Algorithms B Exam, May 1996

Do any 3 out of the following 4 questions:

1. Recall that a simple path is a path with no repeated vertices. Given a graph G with positive and negative edge weights and two designated vertices s and t, the *negative s-t path problem* is whether there exists some simple path between s to t whose weight is negative (the weight of a path is the sum of the weights of the edges on the path). Recall that the Hamiltonian cycle problem is whether, given an arbitrary graph, there exists a cycle which includes all the vertices. You have to show that the negative s-t path problem is NP-hard by finding a polynomial-time reduction from Hamiltonian cycle to it i.e. given an arbitrary unweighted graph G construct an edge-weighted graph G' (with two designated vertices s and t) such that G has a Hamiltonian cycle if and only if G' has a simple path from s to t with a negative weight.

2. Let L be an array of n distinct integers. Give an efficient algorithm to find the length of a longest increasing subsequence of entries in L. For example, if the entries are 11, 17, 5, 8, 6, 4, 7, 12, 3, a longest increasing subsequence is 5, 6, 7, 12. How much time does your algorithm take ?

3. Consider the following sorting algorithm:

   Procedure NewSort(i,j) { sort the elements from A[i] to A[j] }

   if $A[i] > A[j]$ then swap$(A[i], A[j])$
   if $i + 1 \geq j$ then return
   $k = \lfloor (j - i + 1)/3 \rfloor$ { Round down }
   NewSort(i,j-k) { First two-thirds }
   NewSort(i+k,j) { Last two-thirds }
   NewSort(i,j-k) { First two-thirds again }

   (a) Argue that NewSort(1,n) correctly sorts the input array $A[1 \ldots n]$.

   (b) Give a recurrence for the worst-case running time of NewSort and solve the recurrence to get an asymptotic upper bound on the worst-case running time.

4. (a) Consider the following claim about binary search trees. Suppose that the search for key k in a binary search tree ends up in a leaf. Consider three sets: A, the keys to the left of the search path; B, the keys on the search path; and C, the keys to the right of the search path. The claim is that any three keys $a \in A, b \in B, c \in C$ must satisfy $a \leq b \leq c$. Show that the claim is false by giving the smallest possible counterexample i.e. give a counterexample, and show that the claim is indeed true for any tree with smaller number of nodes.

   (b) Find a heap with no repeated elements such that deleting the maximum element and then re-inserting it leads to a different heap.