Fundamental Algorithms Sample Final Exam Summer 2011

In any of the problems below, you may need not explain any of the standard algorithms or data structures discussed in class. For example, if you wish to use a 2-3 tree for some problem, you may simply say "Use a 2-3 tree with the standard ADD and DELETE operations." You do not have to describe the details of the data structure or operation unless the problem specifically asks for it.

Problem 1: (5 points)
If you are sorting a million items, roughly how much faster is heapsort than insertion sort? (Note: log(1,0000,000) = 20.)

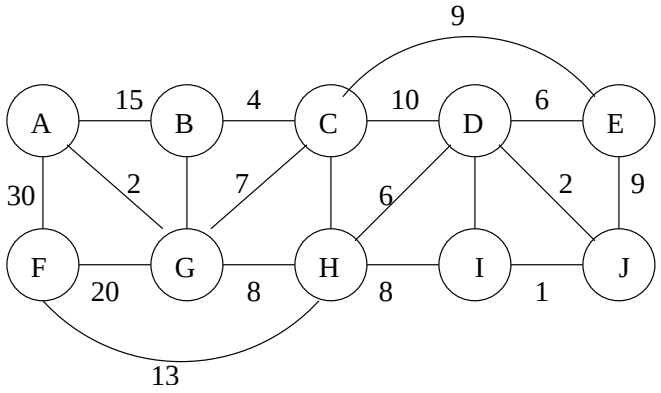
Problem 2: (10 points)
Given a list of integers, you wish to find the *mode*; that is, the value that appears most often in the list. Let n be the length of the list, and let k be the number of different values in the list. For instance, in the list $\langle 1, 5, 2, 5, 2, 5, 5, 1\rangle$, $n = 9$, $k = 3$, and the mode is 5, which appears four times.

Assume that the values are all between 1 and M, and that you have enough memory to construct an array of size M. Given an algorithm to find the mode in time $O(n)$.

Problem 3: (20 points)

A. Give a trace of Dijkstra's single-source shortest path algorithm running on the graph below from source vertex A. You should show the successive states of the array that holds the current estimate of the shortest path. Do not worry about path recovery.

B. Show the sequence in which Kruskal's algorithm adds edges to the minimum spanning tree in the graph below. You need not show the union-find sets.



Problem 4: (10 points)
Show how the disjoint tree implementation of Union-Find sets with merge by rank and path-compression, can be modified to support the following three operations with specified running times. Be sure to explain how the UNION operation is affected by the need to maintain this additional information. The UNION operation should still run in time $\Theta(1)$. It should not be necessary to modify the FIND operation.

MIN(r) — Find the smallest element in the set with root r. Time = $\Theta(1)$. MAX(r) — Find the largest element in the set with root r. Time = $\Theta(1)$.
ENUMERATE(r) — List the elements in the set with root r. Time = $\Theta(\mid S \mid)$.

Problem 5: (20 points) Describe an implementation of an ADT for sets of ordered elements that supports all of the following operations in worst-case time $\Theta(n \log n)$. (The same data structure should support all three operations simultaneously.)

ADD(x,S) — Add element x to set S.
DELETE(x,S) — Delete element x from set S.
SUM-BETWEEN(x,y,S) — Find the sum of the values in S that lie between x and y inclusive. For example, if S currently has the value { 2, 3, 5, 7, 11, 13, 17, 19 }, then SUM-BETWEEN(5,14,S) should return 36 (= 5 + 7 + 11 + 13).

For this problem, you should use a modification of a standard ADT. In your answer, it suffices to sketch the changes that you would make to the data structure and the operations; you do not have to give a complete account of the algorithm.

Problem 6: (20 points)
Construct an algorithm with the following properties:

A. The input is an integer k and an array A of integers of arbitrary size. The length of A is at least $2k$. When the algorithm is complete, the k smallest values in A are in order in the first k places of A and the k largest values in A are in order in the last k places of A. For instance, if $k = 2$ and A on input is the array [69, 2, 300, 10, 91, 4, 106, 289], then an acceptable output would be [2, 4, 91, 10, 69, 289, 300].

B. The algorithm runs in place; that is, it requires no additional storage beyond A.

C. The worst case running time is $\Theta(N + K \log N)$ where N is the length of A.

(Note: If you cannot find an algorithm with the desired running time, some partial credit will be given for an algorithm with longer running time. In that case, you should specify the running time of your algorithm.)

Problem 7: (15 points)
Modify the Floyd-Warshall algorithm so that it additionally creates an array $M[I, J]$ which is the first vertex after I on the shortest path from I to J.