CS 577: Introduction to Algorithms Fall 2020

Homework 3

Instructor: Dieter van Melkebeek TA: Andrew Morgan

This homework covers dynamic programming. **Problems 1 and 3 must be submitted for grading by 11:59pm on 9/28.** Problem 1 will be discussed in your discussion section on Friday. Please refer to the homework guidelines on Canvas for detailed instructions.

Warm-up problems

1. [Graded, 2 points] Suppose you are given a string of letters representing text in some foreign language, but without any spaces or punctuation. You want to break this string into its individual constituent words. For example, you might be given the following passage from Cicero's famous oration in defense of Lucius Licinius Mureta in 62BCE, in the *scriptia continua* of classical Latin:

   PRIMVSDIGNITASINTAMTENVISCIENTIANONPOTEST
   ESSERESENIMSVNTPARVAEPROPEINSINGVLISLITTERIS
   ATQVEINTERPVNCTIONIBUSVERBORVMOCCVPATAE[1]

   A fluent Latin reader would parse this string (in modern orthography) as *Primus dignitas in tam tenui scientia non potest esse; res enim sunt parvae, prope in singulis litteris atque interpunctionibus verborum occupatae.*

   Some strings can be parsed in multiple ways, but you are not concerned with that. You want to know, given a string S of n characters, can it be segmented into words *at all*? Assume you have access to a subroutine $\textsc{IsWord}(i, j)$ that takes indices i, j with $i \leq j$ as input and indicates whether $S[i, \ldots, j]$ is a "word" in the foreign language, and that it takes constant time to run.

   Design an algorithm that solves this problem in $O(n^2)$ time.

2. Recall that a subsequence of an array is obtained by deleting any number of positions (possibly none, possibly all). A subarray is a sequence in which the positions that are not deleted form an interval (possibly empty). For example, consider the array $[-1, -2, -3, -4, -5]$. A valid subsequence is $[-1, -3, -5]$; it is not a subarray. A valid subarray is $[-2, -3]$.

   You are given an array $A[1, \ldots, n]$ of integers and want to find the maximum sum of the elements of (a) any subsequence, and (b) any subarray. The sum of an empty subarray is 0. For the example above, the maximum-sum subarray and subsequence has length zero.

   Design an $O(n)$ algorithm for both problems.

Regular problems

3. [Graded, 3 points] You want to go running and have n minutes to spare. You want to run as long a distance as possible, but your exhaustion level cannot exceed a given limit e. Initially

[1] "First of all, dignity in such paltry knowledge is impossible; this is trivial stuff, mostly concerned with individual letters and the placement of points between words."

your exhaustion level is zero. During each minute, you can choose to either run or rest for the whole minute. When you choose to run the i-th minute, you run exactly $d[i]$ feet during that minute, and your exhaustion level increases by one. When you choose to rest, you run zero feet during that minute, and your exhaustion level decreases by one (if your exhaustion level is already zero, it will stay zero). Moreover, when you choose to rest, you must continue to rest until your exhaustion level reaches zero; once it reaches zero, you can again choose to run or rest. Finally, your exhaustion level at the end of your run must be zero.

For example, for $e = 2$ and $d[1, \ldots, 5] = (500, 300, 400, 200, 1000)$, the best you can do is to run during minutes 1 and 3, and rest the other minutes, so the answer is $500 + 400 = 900$.

Develop an algorithm that takes a positive integer e and an array $d[1, \ldots, n]$ of $n \geq 1$ positive integers as input, and ouputs the maximum distance you can run subject to the constraints above. Your algorithm should run in time $O(ne)$ and space $O(e)$.

4. When you were little, every day on your way home from school you passed the house of your grandmother. If you stop by for a chat on day i, Grandma would give you a number ℓ_i of lollipops but also tell you that she won't give you any more lollipops for the next k_i days. For example, if day 1 is a Monday and $k_1 = 3$, then if you visit her that day, you would have to wait patiently until Friday to get your next lollipop.

   Design an $O(n)$ algorithm that takes as input the numbers (ℓ_i, k_i) for $i \in \{1, 2, \ldots, n\}$, and outputs the maximum number of lollipops you can get during those n days.

5. The library has n books that must be stored in alphabetical order on adjustable-height shelves. Each book has a height, and a thickness. The width of the shelf is fixed at w, and the sum of the thicknesses of books on a single shelf cannot exceed w. The next shelf will be placed atop the tallest book on the shelf. You can assume the shelving takes no vertical space.

   Design an algorithm that minimizes the total height of shelves used to store all the books. You are given the list of books in alphabetical order. Your algorithm should run in time $O(n^2)$.

Challenge problem

6. There is a famous joke-riddle for children:

   > Three turtles are crawling along a road. One turtle says, "There are two turtles ahead of me." Another turtle says, "There are two turtles behind me." The third turtle says, "There are two turtles ahead of me and two turtles behind me." How could this have happened? Of course, the third turtle is lying!

   In this problem you have n turtles crawling along a road. Some of them are crawling side-by-side, so there may be turtles that are neither ahead nor behind one another. Each turtle makes a statement of the form: "There are a_i turtles ahead of me, and b_i turtles behind me."

   Your task is to find the minimal number of turtles that must be lying. More formally, let x_i denote the position along the road of turtle i, $1 \leq i \leq n$. Some turtles may be at the same position. Turtle i tells the truth if and only if a_i is the number of turtles j such that $x_j > x_i$ and b_i is the number of turtles j such that $x_j < x_i$. Otherwise, turtle i is lying.

   Design an $O(n \log n)$ algorithm for this problem.

Programming problem

7. SPOJ problem Coins Game (problem code MCOINS).