Problem Set 3

Assigned: June 11
Due: June 18

Problem 1

Suppose that you are given the problem of returning in sorted order the k smallest elements in an array of size n, where k is much smaller than n, but much larger than 1.

a. Describe how each of the following algorithms can be modified to solve this problem: selection sort, insertion sort, heapsort, mergesort (you may use the simple recursive version), quicksort. Your description need not give the pseudo-code for the modified algoriths; it is enough simply to describe what changes should be made, as long as your description is clear.

b. Give the worst case running time as a function of k and n for all your modified algorithms except quicksort.

Problem 2

Consider the implementation of a heap as a dynamic binary tree (rather than an array implementation) where each node is an object with a pointer to the parent and the two children.

It will not suffice to have just pointers to parent and children nodes, and a global pointer to the root. Why not? Describe how the standard tree implementation can be extended to support the heap operations `add` and `deleteMin`, and describe briefly how these to operations can be implemented in this data structure.

Problem 3.

(CLR&S 7.5-6)

Give an $O(n \cdot \lg(k))$ time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. (Hint: Use a heap for k-way merging.)

Problem 4.

(Modified from Siegel 5.24.) Design an efficient algorithm to determine whether two unsorted sets of m and n integers are disjoint. Assume that $m < n$. Full credit will be given for an algorithm that runs in time $O(n \log m)$; half credit will be given for an algorithm that runs in time $O(n \log n + m \log m)$ (which is the same thing as $O(n \log n)$.