8.4 Minimum Spanning Tree

The minimum spanning tree (MST) of a weighted connected graph is a subgraph that contains all of the nodes of the original and a subset of the edges so that the subgraph is connected and the total of the edge weights is the smallest possible.

- Brute force way to find MST:
  - look at all possible subsets of edge set until we find MST
  - very time consuming...
  - if there are N edges, there are $2^N$ subsets
    - for each subset, must check that it spans all nodes and has no cycles
    - then we could calculate total weights

8.4.1 Dijkstra-Prim Algorithm

- MST algorithm developed in the late 1950s
- published independently
- to find MST, will use "greedy" algorithm
  - "greedy" algorithm- work by looking at a subset of the larger problem and making the best decision based on that subset of info
  - in this case, we will, at each step of the process, examine the collection of potential edges to add to the spanning tree and pick the one with the smallest weight
  - by doing this repeatedly, we will grow a spanning tree that has a minimum overall total
- need to confer the nodes of the graph to be in one of three categories:
  - in the tree
  - on the fringe of the tree
  - not yet considered
- begin by picking one node of the graph and putting that into the spanning tree
  - then place all nodes connected to this initial one into the fringe category
  - loop through the process of picking the smallest weighted edge connecting a tree node with a fringe node
    - adding new node to the tree and then updating the nodes in the fringe category.
    - when all nodes have been added to the tree, work is done.

Algorithm:

- select a starting node
- build initial fringe from nodes connected to starting node
- while nodes left:
  - choose the edge to fringe w/ smallest weight
  - add associated node to tree
  - update fringe by:
    - adding nodes to the fringe connected to the new node
    - updating the edges to the fringe so that they are the smallest

Example:
**FIGURE 8.5C**
Second node added. Edges to nodes D, E, and G updated. (Solid lines show edges in the tree.)

**FIGURE 8.5D**
Third node added. Edge to node G updated.

**FIGURE 8.5E**
Node C added to the tree

**FIGURE 8.5F**
Node F added to the tree. Edges to nodes D and G updated.
8.4.2 Kruskal Algorithm

- Instead of focusing on nodes, Kruskal algorithm focuses on edges.
- Begin with an empty spanning tree.
- Add edges in increasing weight until all nodes are connected.
- If we run out of edges before all nodes are connected,
  - Original graph was not connected.
  - We will have generated MSTs for each subgraph.
- Must avoid cycles!!

Algorithm:
- Sort edges in non-decreasing order by weight.
- EdgeCount = 1.
- IncludedCount = 0.
- While EdgeCount <= |E| and IncludedCount <= N-1:
  - N1 = findRoot(Edge[EdgeCount].start).
  - N2 = findRoot(Edge[EdgeCount].end).
  - If N1 ≠ N2:
    - Add Edge[EdgeCount] to spanning tree.
    - IncludeCount++.
    - Union(N1, N2).
  - EdgeCount++.

**NOTE:**
- findRoot( ) checks for cycles.
- Union( ) -> combines two partitions into one.

Example:
8.5 Shortest Path Algorithm

- given two nodes
  - SP algorithm will find series of edges w/ smallest total weight
  - not necessarily MST...
    - consider:

    ![Diagram showing shortest path from A to B.]

8.5.1 Dijkstra's algorithm

- modify the MST algorithm to choose edge to the fringe that is part of the shortest entire path from the starting node
- algorithm:
  - select a starting node
  - build the initial fringe set of nodes connected to starting node
  - while not at destination:
    - choose fringe node w/ shortest path to starting node
    - add that node and edge to tree
    - update the fringe:
      - add nodes to fringe connected to new node
      - for each node in fringe:
        - update edge to one connected to tree on shortest path to starting node

- example: shortest path from A to G
**Figure 8.8A**
The original graph

**Figure 8.8B**
The shortest path is to node B

**Figure 8.8C**
Path of length 4 to node C is the shortest of the options

**Figure 8.8D**
The path of length 5 to either node E or node F is shortest

**Figure 8.8E**
The other path of length 5 to node F is next

**Figure 8.8F**
The path of length 6 to node D is shorter than the path to node G
**FIGURE 8.8G**
The path to node G is the only one left.

**FIGURE 8.8H**
The complete shortest path tree starting at node A.