Problem Set 2

Assigned: June 5
Due: June 12

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 $\log(n)$, but much larger than 1.

a. Describe how selection sort, insertion sort, mergesort, and heapsort can be adapted to this problem. Your description need not give the pseudo-code for the modified algorithms; it is enough simply to describe what changes can be made, as long as your description is clear. You may use the recursive version of mergesort.

b. Find the worst-case running times of these algorithms as functions of $k$ and $n$.

Problem 2.

Give an $O(n \cdot \log(k))$ 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 3.

(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 $O(n \log n)$).

Problem 4

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.