Algorithms Lab 3

The in-lab problems are to be solved during the lab time. Work with your team, but write your solutions individually. Your answers will not be graded, however you need to work through these problems; please staple the answers to the in-class exercises to the assignment that you hand in.

The homework problem set is due in one week. Work with your team, but write your solutions individually. List the people with whom you discussed the problems.

1 In lab exercises

1. Finish the class exercises on heaps.
2. (C-4.9) Suppose we are given a sequence S of n elements, each of which is colored red or blue. Assuming S is represented as an array, give an in-place method for ordering S so that all blue elements are listed before all the red elements.
3. Read the HOARE PARTITION algorithm from the textbook (problem 7-1, page 185), and understand how it works.
4. (GT C-4.16) Suppose we are given a sequence S of n elements, on which a total order relation is defined (meaning any two elements can be compared). Describe a method for determining whether there are two equal elements in S. What is the running time of your method?

2 Homework problems

1. (R-4.9, R-4.10) Suppose we modify the deterministic version of the quicksort algorithm so that, instead of selecting the last element as the pivot, we chose the element at index $\lfloor n/2 \rfloor$, that is, an element in the middle of the sequence. What is the running time of this version of quicksort on a sequence that is already sorted? What kind of sequence would cause this version of quicksort to run in $\Theta(n^2)$ time?
2. (CLRS 8-2) Suppose we have an array of n data records to sort and that the key of each record has the value 0 or 1. An algorithm for sorting such a set of records might possess some subset of the following three desirable characteristics.:

   (1) The algorithm runs in $O(n)$ time.

(2) The algorithm is stable.

(3) The algorithm sorts in place, using no more than a constant amount of storage space in addition to the original array.

(a) Give an algorithm that satisfies (1) and (2) above.

(b) Give an algorithm that satisfies (1) and (3) above.

(c) Give an algorithm that satisfies (2) and (3) above.

(d) How would you extend your algorithm from (b) to handle the case when the values are $0, 1$ or 2; that is, you want to sort in place in $O(n)$ time.

3. (CLRS 7-3) Professors Dewey, Cheatham, and Howe have proposed the following "elegant" sorting algorithm:

   STOOGE-SORT(A, i, j)
   if $A[i] > A[j]$
       then exchange $A[i] \leftrightarrow A[j]$
   if $i + 1 \geq j$
       then return
   $k \leftarrow \lfloor (j - i + 1)/3 \rfloor$
   STOOGE-SORT$(A, i, j - k)$
   STOOGE-SORT$(A, i + k, j)$
   STOOGE-SORT$(A, i, j - k)$

   a. Argue that STOOGE-SORT$(A, 1, length[A])$ correctly sorts the input array $A[1..n]$, where $n = length[A]$.

   b. Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

   c. Compare the worst-case running time of STOOGE-SORT with that of insertion sort, merge sort, heapsort, and quicksort. Do the professors deserve tenure?

4. (CLRS 6.5-9) Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. (*Hint: use a min-heap for k-way merging.*)