Problem Set 2

Assigned: June 1
Due: June 8

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 selection sort, mergesort, heapsort, and quicksort can be adapted to this problem. (There is nothing to be done with insertion sort.) Your description need not give the pseudocode 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 the modified selection sort as a function of k and n.

c. Find the worst-case running time of the modified mergesort.

Problem 2.

Let us say that an array A of length n is *almost sorted with errors of size* k for $k < n$ if, for any I, J, if $J - I > k$ then A[J] $\geq$ A[I]. Thus, the array does not have to be completely ordered, but any two elements in the array that are out of order cannot be more than k places apart. For example, the list

50, 80, 10, 60, 150, 120, 110, 200, 190, 250, 300, 350, 320

is almost sorted with errors of size 2. A[3] = 10 is less than A[1] = 50, and $3 - 1 = 2$, but there are no elements out of order that are 3 or more steps apart.

a. Show how quicksort can be modified to produce a list that is almost sorted with errors of size k. What is the *best case* running time of this modified quicksort?

b. If the input array is almost sorted with errors of size k, what is the running time of insertion sort? Justify your answer.