ICS621 Homework 1: Merge sort & Inversions

Problem 2-1 from CLRS. Although merge sort runs in $\Theta(n \log n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ 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) Show that insertion sort can sort the n/k sublists, each of length k, in $\Theta(nk)$ worst-case time.

b) Show how to merge the sublists in $\Theta(n \log(n/k))$ worst-case time.

c) Given that the modified algorithm runs in $\Theta(nk + n \log(n/k))$ 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) How should we choose k in practice?

Problem 2-4 from CLRS: Inversions. Let $A[1 \dots n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an inversion of A.

a) List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

b) What array with elements from the set $\{1, 2, \dots n\}$ has the most inversions? How many does it have?

c) What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

d) Give an algorithm that determines the number of inversions in any permutation on n elements in $O(nlgn)$ worst-case time. (Hint: Modify merge sort.)

You only need to write the answer for part (d). Remember you need to prove both correctness and efficiency.