CS515: Algorithms and Data Structures, Winter 2021 – Midterm

Feb 9, 2021

- This exam starts at 1PM. Submissions are accepted until 4:30PM.
- Submit your solutions on canvas. If there is any problem with canvas, email them to `nayyeria@oregonstate.edu` with title "CS515 Midterm, [YOUR NAME]". You can type your answers, or write them and scan (or take a readable picture). Ideally, submit a pdf file in the end, and if not possible submit a zip file.
- This is a closed book exam.
- *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a *completely* wrong answer will receive zero.
- There are 5 problems in this exam. Submit solutions ONLY to four of them. If you submit five solutions, solutions for problems 1 to 4 will be considered for your grade.

Problem 1. There are lines in front of grocery stores in Corvallis during the pandemic time. There are n spaces in a long row, and each space is either empty or taken by a customer. To be extra careful, a new decision has been made that people cannot stand in adjacent spaces, so there must be at least one empty space between every two customers. Let $S(n)$ be all possible states of such a line with n spaces. For example, $S(3) = 5$; the possibilities are $\{EEE, CEE, ECE, EEC, CEC\}$, where E denotes Empty and C denotes Customer.

Write a recurrence for $S(n)$ with its base cases, and show that your recurrence is correct. You can assume that $S(0) = 1$. You do not need to solve the recursion.

Problem 2. It is going to be cold this weekend, and we need to stay warm. I am thinking I should wear as many coats as I can. It is a bit complicated though, as I have two types of constraints: I) each coat has a size, and I can only wear C_i over C_j if C_i is larger than C_j, and II) some pairs of coats have incompatible materials, so they cannot be in direct contact. So, you cannot wear one exactly on top of the other, however, you can wear both of them if there is one or more layers in between.

The input of my problem is going to be $s[1, \ldots, n]$ the sizes of my coats, they are all positive integers. Also, I have a matrix $incomp[1, \ldots, n][1, \ldots, n]$ which specifies for each i, j if the C_i and C_j have incompatible materials: $incomp[i, j] = incomp[j, i] = 1$ if C_i and C_j have incompatible materials. Otherwise, $incomp[i, j] = incomp[j, i] = 0$.

Describe and analyze an algorithm to tell me the maximum number of coats that I can wear this weekend. Argue that your algorithm is correct (it is going to be cold!). Your algorithm should work in polynomial time because the weekend is close.

For example, let C_1, C_2, C_3, C_4 be the set of coats, with $s(C_1) < s(C_2) < s(C_3) < s(C_4)$. Suppose, the following pair of coats have incompatible materials (C_1, C_2), (C_1, C_4), (C_2, C_3). In this case, I can wear three coats C_1, C_3, C_4 in this order.

Problem 3. Let T be a binary rooted tree with positive, negative or zero weights on its edges. Describe and analyze an algorithm to compute the heaviest possible path that starts from the root. Argue that your algorithm is correct. Your algorithm should work in polynomial time. (Note the negative weights; in particular, if all edge weights are negative, the output will be a path of weight zero, just the root vertex).

Problem 4. Suppose we are given a set L of n line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct. Describe and analyze an algorithm to compute the largest subset of L in which **no pair** of segments intersects. Argue that your algorithm is correct. Your algorithm should work in polynomial time. (Note that this problem is different from the one in HW2)

Problem 5. Let $X[1, \ldots, k]$ be a sequence of numbers. We say that X is kind-of-increasing if for each $i > 3$, we have $X[i] > X[i-3]$. Describe and analyze an algorithm to compute the longest kind-of-increasing subsequence of a given sequence $A[1, \ldots, n]$. Argue that your algorithm is correct. Your algorithm should work in polynomial time.