Basic Algorithms CSCI-UA.0310

Midterm Exam

Exam Time: 3:30 - 4:45 PM EST

You have to upload your solutions on Gradescope before 5:00 PM EST.

Instructions

- The exam has 3 Problems and is 75 minutes long. You have an additional 15 minutes to upload your solutions.
- **No submission after 5:00 PM EST will be accepted.** Please submit your solutions well ahead of time so that you will not encounter any connection issues.
- **You must answer each Problem on a separate page.** Submissions must be uploaded to your account on Gradescope by the due time above.
- **You must match your solutions to the corresponding problems while uploading them to Gradescope.**
- It is highly advised that you typeset your solutions via Latex, MS Word, etc. You can also scan your handwritten solutions but it is your responsibility to make sure that your solutions are clearly written and properly scanned. **Your solutions will NOT be considered if not clearly written.**
- You must stay connected to the Zoom meeting during the whole exam with your videos turned on.
- You may use any algorithm or theorem we saw in the class without proof as long as you state it correctly. For all questions asking to give algorithms, you have to fully explain your algorithm and give a clear description of it.
- Grading is based on correctness, clarity, and conciseness.

Requirements

- This exam is closed book. No calculators, phones, smartwatches, etc. are permitted.
- You may use your computer during the exam to only view this file, typeset your solutions, and upload your solutions.
- You may NOT use any communication tools to collaborate with others during the exam. This includes but is not limited to Chat, Messenger, E-mail, etc.
- Anyone found sharing ideas, solutions, or communicating with another student during the exam period will earn an immediate grade of 0.

You have to write the following statement at the beginning of your submission and sign it. Otherwise, your submission will not be considered.

"I understand the ground rules and agree to abide by them. I will not share ideas, solutions, or assist another student during this exam, nor will I seek assistance from another student or attempt to view their ideas or solutions."

Name:

NYU Net ID:

Problem 1 (30 points)

(a) Consider $f(n) = n^5 \ (\log_2 n)^{50}$ and $g(n) = n^{5.1} \ (\log_2 n)$.
Which of the following holds? Justify your answer.
$f(n) = O(g(n))$, or $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$.

(b) Consider $f(n) = 5^n$ and $g(n) = 125^{\sqrt{n}} + n^2$.
Which of the following holds? Justify your answer.
$f(n) = O(g(n))$, or $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$.

(c) Consider the recursion

$$T(n) = 2T\left(\left\lceil \frac{n}{4} \right\rceil\right) + n$$

for $n > 3$, with $T(1) = T(2) = T(3) = 1$.
You may ignore the ceiling function and assume that that n is a power of 4, i.e., $n = 4^k$ for some positive integer k.
Find a Θ expression for the solution of $T(n)$. For this, you have to draw the corresponding recursion tree and compute the running time of the recursion tree.

Problem 2 (35 points)

Let $A[1 \dots n]$ be an array of length n consisting of some positive and some negative integers. Your task is to reorder the elements of A in such a way that all the positive elements come after all the negative elements.

Example: If $A = [10, 5, -8, -4, -10, 12, 1]$, then one possible output is $[-8, -4, -10, 10, 5, 12, 1]$. Another possible output is $[-4, -8, -10, 1, 12, 10, 5]$.
Note that the original ordering of the negative elements may not be necessarily preserved, and the original ordering of the positive elements may not be necessarily preserved.

(a) Design an algorithm which does the required reordering in $\Theta(n)$ time with $\Theta(n)$ additional space.
You only need to fully explain your algorithm in a few sentences. You do NOT need to write down the pseudo-code of your algorithm.

(b) Design an algorithm which does the required reordering in $\Theta(n)$ time with $O(1)$ additional space.
You only need to fully explain your algorithm in a few sentences. You do NOT need to write down the pseudo-code of your algorithm.

(c) Now suppose the elements of A are sorted. Thus, you are given an array $A[1 \dots n]$ of length n consisting of some positive and some negative integers in sorted order. Design a divide and conquer-based algorithm to find the first positive integer in A. Your algorithm must work in $O(\log n)$ time.
You only need to fully explain your algorithm in a few sentences. You do NOT need to write down the pseudo-code of your algorithm.

Example: If $A = [-7, -3, 6, 8, 11, 12]$, then the first positive integer listed in A is $A[3] = 6$, so the algorithm must return 6.

Problem 3 (35 points)

Little George wants to play a little game. There are n boxes in a row in front of him labelled by $1, \dots, n$. Each box has a number of cookies inside which is known beforehand. If Little George chooses a box, he can have all the cookies inside that box.
His parents do not want him to get sick. So they told him that all the boxes he chooses must be more than

k apart. In other words, Little George cannot choose two boxes i and j with $|i - j| \leq k$.
For example, if $n = 15$ and $k = 4$, Little George cannot select the boxes $3, 7, 12$, since the boxes 3 and 7 are 4 apart: $7 - 3 = 4 \leq 4$.
He cannot also select the boxes $2, 3, 10$, since the boxes 2 and 3 are 1 apart: $3 - 2 = 1 \leq 4$.
However, he can select the boxes $1, 6, 12$, since all of them are more than 4 apart.

Your task is to help greedy George to maximize the total number of cookies he can get by choosing a number of boxes given the aforementioned rule. Let $T(n)$ denote the maximum total number of cookies George can get from selecting a number of boxes given that any two of these selected boxes are more than k apart.
You are given an input integer k and an input array $C = [C[1], \ldots, C[n]]$, where $C[i]$ denotes the number of cookies inside box i.

(a) Write the recurrence relation that $T(n)$ satisfies. Fully justify your answer.
Also, identify the base case(s) for your recursion and find their corresponding value(s).

Hint: You may want to first find the recursion for $k = 1$ (this means that no two boxes can be chosen consecutively), and then generalize it.

(b) Write the pseudo-code for the bottom-up dynamic programming algorithm to compute $T(n)$.

(c) Find the time and space complexities of your algorithm in the form of $\Theta(.)$. Fully justify your answers.