Honors Algorithms
G22.3520-001 Fall 2006

Lecture 16
Read: CLRS 23
Minimum Spanning Trees

The problem:

- Input: a weighted, connected, undirected graph $G = (V, E)$
  
  Each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$

- Output: a subset $T \subseteq E$ such that
  
  - $(V, T)$ is a tree (acyclic and connected)
  
  - $w(T) := \sum_{e \in T} w(e)$ is minimized
Theorem. Let $G = (V, E)$ be an undirected graph. The following are equivalent.

1. $G$ is a tree
2. every pair of vertices in $G$ is connected by a single path, which is simple
3. $G$ is connected, but removing any edge makes it unconnected
4. $G$ is acyclic, but adding any edge makes it cyclic
5. $G$ is connected and $|E| = |V| - 1$
6. $G$ is acyclic and $|E| = |V| - 1$
A generic MST algorithm

- Suppose $A \subseteq E$ is contained in some MST
  
  An edge $e \in E$ is called **safe for** $A$ if $A \cup \{e\}$ is also contained in some MST

- Generic MST algorithm:
  
  $A \leftarrow \emptyset$
  
  repeat $|V| - 1$ times:
  
  find $e \in E \setminus A$ that is safe for $A$
  
  $A \leftarrow A \cup \{e\}$
Recognizing safe edges

- **Definition:**
  - A cut $C$ is a partition $(S, V \setminus S)$, where $\emptyset \subsetneq S \subsetneq V$
  - An edge $e \in E$ crosses a cut $C = (S, V \setminus S)$ if one endpoint of $e$ lies in $S$, and the other lies in $V \setminus S$
  - A cut $C$ respects $A \subseteq E$ if no edge in $A$ crosses $C$
Cut Lemma:

- Let $G = (V, E)$ be a connected, undirected graph with weights $w : E \rightarrow \mathbb{R}$

- Let $A \subseteq E$ be a subset of some MST

- Let $C$ be a cut that respects $A$

- Let $e \in E$ be an edge of smallest weight that crosses $C$

- Then: $e$ is safe for $A$
• **Proof:**

  – Let $T$ be an MST containing $A$
  – If $e \in T$, we’re done, so assume $e \notin T$
  – Goal: construct an MST $T' \supseteq A \cup \{e\}$
  – Let $e = \{u, v\}$
  – Consider the unique path $p$ from $u$ to $v$ in $T$ (which is simple)
  – Since $e$ crosses the cut $C$, there must be some $e'$ along $p$ that crosses $C$
  – Set $T' := (T \setminus \{e'\}) \cup \{e\}$
  – Want to show: $T'$ is an MST that includes $A$
• Proof (cont’d):

- $(V, T')$ is a tree
  \[ |T'| = |V| - 1 \] and $(V, T')$ is connected
- $T' \supseteq A$
  C respects $A$, $e'$ crosses $C \Rightarrow e' \not\in A$
- $T'$ is an MST
  Both $e$ and $e'$ cross $C \Rightarrow w(e) \leq w(e')$
- QED
Kruskal’s MST algorithm

\( \text{MST-Kruskal}(G, w) \)

\[ A \leftarrow \emptyset \]
for each \( v \in V \) do: \( \text{MakeSet}(v) \)  
// union-find
sort edges \( E \) in order of increasing weight:
\[ e_1, \ldots, e_m \]
for \( i \leftarrow 1 \) to \( m \) do
let \( e_i = \{u, v\} \)
\( \tilde{u} \leftarrow \text{Find}(u) \)
\( \tilde{v} \leftarrow \text{Find}(v) \)
if \( \tilde{u} \neq \tilde{v} \) then
\[ A \leftarrow A \cup \{e_i\} \]
\( \text{Union}(\tilde{u}, \tilde{v}) \)
return \( A \)
Correctness of Kruskal

- Loop invariants:
  - $(V, A)$ is a forest of trees
  - each processed edge connects two nodes in the same tree

- Upon termination: $(V, A)$ is a tree
  - follows from loop invariant and connectedness of $G$
When we add an edge $e = \{u, v\}$:

- Define the cut is $C = (S, V \setminus S)$, where $S$ is the set of nodes comprising $u$’s current tree
- $C$ respects $A$
- Because edges are sorted, $e$ has minimum weight among all nodes crossing $C$

- Cut Lemma $\Rightarrow e$ is safe for $A$
Running time of Kruskal

- Sorting $|E|$ edges: $O(|E| \log |E|)$, which is $O(|E| \log |V|)$
- $2|E|$ finds and $|V| - 1$ unions on $|V|$ items: $O(|E| \log^* |V|)$
- Total time: $O(|E| \log |V|)$
Prim’s MST algorithm

Idea:
- Start with an arbitrary node $r$, and “grow” an MST from $r$, one edge at a time
- We use two arrays, $d$ and $\pi$, indexed by $V$. Let $v$ be a node that is not in the current tree.
  - If there is an edge connecting $v$ to the current tree:
    * $d[v]$ is the weight of a minimum weight edge $\{u, v\}$ that connects $v$ to the tree
    * $\pi[v] = u$
    $v$ is called a “fringe” node
  - Otherwise, $d[v] = \infty$
Idea (cont’d):

- At each step of the algorithm:
  - choose a “fringe” node $u$ with minimum $d[u]$ value
  - add $u$ and the edge $\{u, \pi[u]\}$ to the tree
  - for each node $v$ adjacent to $u$, update $d[v]$ and $\pi[v]$ appropriately:
    
    * if $v$ is not a tree vertex, and $w(\{u, v\}) < d[v]$, then set $d[v] \leftarrow w(\{u, v\})$ and $\pi[v] \leftarrow u$
Correctness follows from Cut Lemma
Implementation of Prim

- Fringe nodes are stored in a priority queue
- Priority queue supports Decrease (as well as Insert and ExtractMin):
  - $|V|$ Insert’s
  - $|V|$ ExtractMin’s
  - $|E|$ Decrease’s
Priority Queue Implementations:

- Simple list:
  - *Insert, Decrease*: $O(1)$, *ExtractMin*: $O(|V|)$
  - Total: $O(|V|^2 + |E|)$, which is $O(|V|^2)$

- Binary Heap:
  - *Insert, Decrease, ExtractMin*: $O(\log |V|)$
  - Total: $O(|E| \log |V|)$

- Fibonacci Heap:
  - *Insert, Decrease*: $O(1)$,  
    *ExtractMin*: $O(\log |V|)$
  - Total: $O(|V| \log |V| + |E|)$