COSC3320 Fall 2021 Mid-Term Test

Thur, Oct 21, 2021

Rules and tips:

- The test starts at 4pm and ends at5:20pm. Submit to Blackboard by 5:30pm.
- Close book close notes. No references are allowed.
- Show your face in the camera at all times if you participate online.
- **Read the questions carefully**. Try to get partial credits by writing down clearly partial solutions.
- Good luck! And don't panic.

The test consists of 5 questions (P0, P1, P2, P3, P4) on the next page, each worth 25 points. The total points will be capped at 100pt. **Please pick any 4 questions to answer.**

Please write your name and student ID (PSID) on your answer paper.
Save your paper.

P0 (25pt) Solve the following recurrence in big-Theta.
[Write the result only]

a) $T(n)=T(n/3)+1, T(1)=1$

b) $T(n)=T(n/3)+n, T(1)=1$

c) $T(n)=3T(n/3)+n, T(1)=1$

d) $T(n)=3T(n/2)+n, T(1)=1$

e) $T(n)=4T(n/2)+n^2, T(1)=1$

P1 (25pt) Given an array $A[1\ldots n]$ (not sorted) design an algorithm to find $\max\{A[j]-A[i]\}$ where $1\leqslant i<j\leqslant n$. Strive for a $O(n\log n)$ time complexity algorithm. Slower algorithms (correctly analyzed) can get at most 10pt.

[Hint: try divide-and-conquer. Divide the array into two halves and solve them. Then combine the two solutions.]

P2 (25pt): Let's play a game. You have a $n\times n$ square grid. **Initially you must place a token on a square in the first row**. In each turn, you move your token to one square to the right, or one square down. The game ends when you move your token out of the board. Each square on the grid has a numerical value, which could be positive, zero, or negative. You start with 0 score. Whenever the token lands on a square, you add its value to your score. You try to **score as many points as possible**. **Describe** an efficient algorithm to compute the maximum score, given a board $A[1\ldots n][1\ldots n]$. **Analyze** its time complexity.

For example, given the grid below, and placing initial token on 1st row 2nd column, and moving down, down, right, down, down, the game ends and you have score -9 (which is not the maximum possible).

Backtracking gets you 15pt. Dynamic Programming may get full credit. Greedy algorithm must come with proof of correctness.

−1	7	-8	10	−5
−4	−9	8	−6	0
5	−2	−6	−6	7
−7	4	7	−3	−3
7	1	−6	4	−9

[hint: Try dynamic programming. What's the definition of your table? What's the recurrence + base cases? What's a proper evaluation order? There's no need for pseudo code if you answer this three questions clearly.]

P3 (25pt): Given an array of integers $A[1\ldots n]$ with non-unique elements, printout all **unique permutations** of the array.

For example, the unique permutations of the array $[3,2,3]$ are
[2,3,3], [3,2,3], [3,3,2]
[Describe your algorithm only; no need to prove correctness or analyze complexity].

P4 (25pt) A string w of parentheses (and) and brackets [and] is balanced if it satisfies one of the following conditions:

- w is the empty string
- $w = (x)$ for some balanced string x
- $w = [x]$ for some balanced string x
- $w = xy$ (concatenation) for some balanced strings x and y

For example, the string
w=([()][]())[()()]() is balanced, because $w=xy$, where x=([()][]()) and y=[()()]()

1. Describe and analyze an algorithm to determine whether a given string of parentheses and brackets is balanced.

[No need to analyze time complexity. Backtracking is fine. Dynamic programming is even better, but not required.]