CSC282 Fall 2005 Homework #4

1. You have an array of 1000 records in which only a few are out of order and they are not very far from their correct positions. Which sorting algorithm would you use to order the whole array? Justify your choice.

2. If `E` is an array of n integers (which can be positive, negative, or zero), the *maximum subsequence sum (MSS)* is the maximum sum for a contiguous (sub)sequence of array elements. If all are negative, define the MSS to be the null sequence of sum zero.

A. Give an algorithm that finds the MSS in `E`. What is the asymptotic order of your algorithm's running time? (There are lots of methods with complexities like: $33n, 46n \lg n, 13n^2, 3.4n^3, ...$).

B. There is a $\Theta(n)$ algorithm for this problem. Find it!

3. Suppose you have an unsorted array `A` of n elements and you want to know if it contains any duplicate elements.

A. Outline (clearly) an efficient solution.

B. Give and justify its asymptotic running time.

C. Suppose you know the elements are integers taken from the range $1, 2, \ldots, 2n$, so other operations besides key comparisons can be done (sorting elements into buckets would be such an operation, for example). Give an algorithm specialized to use this information, and its worst-case asymptotic running time (which should be an improvement over the answer in part B, or what's the point?).

4. Suppose `E1` and `E2` are arrays, each with n keys sorted in ascending order.

A. Devise an $O(\log n)$ algorithm to find the nth smallest of the $2n$ keys (the median of the combined set). Assume the keys are all distinct.

B. Give a lower bound for this problem.

5. CLRS 11-3.

6. You come up with this closed hashing collision resolution strategy: As usual for key x , you will try cells $h_0(x), h_1(x), h_2(x), \ldots$ in order where $h_i(x) = (Hash(x) + f(i))$ mod `HTableSize`, $f(0) = 0$. You decide to define your sequence $f(i) = r_i$, where $r_0 = 0$ and $r_1, r_2, r_3, \ldots, r_n$ is a random permutation of the first n integers (each one appearing exactly once of course).

A. Prove your strategy always resolves collisions unless the table is full.

B. Will your strategy eliminate clustering?

C. If the load factor in the table is α, what is the expected time to perform an insertion?

D. If the load factor in the table is α, what is the expected time for a successful search?

E. What is a theoretically and practically efficient way to generate the random permutation sequence?