Problem Set 8

Assigned: November 15
Due: November 22

Problem 1

Show a trace of mergesort operating on the following array (length=18):

\[26, 6, 29, 54, 71, 25, 39, 8, 2, 14, 36, 43, 22, 50, 4, 18, 62, 34\]

Use the iterative version of mergesort, and produce a trace in the style of Example 2, Lecture Notes 16. Do not give an internal trace of “merge”. Assume that smallSize=3.

Problem 2

Show a trace of quicksort operating on the same array as in problem 1.

Use the in-place version of quicksort, and produce a trace in the style of Example 2, Lecture Notes 17. Do not give an internal trace of partition. Assume that smallSize=3.

Problem 3

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, heapsort, and quicksort 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 the modified selection sort as a function of \( k \) and \( n \).

Problem 4

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.
Honors Problem

Solve problem 3.a above for insertion sort and mergesort as well. (Hint: for both, at each stage, throw away anything that you have determined is not one of the $k$ smallest elements.)