Min. Weight Spanning Tree

Algorithms

- Connected graph
  - Spanning tree can be found in many ways defined in prev lecture
  - Weighted (connected) graph: each edge associated with nonnegative real #
  - Minimum weight spanning tree: if you wanted maximum you could multiply weights by -1 and still use the algorithms to find them
  - Prim's Algorithm, Kruskal Algorithm
  - there is an optimal algorithm but we don't know what it is

Kruskal's Algorithm (in book)

- order the edges by increasing weight
- at each stage you choose next edge that doesn't make a cycle with the edges you've already chosen
- book says "marking" edges rather than "choose"

1. Pick UV

2. Pick XY

3. Pick UV
4. Next lowest is 4 but if you pick u0 you'd have a cycle
So, pick u1

\[ \begin{array}{c}
4 \\
+2 \\
+3 \\
\end{array} \]

This is the minimum weight spanning tree.

This algorithm can produce different trees, it has no tie-breaking system.
There can be a large # of spanning trees all having the same weight.
Implement cycle detection.

Theorem: Kruskal's algorithm gives a minimum weight spanning tree.

Proof: G is a connected, weighted graph. T is produced by K's algorithm.
T is a spanning tree.
Edges by Kruskal's algorithm in order:
e_1, e_2, \ldots, e_{n-1} (tree of order n has n-1 edges)
with \( w(e_1) \leq w(e_2) \leq \ldots \leq w(e_{n-1}) \)
\( w(T) = \sum_{i=1}^{n-1} w(e_i) \)

Suppose T isn't minimal and let H be a minimum weight spanning tree that has maximum number of edges in common with T.

T \# H. Let e_i be first edge of T that's not in H. e_1, \ldots, e_{i-1} are both in T & H. Let \( G_0 = H + e_i \) This has a cycle (involving e_i), C. C has an
edge that's not in \( T \). So, \( G_0 = e_0 \rightarrow T_0 \) is another spanning tree
\[ T_0 = H + e_i - e_0 \]
\[ w(T_0) = w(H) + w(e_i) - w(e_0) \geq w(H) \]
So, \( w(e_i) - w(e_0) \geq 0 \)
\[ w(e_i) \geq w(e_0) \]
If \( i > 1 \), \( e_0 \) doesn't make a cycle with \( e_1, \ldots, e_{i-1} \). Since Kruskal's algorithm chooses \( e_i \) next, must have \( w(e_i) \leq w(e_0) \)
\[ w(T_0) = w(H) \]
So, \( T_0 \) is a minimum weight spanning tree with more edges in common with \( T \) than \( H \) has. This is a contradiction so \( T \) is a minimum weight spanning tree.

**Theorem:** \( G = (V, E) \) weighted graph
\( U \subseteq V \) and \( e \) is an edge with one end vertex in \( U \) & the other in \( V-U \) having of minimum weight then there is a minimum weight spanning tree containing \( E \).

4. Important result of Prim's Algorithm (in book)

**Proof:** Let \( T \) be a minimum weight spanning tree for \( G \). If \( e \) is not an edge of \( T \). Consider \( H = T + e \). This has a cycle containing \( e \) & the cycle must also contain an edge \( f \) w/ \( u \in U \) & \( v \not\in U \)
\[ w(e) \leq w(f) \]
So, \( H - f = T + e - f \) is a spanning tree of weight \( w(T) + w(e) - w(f) \)
Since \( T \) is a min weight spanning tree \( w(e) = w(f) \)
and $T + e - f$ is a min. weight spanning tree containing $e$

Next section: (won't be on test probably)
- no good formula for # non-isomorphic trees of order $n$
- formula for # of labeled trees of order $n$

\[
\text{vertices } v_1, \ldots, v_n
\]

\[
v_1 \rightarrow v_2 \rightarrow v_3 \Rightarrow v_1 \rightarrow v_2 \rightarrow v_3
\]

isomorphic but different labeled trees because set of edges different

\[n^{n-2} \rightarrow \text{Cayley's formula}\]

discovered by mathematician in 1860
Cayley extended work in 1881
10+ different proofs, one in book