Sorting (2 points)

1. [2 points] We can express insertion sort as a recursive procedure as follows. In order to sort $A[1 \ldots n]$, we recursively sort $A[1 \ldots n-1]$ and then insert $A[n]$ into the sorted array $A[1 \ldots n-1]$. Write a recurrence for the running time of this recursive version of insertion sort.

Mergesort / Quicksort (8 points)

2. Although merge sort runs in $\Theta(n \lg n)$ worst-case time and insertion sort runs in $\Theta\left(n^2\right)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to coarsen the leaves of the recursion by using insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

   (a) [2 points] Show that insertion sort can sort the n/k sublists, each of length k, in $\Theta(nk)$ worst-case time.

   (b) [2 points] Show how to merge the sublists in $\Theta(n \lg(n/k))$ worst-case time.

   (c) [2 points] Given that the modified algorithm runs in $\Theta(nk + n \lg(n/k))$ worst-case time, what is the largest value of k as a function of n for which the modified algorithm has the same running time as standard merge sort, in terms of Θ-notation?

(d) [1 point] How should we choose k in practice?

3. [1 point] What is the running time of quicksort when all elements of array A have the same value?