CS 344 - Spring 2021
Homework 2.
100 points total plus 15 points extra credit

1 Problem 1 (30 points total)

For all the recursive formula problems below, you can assume that $T(1) = T(2) = 1$.

- **Part 1:** (15 points) Consider an algorithm that solves a problem of size n by dividing it into 4 pieces of size $n/4$, recursively solving each piece, and then combining the solutions in $O(n^2)$ time. What is the recurrence formula for this algorithm? Use a recursion tree to figure out the running time of the algorithm (in big-O notation, as always), and then use a proof by induction to show that the result is correct.

- **Part 2:** (7 points) Say that you have an algorithm with recurrence formula $T(n) = 8T(n/2) + n^3$. Use a recursion tree to figure out the running time of your algorithm. *No need for an induction proof of this one.*

- **Part 3:** (8 points) Say that you have an algorithm with recurrence formula $T(n) = 5T(n/4) + n$. Use a recursion tree to figure out the running time of your algorithm. *No need for an induction proof on this one.*

2 Problem 2 (10 points total)

Let's say you are given a sorted array A of length n (sorted from smallest to largest). Assume that all the elements of A are integers and that they are distinct (no duplicates). *Note that the integers in A can be negative.* Your goal is to find an index i such that $A[i] = i$, or output "no solution" if no such index exists. (If there are multiple i for which $A[i] = i$, you only have to return one of them.)

Give pseudocode for a recursive algorithm for the above problem that runs in $O(\log(n))$ times. You should also justify that the algorithm is correct and state the recurrence formula for your algorithm.

3 Problem 3 (25 points total)

Given an array A of length n, we say that x is the majority element of A if x appears at least $2n/3$ times in A.

Now, Say that your array A consists of incomparable objects, where you can ask whether two objects are equal (i.e. check if $A[i] == A[j]$), but there is NOT a notion of one object being bigger than another. In particular, we cannot sort A or find the median of A.

Given such an array A of incomparable objects, write pseudocode for an agorithm that finds the majority element of A, or returns "no solution" if none exists. Your algorithm should run in $O(n \log(n))$ time.

In addition to pseudocode, you should justify correctness and state the recurrence formula for the algorithm.

NOTE: don't try anything silly like converting the incomparable objects to integers so that you can then sort the integers. I mean it when I say no sorting, no selection!

4 Problem 4 (35 points total)

Consider an array A, possibly with repeat elements. We say that a subarray $A[i...j]$ is strictly increasing if $A[i] < A[i+1] < ... < A[j-1] < A[j]$.

Now consider the following problem:

- Input: an array A of integers. The input array A is NOT sorted and there can be duplicates.
- Output: the length of the longest strictly increasing subarray $A[i...j]$

For example, if A = 2,5,1,6,10,7,9 then the output is 3, since the longest strictly increasing subarray is 1,6,10.

Another example: if A = 4,3,3,1, then the output is 1, since the only increasing subarrays have a single element. e.g. A[2] is an increasing subarray of length 1.

Part 1 (20 points) Show a *recursive* algorithm for the above problem which runs in $O(n \log(n))$ time. You need to write the pseudocode and state what the recurrence formula for your algorithm is.

NOTE: if you end up with a recursive formula that we've seen in class before, then you don't need to use a recursion tree to solve it, you can just state what the solution is. But if you end up with a new recursive formula then you need to prove that $T(n) = O(n \log(n))$.

IMPORTANT NOTE: there exists a pretty simple fast non-recursive algorithms for this problem. *But on this HW you must use a recursive algorithm. You will get zero points if you use a non-recursvie algorithm.*

Part 2 (15 points) Give pseudocode for a recursive algorithm for the above problem that runs in $O(n)$ time. As in part 1, you need to write the pseudocode and state what the recurrence formula for your algorithm is.

NOTE: As in Part 1, if you end up with a recursive formula that we've seen in class before, then you don't need to use a recursion tree to solve it, you can just state what the solution is. But if you end up with a new recursive formula then you need to prove that $T(n) = O(n)$.

NOTE: once again, your algorithm must be recursive. You will get zero points if you use a non-recursive algorithm.

HINT: as we showed for the Max Profit problem in class, you will want to solve a problem LongestIncreasingSubarrayX(A) which outputs more information than just the length of the longest sequence.

Grading Note: If you feel confident in your answer, you can use your solution for part 2 in part 1 as well, since an $O(n)$ algorithm is also an $O(n \log(n))$ algorithm. But note that Part 2 is harder, so if you don't fully confident in your approach, I recommend first getting a good solution to part 1 and then writing up a separate solution for part 2.

5 Problem 5 – Extra Credit (15 points)

Give an $O(n)$ algorithm for problem 3. *Your algorithm does not need to be recursive.* (The simplest solution I can think of of is indeed not recursive, though there exists a recursive solution as well.)

Make sure to justify both correctness and running time.

IMPORTANT NOTE: You may not use hash tables in your solution.