Problem Set 3

Assigned: June 12
Due: June 19

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 descrive 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.

Show that any comparison method for solving the problem in problem 1 must take at least $\Omega(k \cdot \lg(n))$ in the worst case

Problem 3

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 4

One can modify the definition of a heap so that each node has k children, for any $k \geq 2$. Assume that $k \ll n$.

a. What is the height of a heap with n elements, as a function of n and k?

b. What are the running times of the functions `add(x)` and `deleteMin()` as functions of n and k? What would be the running time of a heapsort, implemented with this kind of heap?