CS140: Algorithms

Fall 2000

Solutions to Assignment 11

1. The Maximum Weight Spanning Tree problem, as its name implies, asks for a maximum weight spanning tree of the (connected) input graph. Show how to reduce the Maximum Weight Spanning Tree problem to MST.

   Simply negate the weights of the edges of the input graph and find an MST. The associated edges in the original graph form a maximum weight spanning tree.

2. A Bottleneck Spanning Tree (BST) of a connected graph G is a spanning tree whose maximum weight edge is a small as possible. In other words, the cost of a spanning tree for this problem is the weight of its maximum weight edge. The problem is to find a spanning tree with minimum cost. (Assume G has n vertices and m edges.)

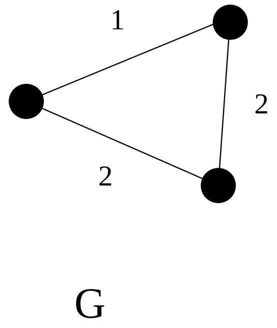
   (a) Suppose T is an MST G. Show that T is also a BST of G.
   Suppose T is an MST but not a BST of G. Let T' be a BST of G. Since T is not a BST of G there is some edge e of T such that $w(e) > w(e')$ for every edge e' of T'. $T - \{e\}$ is a forest consisting of two non-empty trees. Let us color the vertices in one of the trees red and in the other tree blue. Since T' is connected it must have some edge e' that has one red endpoint and one blue endpoint. By our earlier remark $w(e') < w(e)$. Thus $T - \{e\} + \{e'\}$ is a spanning tree of G with weight less than $w(T)$. But this contradicts the fact that T is an MST of G. Hence T must be a BST of G.

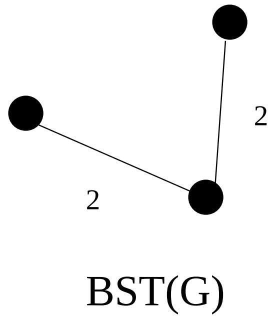
   (b) Give a weighted, connected graph G and a BST, T, of G such that T is not an MST of G.
   See figure.

   (c) By part (a) we can clearly reduce BST to MST. But by part (b) we might guess that there is a more efficient algorithm for finding a BST of G.

      i. Suppose I tell you that the cost of a BST of G is W. Give an $O(m)$ algorithm that then finds a BST of G.
      Remove all edges of G with weight greater than W in $O(m)$ time. Use DFS to find a spanning tree of the resulting graph in $O(m)$ time.

      ii. Suppose I simply want to know if the weight of a BST of G is less than or equal to W. Give an $O(m)$ algorithm that answers this question.





Remove all edges of G with weight greater than W in $O(m)$ time. Check if the resulting graph is connected using DFS in $O(m)$ time.

iii. Now design an $O(m)$ algorithm that finds a BST of G. Hint: Use divide and conquer.

Our algorithm first finds a median weight edge e in G. Then we construct G' by removing all edges of G that have weight at least $w(e)$. If G' is connected then any BST of G' is also a BST of G so we will recursively compute BST(G').

If G' is not connected then the weight of the BST of G is W or greater. In this case, let $G_1, G_2, \ldots, G_k$ be the connected components of G'. To build a BST of G we first find spanning trees of each component G_i. Next we have to find "good" edges of G to connect these spanning trees.

To do so we construct another graph G'' with k vertices $w_1, \ldots, w_k$. For any i and j such that there is an edge between a vertex in G_i and a vertex in G_j in G, G'' has an edge (w_i, w_j) with weight equal to the minimum weight of any such edge. Now notice that the weight of the BST of G is exactly the weight of the BST of G''. Next call BST(G''). To complete the construction of the BST of G we do the following. For each edge $e = (w_i, w_j)$ of the BST of G'', add the corresponding edge connecting vertices in G_i and G_j to the BST of G.

Notice that do at most $O(m)$ work and then make a recursive call on a connected graph with no more than $m/2$ edges. Therefore our algorithm has running time $T(m) = T(m/2) + O(m) = O(m)$.