Name:_____

Midterm 1 Redemption Homework

This homework is optional, and provides an avenue to recover some points from your midterm exam. Completing this homework cannot in any way harm your grade. Due by 11am on Tuesday, October 24. Accepted with no penalty until 5pm on Tuesday Oct. 24; after which it will not be accepted.

1. Short Answer

   (a) (Circle the correct answer) The optimal comparison based sorting algorithm has worst-case running time O(n log n) when operating with which data structure(s): ordered linked list, ordered array, or both?

   (b) Define what it means for a sorting algorithm to be in-place.

   (c) Suppose quicksort always uses the first element as a pivot. Give an example array of 7 elements where quicksort would perform as *few* comparisons as possible.

   (d) Give an example problem domain (for example: the input is a list of 700 prime numbers) where each of the following sorting algorithms are optimal:

   Counting Sort

   Merge Sort

   (e) Consider the following four functions. List them in order of increasing asymptotic complexity (e.g. if you think that f_1 and f_2 have the same asymptotic complexity, then f_4 is larger and f_3 is larger than that, then your answer should look something like: $(f_1, f_2), f_4, f_3$):

   $f_1(n) = log(n^2)$
   $f_2(n) = \sqrt{n}$
   $f_3(n) = (logn)^2$
   $f_4(n) = \sum_{i=1}^{n} \frac{1}{i}$

2. I define an "almost sorted" array A as one where all but a few elements are in order. For example, the array:

   [1 3 4 5 2 6 7 9 0 9]

   is completely in order if you take out the 0 and the 2.

   Suppose you are given an array A that is "almost sorted" like the above array, so that there are 2 elements you can take out and the array is in order (but you don't ahead of time know which they are). Give an O(n) time algorithm that returns the array in the correct order, argue that your algorithm is corrent and analyze its running time.

3. Suppose you are different k sorted lists, each with n elements. Design an $O(nklogk)$ algorithm that merges all these sorted lists into one long, correctly sorted list. Describe your algorith, argue for its correctness and analyze its run time. (Note that your final list has nk total elements).