Section 3.2

Minimum Weight Spanning Trees

KRUSKAL’S ALGORITHM

Algorithm 3.2.1 Kruskal’s Algorithm.

Input: A connected weighted graph $G = (V, E)$.
Output: A minimum weight spanning tree $T = (V, E(T))$.
Method: Find the next edge $e$ of minimum weight $w(e)$ that does not form a cycle with those already chosen.

KRUSKAL’S ALGORITHM (CONCLUDED)

1. Let $i \leftarrow 1$ and $T \leftarrow \emptyset$.
2. Choose an edge $e$ of minimum weight such that $e \notin E(T)$ and such that $T \cup \{e\}$ is acyclic.
   If no such edge exists,
   then stop;
   else set $e_i \leftarrow e$ and $T \leftarrow T \cup \{e_i\}$.
3. Let $i \leftarrow i + 1$, and go to step 2.
KRUSKAL’S ALGORITHM PRODUCES A MINIMUM WEIGHT SPANNING TREE

**Theorem 3.2.1:** When Kruskal’s algorithm halts, $T$ induces a minimum weight spanning tree.

GREEDY ALGORITHMS

- A *greedy* algorithm is an algorithm that proceeds by selecting the choice that looks best at the moment.
- Kruskal’s algorithm is greedy.
- Sometimes greedy algorithms can be arbitrarily bad.

A THEOREM ON MINIMUM WEIGHT SPANNING TREES

**Theorem 3.2.2:** Let $G = (V, E)$ be a weighted graph. Let $U \subseteq V$ and let $e$ have minimum weight among all the edges from $U$ to $V - U$. Then there exists a minimum weight spanning tree that contains $e$. 
PRIM'S ALGORITHM
Algorithm 3.2.2 Prim's Algorithm.

Input: A connected weighted graph \( G = (V, E) \) with \( V = \{v_1, v_2, ..., v_n\} \).

Output: A minimum weight spanning tree \( T \).

Method: Expand the tree \( T \) from \( v_1 \) using the minimum weight edge from among \( T \) to \( V - V(T) \).

PRIM'S ALGORITHM (CONCLUDED)

1. Let \( T \leftarrow \{v_1\} \).
2. Let \( e = tu \) be an edge of minimum weight joining a vertex \( t \) of \( T \) to a vertex \( u \) of \( V - V(T) \) and set \( T \leftarrow T \cup \{e\} \).
3. If \( |E(T)| = p - 1 \) then halt, else go to step 2.