CS140: Algorithms

Fall 2001

Solutions to Assignment 8

1. Give an algorithm for reducing the key of a record in a heap that runs in time $O(h)$, where h is the height of the heap.

   Compare the revised key with the key at the parent. If the heap property is violated swap and repeat.

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.
   Let T' be a BST of G and let W be the weight of T'. Suppose T is not a BST of G. Then T must contain an edge e with weight greater than W. If we remove e from T we get a forest of two trees. Consider the cut of G defined by these trees. T' must have an edge e' that crosses this cut and $w(e') < w(e)$. Thus $T - e + e'$ is a spanning tree of G with weight less than T, which is a contradiction.

   (b) Give a weighted, connected graph G and a BST, T, of G such that T is not an MST of G.
   Left to the student.

   (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. The resulting graph must be connected and any spanning tree will be a BST of G. We can find a spanning tree in $O(m)$ time with a depth-first search.

      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. The resulting graph is connected if and only if the weight of a BST of G is less than or equal to W. We can determine if the graph is connected in $O(m)$ time with a depth-first search.

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

Find the median weight, W, of the edges of G. Remove all edges with weight greater than W to create G'. (If more than one edge of G has weight W, also remove half of the edges with weight W.) Test if G' is connected. If so, then any BST of G' is also a BST of G. Recursively find a BST of G'. Note: G' has $m/2$ edges.

If G' is not connected, let $G_1, G_2, \ldots, G_k$ be the connected components of G'. Construct a graph G'' as follows. G'' has a node v_i for each connected component of G_i of G'. If G contains an edge e between a vertex of G_i and a vertex of G_j, let w be the minimum weight of any such edge. Add an edge between v_i and v_j in G'' with weight w. Note: G'' has at most $m/2$ edges. Recursively find a BST T'' of G''. Return the union of T'' and spanning trees of each G_i.

The running time of this algorithm is $T(m) = T(m/2) + cm = O(m)$.