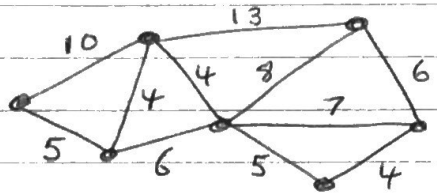
11.5 Minimum Spanning Trees

Going back to our towns in Maine, we can show the road lengths, in miles, between them:



So this is a weighted graph. After another snow storm, what is the shortest length of road that can be plowed to keep a path between any two towns?

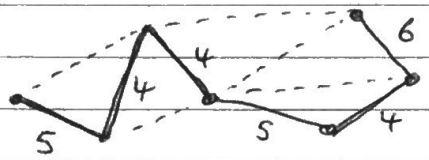
Definition. A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

To solve the snow plow question we need a minimum spanning tree. We saw that shortest path problems can be complicated (remember Dijkstra's algorithm). Luckily there are two simple algorithms to find minimum spanning trees.

The first is Prim's algorithm. Let G be a connected weighted graph with n vertices. Pick an edge with smallest weight. Add edges of smallest possible weight that are connected to a vertex already in the tree you have built. Make sure there are no circuits. Stop when you have added n-1 edges.

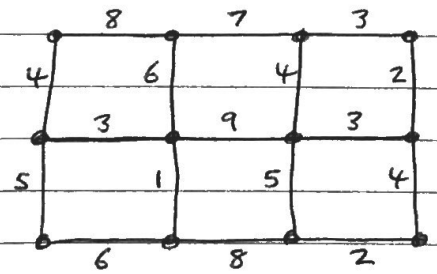
We see that Prim's is a greedy algorithm adding a smallest weight edge in each step.

For our Maine roads, start with one of the 4s and build up the tree to get



This is a minimum spanning tree and only 28 miles of road need to be plowed.

Example ② Use Prim's algorithm to find a minimum spanning tree for this graph



Try it, you should get a minimum weight of 40.

The second algorithm to find minimum spanning trees is Krustal's algorithm.

Let G again be a connected weighted graph. At each step choose an edge of smallest weight that doesn't make a simple

circuit with edges already chosen. If G has n vertices then stop after you have chosen n-1 edges.

Kruskal's algorithm is similar to Prim's and also greedy. It is simpler in that the edges you choose don't have to be connected to those already chosen.

Example (2) again using Kruskal.
Choose edge 1 first. Next choose an edge of weight 2 and after that the other edge of weight 2. Keep going at you get the same minimum spanning tree of weight 40.

These two algorithms give very efficient ways to compute minimum spanning trees. Prim's is slightly more efficient if the graph G has many edges, Kruskal's more efficient if there are fewer edges.

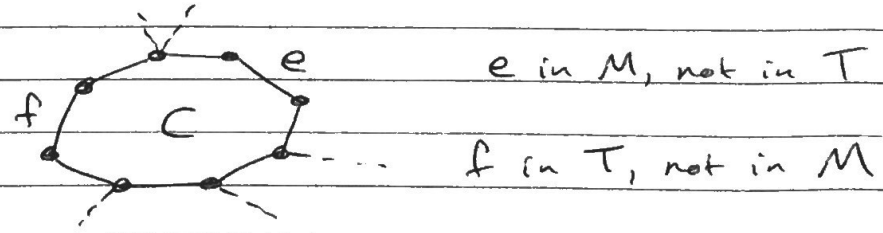
(can skip)

Proof that Kruskal's algorithm works

Suppose the algorithm is applied to a connected graph G with n vertices to get the graph T.

Then T is a graph with n-1 edges and at most n vertices and no simple circuits. So T must be a collection of trees. Use the fact that a tree with k edges has k+1 vertices to prove that T is a tree containing n vertices. So T is a spanning tree for G.

We want to show T is a minimal spanning tree. Let M be a minimal spanning tree. If T = M then we are finished. Otherwise M has $k \geq 1$ edges that are not in T. Let e be one of these edges. If we add edge e to T there will be one simple circuit C. Since C is not in M there must be an edge f of C that is not in M:



Note that the weight of edge f must be ≤ weight of e (or else the algorithm would have added e instead of f):

$$wgt(f) \leq wgt(e)$$

Let T_1 be T with f removed and e added.

Then T_1 is a spanning tree for G and

$$wgt(T) \leq wgt(T_1).$$

Also M has $k-1$ edges that are not in T_1 (one fewer than T). Repeat this process k times until $M = T_k$ and we have

$$wgt(T) \leq wgt(T_1) \leq wgt(T_2) \leq \cdots \leq wgt(T_k) = wgt(M).$$

Since M has smallest possible weight it follows that $wgt(T) = wgt(M)$ and so T is a minimal spanning tree.

□