COM S 311 SPRING 2022
HOMEWORK 2

Due: February 24, 11:59 p.m.

Late Submission Due: February 25, 11:59 a.m. (25% penalty)

(1) Solving the following recurrence equations: Assume that $T(n)$ is constant for sufficiently small n.
   (a) $T(n) = 5T(n/4) + \log n$
   (b) $T(n) = 16T(n/2) + n^4$.
   (c) $T(n) = 9T(n/3) + n^2 \log n$

In the next problems, assume that array indices start at 1.

(2) The algorithm below can be proved to correctly (although inefficiently) sort and array A. *You do not need to prove that this algorithm is correct.*

```
1  SillySort(A, i, j)
       Input: An array A of n integers, and two indices, i and j, such that 1 ≤ i ≤ j ≤ n.
       Result: Subarray A[i..j] is sorted in nondecreasing order .
2      n = j − i + 1                          // The size of the subarray we are sorting
3      if n == 2 then
4          if A[i] > A[j] then
5              Swap A[i] and A[j]
6      else if n > 2 then
7          m = ⌊n/3⌋
8          SillySort(A, i, j − m)                              // Sort the first part
9          SillySort(A, i + m, j)                               // Sort the last part
10         SillySort(A, i, j − m)                          // Sort the first part again
```

   (a) Write a recurrence equation for the running time $T(n)$ of `SillySort`.
   (b) What is the solution to the recurrence of part (a)?
   (c) Write the recurrence equation for the running time $T(n)$ of `SillySort` assuming that we replace Line 7 with $m = \lfloor n/4 \rfloor$.
   (d) What is the solution to the recurrence of part (c)?

(3) Design a divide-and-conquer algorithm for the following problem. The input is an array A consisting of n distinct integers, which can be positive, negative, or zero. Assume that A is sorted from low to high. The algorithm should determine if there is an index i such that $A[i] = i$. Make your algorithm as fast as you can possibly make it. For full credit, you must
   - give pseudocode for your algorithm,
   - briefly justify the correctness of your algorithm, and
   - analyze the running time of your algorithm.

(4) Suppose you have an n-element array A of intervals (x_i, y_i), where x_i, y_i are integers such that $x_i \le y_i$. The interval (x_i, y_i) represents the set of integers between x_i and y_i. For example, the interval $(3, 6)$ represents the set $\{3, 4, 5, 6\}$. Define the ***overlap*** of two intervals I, I' to be the number of integers that are members of both intervals. For example, the overlap of intervals $I = (3, 6)$ and $I' = (5, 8)$ is 2, since there are only two integers, namely 5 and 6, that are in both I and I'.

The goal of this problem is to design a divide-and-conquer algorithm that takes as input an n-element array A of intervals and returns the highest overlap among all pairs of intervals in A. Assume that A is sorted by the x_i-values of its intervals.

(a) A naïve approach is to compute the overlap between every pair of intervals. What is the exact number of comparisons that this approach needs? What is the running time of this approach?

(b) Consider the following potential divide-and-conquer algorithm for the problem. First, divide A into two parts L and R of length $n/2$, such that all the x_i-values of intervals in L are smaller than the x_i-values of intervals in R. Next, recursively determine the maximum overlap p_L in L and the maximum overlap p_R in R. Finally, return $\max\{p_L, p_R\}$. Why is this not enough to give the correct answer?

(c) Taking advantage of the fact that the array A is sorted, design a divide-and-conquer algorithm that is faster than the naïve approach. For full credit, you must
- give pseudocode for your algorithm and
- briefly justify the correctness of your algorithm.

(d) What is the recurrence for the running time of your divide-and-conquer algorithm in part (c)?

(e) What is the running time of the algorithm in part (c)?

Guidelines

- For each problem, if you write the statement "I do not know how to solve this problem" (and nothing else), you will receive 20% credit for that problem. If you do write a solution, then your grade could be anywhere between 0% to 100%. To receive this 20% credit, you must explicitly state that you do not know how to solve the problem.
- You may discuss homework with classmates, but you must write the final solutions alone, without consulting anyone. Your writing should demonstrate that you completely understand the solution you present.
- When proofs are required, you should make them clear and rigorous. Do not hand-waive.
- *Homework solutions must be typed.* We can make exceptions for certain diagrams, which can be hand-drawn and scanned. We reserve the right not to grade homework that does not follow the formatting requirements.
- Submit a pdf version of your assignment via Canvas. Please make sure that the file you submit is not corrupted and that its size is reasonable (no more than, say, 10-11 MB).

If we cannot open your file, your homework will not be graded.

- Any concerns about grading should be expressed within one week of returning the homework.

Note. We reserve the right to grade only a subset of the problems assigned. Which problems will be graded will be decided after the submission deadline.