Computer Science C73 October 4, 2023
Scarborough Campus University of Toronto

Homework Assignment #5
Due: October 25, 2023, by 11:59 pm

• ***You must submit your assignment through the Crowdmark system.*** *You will receive by email an invitation through which you can submit your work in the form of separate PDF docucments with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups.* ***The course policy that limits the size of each group to at most two remains in effect:*** *submissions by groups of more than two persons will not be graded.*
• *It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
• *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
• *For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.*
• *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description. Do not provide executable code.*
• *Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on* ***the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation****.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Recall that a dynamic programming algorithm to solve a given problem P involves the following elements:
(a) *A definition of the subproblems to be solved (and from whose solution we will compute the solution to P — see (c) below).*
(b) *A recursive formula to compute the solution to each subproblem from the solutions to smaller subproblems. This induces a partial order on the subproblems defined in (a).*
(c) *A way to compute the solution to P from the solutions to the subproblems computed in (b).*

Proving the correctness of a dynamic programming algorithm amounts to justifying (i) why the recursive formula in step (b) correctly computes the subproblems defined in step (a), and (ii) why the computation in (c) yields a solution to the given problem. Part (ii) is often immediate from the definition of the subproblems.

Question 1. (20 marks) Let $A[1..n]$ be a non-empty array of integers (they may be positive, negative, or zero) and k be a positive integer. We want to find the maximum sum of a (contiguous) subarray of A ***of length at least*** k. That is, we want to find $\max\{\sum_{t=i}^{j} A[t] : 1 \le i \le j \le n \text{ and } j - i + 1 \ge k\}$.

Use dynamic programming to give an $O(n)$ algorithm that solves this problem. Describe your algorithm in pseudocode, explain why it is correct, and analyze its running time.

Hint: You may find it useful to first solve the special case $k = 1$ (i.e., when there is no length constraint on the subarray), and then generalize. Do not submit your answer to the special case. For the general case you may find it useful to precompute the sum of every length k subarray of A.

Question 2. (20 marks) Recall that a ***substring*** of a string x is a (possibly empty) string x' such that, for some (possibly empty) strings y and z, $x = yx'z$. A string is a ***palindrome*** if it reads the same forward and backward — for example, the inscription on the fountain in the courtyard of Agia Sophia: $\nu\iota\psi o\nu\alpha\nu o\mu\eta\mu\alpha\tau\alpha\mu\eta\mu o\nu\alpha\nu o\psi\iota\nu$, which (with inter-word spaces omitted) means "wash the sins, not just the face", is a palindrome.

Give an efficient dynamic programming algorithm that takes a string $x[1..n]$ as input and returns the ***number*** of substrings of x that are palindromes. Justify the correctness of your algorithm and analyze its running time.

Question 3. (25 marks) (DPV Exercise 6.2, slightly rephrased.) You are going on a long trip. Along the way, there are n hotels, at distances $d_1 < d_2 < \ldots < d_n$ from your starting point. You are only allowed to stop at these hotels, but you can choose at which ones to stop — except that you ***must*** stop at the last hotel, your destination.

Ideally, you want to travel 300 km between successive stops, but this is not always possible because the spacing of the hotels may not allow it. If you travel a distance of x km between successive stops, you pay a penalty of $(300 - x)^2$ for that interval of travel. For a given sequence of stops made during the trip, the ***total penalty*** for that sequence, is the sum of the penalties for the between-stop intervals. (We think of the starting point as a "stop".)

You wish to determine an optimal sequence of stops, i.e., one that minimizes the total penalty.

a. Consider the following greedy strategy, applied after each stop. Among all hotels you have not already passed or stopped at, find one that is closest to 300 km from your present location; that hotel will be your next stop. Give a simple example that demonstrates that this greedy algorithm does not yield an optimal sequence of stops.

b. Give a polynomial-time dynamic programming algorithm that, given the sorted sequence $d_1, d_2, \ldots, d_n$, returns a pair (S, p), where $S = s_1, \ldots, s_k$ is an optimal sequence of stops in increasing order, and p is the associated total penalty. Justify the correctness of your algorithm and analyze its running time.