Lecture #9

CS 170  
Spring 2021
Introduction to Minimum Spanning Trees (MSTs)

Given an undirected graph $G=(V,E)$ with edge weights $w(e) > 0$

Find a subset $T \subseteq E$ such that
1. $(V,T)$ connected
2. sum of weights of $T = \sum_{e \in T} w(e)$ minimized

Fact: $T$ has no cycles, i.e. a tree

What would be a greedy algorithm?

add cheapest edge to $T$ as long as no cycle
Properties of Trees

Def: An undirected graph $T(V, E)$ is a tree if
1) $T$ is connected, and 2) $T$ has no cycles

Note: Any vertex could be root

Claim: Any 2 of following 3 properties implies the 3rd:
1) $T$ is connected
2) $T$ has no cycles
3) $|E| = |V| - 1$

Proof: 1) and 2) $\Rightarrow$ 3)

pick any vertex of to be root, run DFS,
Every vertex has one parent, except root
$\Rightarrow |E| = |V| - 1$

2) + 3) $\Rightarrow$ 1) start with no edges, $|V|$ disconnected
vertices $\Rightarrow |V|$ connected comps. Add an edge
either # comp comp drops by 1, or get a cycle $\Rightarrow |E| = |V| - 1$ 2
Cuts in a graph

Def: A cut in \( G(V, E) \) is a partition \( V = S \cup (V \setminus S) \).

Also refers to edges connecting \( S \) and \( V \setminus S \).

Claim: Lightest edge (smallest \( w(e) \)) in a cut appears in some MST.

Proof: Let \( T \) be a MST, \( e = (u, v) \) be light edge connecting \( S \) and \( V \setminus S \), \( e' \in T \) connects \( S \) and \( V \setminus S \), but \( e' \neq e \) (\( e' \) not unique).

Wand MST containing \( e \): \( T \cup \{e\} \Rightarrow \text{too many edges} \Rightarrow \text{cycle} \).

Choose edge \( e' = \text{be edge in cycle in } T \) connecting \( S \) and \( V \setminus S \).

\( T' = T \cup \{e\} \setminus \{e'\} : \text{Claim } T' \text{ is a tree} \) (connected)

\( w(T') = \sum_{e \in T'} w(e) = w(T) + w(e) - w(e') \)

\( \Rightarrow T' \text{ a MST} \) since \( w(e) \leq w(e') \).
How to add one more edge to a partial MST

Claim: Suppose $X \subseteq E$ and $X \subseteq T$ where $T$ is some MST.
Suppose $X$ has no edges connecting $S$ and $V \setminus S$ and $e$ is lightest edge connecting $S$ and $V \setminus S$.
Then $X \cup \{e\} \subseteq T'$ where $T'$ is some MST.

Proof: 

Proof:

$T' = T \cup \{e\} \setminus \{e\}$

has cycle

breaks cycle

is MST by same argument
MST Algorithm

Meta-Algorithm:

\[ X = \emptyset \]

repeat

pick cut \((S, V \setminus S)\) s.t. \(X\) doesn't cross cut

add edge \(e\) with smallest weight in cut to \(X\)

until \(|V|-1\) edges added (or graph connected)

Kruskal

\[ X = \emptyset \]

sort all \(e\) by \(w(e)\)

for all \(e\) in increasing order

if \(X \cup \{e\}\) has no cycle, \(X = X \cup \{e\}\)
Kruskal's Algorithm is Correct

\[ \begin{align*}
X &= \emptyset \\
\text{sort all } e \in E \text{ by } w(e) \\
\text{for all } e \in E \text{ in increasing order} \\
\text{If } X \cup \{e\} \text{ has no cycle, } X &= X \cup \{e\}
\end{align*} \]

Claim: At any point, \( X \) is a subset of some MST \( T \)

Proof: Induction base case: adding first edge ok

If Kruskal adds \( e = (u, v) \) to \( X \),
no cycle in \( X \cup \{e\} \)
\( S = \) connected comp. of \( u \) in \( X \)

Can there be a lighter edge \( e' \) connecting \( S \) and \( V \setminus S \)?
If there were, would have been considered already and not chosen it, contradiction because it also would not have created cycle \( \Rightarrow w(e') \geq w(e) \)

by Slide 4, \( X \cup \{e\} \subseteq \text{some MST } T \)
Implementing Kruskal (1) $\text{Cost}$

\[ X = \emptyset, \text{sort all } e \in E \text{ by } \omega(e) \]

for all $e \in E$ in increasing order

\[ \text{if } X \cup \{ e \} \text{ has no cycle,} \]

\[ X = X \cup \{ e \} \]

$X = \emptyset$, sort $e$

for all $v \in V$, make $\text{set}(v)$ ... each set is a conn. comp

for all $e$ in order ... $e = (u, v)$

\[ \text{if } \text{find}(u) \neq \text{find}(v) \ldots \text{find}(v) = \text{name of } u \text{'s conn. comp} \]

\[ \text{union}(u, v) \ldots \text{merge conn. comps of } u \text{ and } v \]

$\text{Cost: } |V| \cdot \text{cost}(\text{make set}) = O(|V|)$

\[ |E| \cdot \text{cost}(\text{find}) = O(|E| \log |V|) \]

\[ |E| \cdot \text{cost}(\text{union}) = O(|E| \log |V|) \]

\[ \frac{1}{2^{2^n} \log \left( \frac{1}{2^{2^n}} \right)^2 \log n} \leq 1 \]

\[ \log^2(n) = \# \log^3 \]
Implementing Kruskal (2)

For each \( v \in V \) add:

\[ \pi(v) = \text{"parent of v"} = \text{pointer to parent in tree} \]

\[ \text{defining connected component to which } v \text{ belongs} \]

\[ \text{rank}(v) = \text{height of subtree rooted at } v \]

\( \forall v \in V \) makeset(\( v \)): \( \pi(v) = v \), \( \text{rank}(v) = 0 \)

find(\( v \)) while \( \pi(v) \neq v \), \( v = \pi(\pi) \), return \( v \)

union(\( u, v \))

\[ \Rightarrow \text{all trees have depth } O(\log |V|) \]

\[ \Rightarrow \text{find, union each cost } O(\log |V|) \]

ranks all fit in in some set

\[ \{13, 923, 3343, 55... 163, 317, ... , 2^{63}, 3 \cdot 2^{43}, ..., 2^{26} \cdot 3 \} \]

\[ \approx 65536 \]
Implementing Kruskal (3)

Even better: when doing find, make all vertices on path to root point to root

\[
\text{find}(v) \\
\text{if } v \neq \pi(v) \quad \pi(v) = \text{find}(\pi(v)) \\
\text{return } \pi(v)
\]

all paths to root keep getting shorter

\[\Rightarrow O(1E1 \cdot \log^* |V|) \text{ cost of all finds and unions}\]
**MST Algorithm**

**Meta-Algorithm:**

\[ X = \emptyset \]

repeat

pick cut \( (S, V \setminus S) \) s.t. \( X \) does not cross cut

add edge \( e \) with smallest weight in cut to \( X \)

until \( |V| - 1 \) edges added (or graph connected)

**Prim:** \( S = \) vertices touched by \( X \)

**Kruskal:** \( S = \) connected comp of \( v \) in \( X \)

---

**Graph Diagram**

**Table**

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0/nil</td>
<td>5/A</td>
<td>6/A</td>
<td>4/A</td>
<td>add</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>add</td>
<td>add</td>
<td>add</td>
<td>add</td>
<td>add</td>
<td>add</td>
<td>add</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2/D</td>
<td>2/D</td>
<td>4/H</td>
<td>4/H</td>
<td>4/H</td>
<td>4/H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>nil</td>
</tr>
</tbody>
</table>

**Notes**

- **Tree**: Green edges
- **Path**: Blue edges
- **Cut**: Red edges
Implementing Prim

for all \( v \in V \)

\[ \text{cost}(u) = \infty, \quad \text{prev}(u) = \text{nil} \]

pick any initial \( u_0 \), \( \text{cost}(u_0) = 0 \)

\[ H = \text{makequeue}(V) \quad \ldots \quad \text{priority queue, based on cost}() \]

while \( H \neq \emptyset \)

\[ v = \text{deletemin}(H) \quad \ldots \quad \text{pick } v \text{ with lowest } \text{cost}() \]

for each \( (v, u) \in E \)

if \( \text{cost}(u) > \omega(v,u) \)

\[ \text{cost}(u) = \omega(v,u) \]

\[ \text{prev}(u) = v \]

\( \text{cost}(\text{Prim}) = \text{cost}(\text{Dijkstra}) \)

only difference is value used by priority queue