Data Structures and Algorithms

COSC 336 Assignment 8

Instructions.

1. Due November 11.
2. Your programs must be written in Java.
3. Write your programs neatly - imagine yourself grading your program and see if it is easy to read and understand. At the very beginning present your algorithm in plain English or in pseudo-code (or both). Comment your programs reasonably: there is no need to comment lines like "i++" but do include brief comments describing the main purpose of a specific block of lines.
4. You will submit on Blackboard 2 files. The first file should be a .pdf file with the solution to the exercises and with descriptions in English or in pseudocode of the algorithm for the programming task you are required to do and the results that you are required to report. Make sure you label the results as indicated below. Also insert images/screenshots with the output you obtain for each testing data. The second file will contain the Java source of your program.

   For editing the above document with Latex, see the template posted on the course website.

   assignment-template.tex and

   assignment-template.pdf

   To append in the latex file a pdf file, place it in the same folder and then include them in the latex file with

   ```
   \includepdf[pages=-,pagecommand={},width=\textwidth]{file.pdf}
   ```

   To append in the latex file a .jpg file (for a photo), use

   ```
   \includegraphics[width=\linewidth]{file.jpg}
   ```

Exercise 1. Show similarly to Fig 8.3 on page 198 in the textbook, how RadixSort sorts the following arrays:

1. 34, 9134, 20134, 29134, 4, 134
2. 4, 34, 134, 9134, 20134, 29134
3. 29134, 20134, 9134, 134, 34, 4

Exercise 2. Present an $O(n)$ algorithm that sorts n positive integer numbers $a_1, a_2, \ldots, a_n$ which are known to be bounded by $n^2 - 1$ (so $0 \leq a_i \leq n^2 - 1$, for every $i = 1, \ldots, n$. Use the idea of Radix Sort (discussed in class and presented in Section 8.3 in the textbook).

Note that in order to obtain $O(n)$ you have to adapt the Radix Sort algorithm. The idea is to represent each number in a base chosen so that each number requires only 2 "digits." Explain what are the "digits, " and how the "digits" of each number are calculated. Note that you cannot use the base 10 representation, because $n^2 - 1$ (which is the largest possible value) requires $\log_{10}(n^2 - 1)$ digits in base 10, which is obviously not constant and therefore you would not obtain an $O(n)$-time algorithm.

Illustrate your algorithm by showing on paper similar to Fig. 8.3, page 198 in the textbook (make sure you indicate clearly the columns) how the algorithm sorts the following sequence of 12 positive integers:

45, 98, 3, 82, 132, 71, 72, 143, 91, 28, 7, 45.

In this example $n = 12$, because there are 12 positive numbers in the sequence bounded by $143 = 12^2 - 1$.

Programming Task.

We are given a sequence of n numbers $a_1, a_2, \ldots, a_n$. Your task is to determine two things: (1) whether there exists an integer x that occurs in the sequence more than $n/2$ times, and (2) whether there exists an integer x that occurs in the sequence more than $n/3$ times. See the example in the file input-8.1.txt and the answers in the file answer-8.1.txt. Design an algorithm that runs in time $O(n)$ using the Randomized Selection algorithm - see Section 9.2 in the textbook.

Hint: Using Randomized Selection, you can find a single "suspect" for the number that may appear more than $n/2$ times. To find the 'suspect', think what would be the median of an array containing some value more than $n/2$ times. Once you have a suspect, you can verify if it actually appears more than $n/2$ times or not by doing one pass through the area to count the number of occurrences.

For the second part with more than $n/3$ appearances, using Randomized Select, you can find two possible "suspects", and then you can check each one of them to see if it has the required number of occurrences.

(Note: since the algorithm is randomized, the $O(n)$ bounds the *expected* running time).

Input specification: The input contains two lines. The first line contains n and the second line contains the integers $a_1, a_2, \ldots, a_n$, separated by spaces. You may assume that all numbers fit within int and that n is not bigger than 10,000.

Output specification: The output contains two lines containing as answers to the two questions the strings "YES" or "NO" (see the sample outputs below) .

Sample input :

input-8.1.txt

Sample output:

answer-8.1.txt

Test your program on the following input files:

input-8.2.txt input-8.3.txt

and report the results you have obtained for these two inputs.