CS 466/666, Fall 2006 due Friday Sept. 29

A. Lubiw

ASSIGNMENT 1

Do not copy from others, and acknowledge your sources.

1. This question is an NP-completeness review. You may assume that the Hamiltonian Cycle Problem and the Travelling Salesman Problem are NP-complete.
   (i) Prove that the following problem is NP-complete: Given an edge weighted graph and a number K, is there a cycle that visits every vertex at least once (possibly more often) and has sum of edge weights at most K. NOTE: this differs from the standard TSP in that the cycle may re-visit some vertices.
   (ii) Prove that the decision version of TSP is *strongly NP-complete* meaning that it is still NP-complete if the input numbers are encoded in unary rather than binary.

2. Show that the amortized cost of merging binomial heaps is actually $O(1)$. As the potential of a binomial heap, use the number of trees plus the rank of the largest tree. Analyze the amortized cost of the other operations (insert, delete-min) as well.

3. When we want to store a set and perform searches, but use only equality tests on the elements (not comparisons), a list is an appropriate data structure. A *self-adjusting* list is one that reorders itself after each access. One good way to do this is the "move-to-front" heuristic: after accessing an item, move it to the front of the list. Give an amortized analysis to show that for any sequence of accesses the move-to-front heuristic uses at most twice the number of probes into the list as the optimum static ordering of the list. Note that for the optimum static ordering you count the number of times each element is accessed, and put more frequently accessed elements at the front of the list. You may assume that move-to-front starts with this optimum static ordering. Hint: Use as your potential function the number of inversions in the move-to-front list compared to the optimum static ordering. An inversion is a pair i, j where i appears before j in one list, but after j in the other.

4. This question explores alternate potential functions for analyzing splay trees. As the result of a splay, most of the nodes on the access path are moved halfway towards the root, while a couple of nodes on the path move down one level. This suggests using the sum over all nodes of the logarithm of each node's depth as a potential function.
   (i) What is the maximum value of the potential function for a tree of n nodes?
   (ii) What is the minimum value of the potential function?
   (iii) The difference between the answers to (i) and (ii) gives some indication that this potential function isn't too good. Show that a splay operation could increase the potential by $\Theta(n/\log n)$.