes, but...)

Greedy Choice

Greedy choice property:
Locally optimal (greedy) choice yields a globally optimal solution!
Theorem: Let \( T \) be MST of \( G \), and let \( A \subseteq V \).
Let \((u, v)\) be min weight edge in \( G \) connecting \( A \) to \( V - A \).
Then, \((u, v) \in T \). Proof: “cut and paste”

Suppose \((u, v) \notin T \)
Look at path from \( u \) to \( v \) in \( T \)
Swap \((u, v)\) with first edge on path from \( u \) to \( v \) in \( T \) that crosses from \( A \) to \( V - A \).
This improves \( T \)!
Kruskal’s algorithm for MST

Disjoint-set data structure
Sets \( S = \{ S_i \}, S_i \) intersects \( S_j = \) empty set
Operations:

- **Insert(x)**: \( S \leftarrow S \cup \{ \{ x \} \} \)
- **Union\( (S_i, S_j)\)**: \( S \leftarrow S - \{ S_i, S_j \} \cup \{ S_i \cup S_j \} \)
- **FindSet(x)**: returns unique \( S_i \in S \) where \( x \in S_i \)

\[ T \leftarrow \text{empty set} \]
for each \( v \in V \)
\[ \text{do Insert}(v) \]
Sort \( E \) by edge weight
for each edge \( (u, v) \in E \)
\[ \text{do if } \text{FindSet}(u) \neq \text{FindSet}(v) \]
\[ \text{then } T \leftarrow T \cup \{(u, v)\} \]
\[ \text{Unite}(\text{FindSet}(u), \text{FindSet}(v)) \]

That is, adds cheapest edge that connects
two trees of “forest”.
Why is this algorithm correct?

**Example**

![Graph Example 1]

![Graph Example 2]

**Example (cont.)**

![Graph Example 3]

![Graph Example 4]
Kruskal’s Running Time

Sort: \( \Theta(E \times \log E) = \Theta(E \times \log V) \), since \( |E| \leq |V^2| \)
\( \Theta(E) \) calls to Insert
\( \Theta(V) \) calls to FindSet
\( \Theta(V) \) calls to Union
Thus latter part takes \( \Theta(E \times \alpha(E, V)) \) time using best

data structure (CLR, Ch. 22)
\( m \) operations on \( n \) sets require \( \Theta(m \cdot \alpha(m, n)) \) time.
\( \alpha(m, n) \) is “functional inverse” of Ackermann’s \( \text{fn.} \)
\( \alpha(m, n) \leq 4 \) even for \( m, n = 10^{80} \) (atoms in the un-
iverse)
\( \alpha(m, n) \) grows very slowly, but not constant!

Overall running time \( \Theta(E \log E) \).
Best MST to date: 1993 Karger, Klein, Tarjan
\( \Theta(E) \) time randomized (uses fast MST test)

---

Example

\[
\begin{bmatrix}
\text{array} & T(\text{Extract-Min}) & T(\text{Decrease-Key}) \\
\text{binary heap} & \log(V) & \log(V) \\
\text{Fibonacci heap} & \log(V) & \log(V + E)
\end{bmatrix}
\]

Time = \( |V| \times T(\text{Extract-Min}) + \)
\( \Theta(E) \times T(\text{Decrease-Key}) \)