Algorithms in Systems Engineering
IE170

Lecture 18

Dr. Ted Ralphs

References for Today's Lecture

- Required reading
  - CLRS Chapter 23
- References
  - R. Sedgewick, *Algorithms in C++* (Third Edition), 1998.

Spanning Trees

- Given a connected undirected graph $G = (V, E)$, a *spanning tree* T of G is a subgraph that is a tree and whose vertex set is all of V.
- Since the vertex set of any such spanning tree is V, we will sometimes equate the edge set of a spanning tree with the spanning tree itself.
- Every minimal connected subgraph is a spanning tree (and vice versa).
- In other words, a subgraph is a spanning tree if and only if it is connected and removing any edge will disconnect it.
- If we are looking for the most inexpensive set of links that connect a set of geographically dispersed points, we want a spanning tree.
- Spanning trees arise frequently in applications, especially those with a network design component.
- They also arise in other applications, such as the design of integrated circuits.

Minimum Weight Spanning Trees

- Consider an undirected graph $G = (V, E)$ with weight vector $w \in \mathbb{R}^E$.
- If $T \subseteq E$ is a spanning tree of G, the *weight* of T is

$$\sum_{e \in T} w_e$$

- The *minimum weight spanning tree* (MST) problem is that of finding, among all spanning trees of G, one that has minimum weight.
- How many spanning tree are there?
- What about simply enumerating all of them?

Finding a Minimum Weight Spanning Tree

- Although finding an MST may seem to be a difficult problem, it can be solved efficiently.
- The first algorithm we'll consider uses another variant of graph search to build a search tree that is guaranteed to be an MST.
- We build the tree up, adding one vertex at a time.
- At each iteration, we have a partially completed tree that spans the vertices that have been processed so far.
- The vertex that is processed next is the one that is "closest" to the partially completed tree.
- The algorithm is almost identical to Dijkstra's Algorithm.

Algorithm Summary: Prim's Algorithm

- We are given a connected undirected weighted graph $G = (V, E)$ and we want to find an MST of G.
- Prim's Algorithm
  - Arbitrarily choose a source node r.
  - Initialize by assigning $d(r) = 0$ for the source node and $d(v) = \infty$ for all other nodes $v \in V \setminus \{r\}$.
  - Place r on the list L of unprocessed nodes.
  - While L is not empty
    * Choose $v \in L$ such that $d(v) = \min_{u \in L} d(u)$.
    * For each neighbor x of v, set $d(x) = \min\{d(x), w_{\{v,x\}}\}$.
- When we're finished, the search tree will be an MST.
- Why is this algorithm correct?
- How do we implement it?
- What is the running time?