CMPSC 465 Data Structures & Algorithms
Fall 2021 Antonio Blanca and David Koslicki

HW 4

Due September 27, 10:00 pm

Instructions: You are encouraged to solve the problem sets on your own, or in groups of three to five people, but you must write your solutions strictly by yourself. You must explicitly acknowledge in your write-up all your collaborators, as well as any books, papers, web pages, etc. you got ideas from.

Formatting: Each part of each problem should begin on a new page. Each page should be clearly labeled with the problem number and the problem part. The pages of your homework submissions must be in order. When submitting in Gradescope, make sure that you assign pages to problems from the rubric. You risk receiving no credit for it if you do not adhere to these guidelines.

Late homework will not be accepted. Please, do not ask for extensions since we will provide solutions shortly after the due date. Remember that we will drop your lowest two scores.

This homework is due Monday, September 27, at 10:00 pm electronically. You need to submit it via Gradescope (Class code 6PERPR). Please ask on Piazza about any details concerning Gradescope.

Extra credit. Five extra credits will be added to your homework score this time if you use Latex to typeset your solutions.

1. (20 pts.) Same Value as Index

Given a sorted zero-indexed array A of n (possibly negative) distinct integers, you want to find out whether there is an index i for which $A[i] = i$. Give an algorithm for this problem with $O(\log n)$ running time.

(a) Describe your algorithm in words and explain why it's correct.

(b) Analyze the running time of your algorithm.

2. (20 pts.) Square vs Multiplication: Matrices

The square of a matrix A is its product with itself, AA.

(a) Show that five multiplications are sufficient to compute the square of a 2×2 matrix.

(b) What is wrong with the statement below on an algorithm for computing the square of an $n \times n$ matrix?
"Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (a). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in $O(n^{\log_2 5})$ time."

(c) In fact, squaring matrices is no easier than multiplying them. Show that if $n \times n$ matrices can be squared in $O(n^c)$ time for a constant c, then any $n \times n$ matrices can be multiplied in $O(n^c)$ time.
(*Hint:* First, show that squaring a $2n \times 2n$ matrix can also be done in $O(n^c)$ time. Then, think about how to obtain the result of multiplying two $n \times n$ matrices from the result of squaring a $2n \times 2n$ matrix.)

3. (20 pts.) Binary search on infinite array

You are given an infinite array $A[\cdot]$ in which the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. You are not given the value of n. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in $O(\log n)$ time. (If you are disturbed by the fact that the array A has infinite length, assume instead that it is of length n, but that you don't know this length, and that the implementation of the array data type in your programming language returns the error message ∞ whenever elements $A[i]$ with $i \geq n$ are accessed.)

4. (20 pts.) Two sorted arrays.

You are given two sorted arrays, each of size n. Give an algorithm with running time $O(\log k)$ to find the k-th smallest element in the union of the two arrays.

5. (20 pts.) Majority Element

An array $A[0,\ldots,n-1]$ is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?".(Think of the array elements as image files, say.) However you can answer questions of the form: "is $A[i] = A[j]$?" in constant $O(1)$ time.

(a) Show that if x is a majority element in the array then x is a majority element in the first half of the array or in the second half of the array.

(b) Show how to check in $O(n)$ time if a candidate element x is indeed a majority element.

(c) Put these observation together to design a divide an conquer algorithm whose running time is $O(n \log n)$. (Hint: Use the master theorem to analyze its running time.)

(d) We will now design a linear-time algorithm. Here's another divide-and-conquer approach:

- Pair up the elements of A arbitrarily, to get $n/2$ pairs. (If n is odd, we keep the singleton unpaired element.)
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them.

Let L be the elements left after this procedure. Show that L has at most $\lceil n/2 \rceil$ elements, and that if A has a majority element, then L has the same majority element.

(e) Use parts (d) and (b) to design an algorithm for this problem with $O(n)$ running time.