Algorithms Lab 4

The in-lab problems are to be solved during the lab time. Work with your team, but write your solutions individually. You do not need to turn in your answers, only discuss them with me. Ask me for feedback.

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 clearly on the first page.

1 In lab exercises

- (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 an efficient method for determining whether there are two equal elements in S. What is the running time of your method?
- Read the Hoare Partition algorithm from the textbook (problem 7-1, page 185), and understand how it works.

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. (C-4.22) Let A and B be two sequences of n integers each. Given an integer x, describe an $O(n \lg n)$ algorithm for determining if there is an integer a in A and an integer b in B such that $x = a + b$.

(extra credit) (CLRS 8.3-4) Suppose we are given a sequence S of n elements, each of which is an integer in the range $[0, n^3 - 1]$. Describe a simple method for sorting S in $O(n)$ time.