Computer Science C73
Scarborough Campus

October 7, 2019
University of Toronto

Homework Assignment #5
(worth 4% of the course grade)
Due: October 21, 2019, by 10 am

- *You must submit your assignment as a PDF file through the MarkUs system by logging in with your UTORid at* `https://markus.utsc.utoronto.ca/cscc73f19`*. To work with a partner, you and your partner must form a group on MarkUs.*
- *It is your responsibility to ensure that the PDF file you submit is 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 file 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.*
- *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.**"

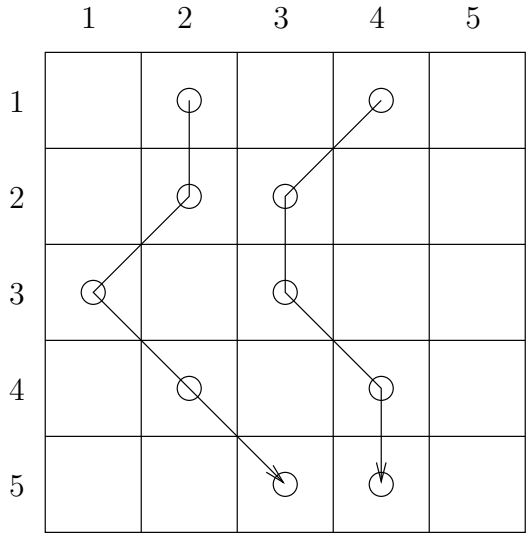
Recall that a dynamic programming algorithm to solve a given problem P involves the following elements:

(a) *A definition of a polynomial number of subproblems that will 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. (10 marks) Consider an $n \times n$ grid, where n is a positive integer. A ***path*** of this grid is a sequence of n grid squares, the first of which is on the first row, the last is on the last row, and each square of the path is one row higher than the previous one and either on the same column or an adjacent column. More precisely, a path is a sequence of grid squares $(1, j_1), (2, j_2), \ldots, (n, j_n)$, such that for every

i, $1 < i \leq n$, either $j_i = j_{i-1}$ or $j_i = j_{i-1} - 1$ (and $j_{i-1} > 1$) or $j_i = j_{i-1} + 1$ (and $j_{i-1} < n$). The diagram below illustrates two paths on a 5×5 grid.



Each square (i, j) on the grid contains a number $G[i, j]$. The ***value*** of a path is the sum of the numbers contained in the squares on that path. A path is ***optimal*** if it has maximum value.

a. (1 marks) Consider the following greedy algorithm:

$i := 1$
$j_1 :=$ a column that contains the maximum element in row 1
while $i < n$ **do**
  $i := i + 1$
  ▷ set j_i to whichever of columnns $j_{i-1} - 1$, j_{i-1}, or $j_{i-1} + 1$ contains a larger value in row i
  $j_i := j_{i-1}$
  if $j_{i-1} > 1$ **and** $G[i, j_{i-1} - 1] > G[i, j_i]$ **then** $j_i := j_{i-1} - 1$
  if $j_{i-1} < n$ **and** $G[i, j_{i-1} + 1] > G[i, j_i]$ **then** $j_i := j_{i-1} + 1$

Give a counterexample for $n = 3$ to show that the path $(1, j_1), (2, j_2), \ldots, (n, j_n)$ constructed by this algorithm is not necessarily optimal. Is there a counterexample for $n = 2$? Justify your answer.

b. (7 marks) Describe a polynomial time dynamic programming algorithm that finds the value of an optimal path, given the array G of numbers on the grid squares. Explain why your algorithm is correct. What is the time complexity of your algorithm?

c. (2 marks) Modify your algorithm in part (b) so that it also computes an optimal path (not only the value of an optimal path). More specifically, your algorithm also computes an array $P[1..n]$, so that $(1, P[1]), (2, P[2]), \ldots, (n, P[n])$ is an optimal path.

Question 2. (10 marks) A non-empty sequence of numbers is called ***single-peak*** if it consists of a (possibly empty) prefix that is increasing followed by a (possibly empty) suffix that is decreasing. For example, $2, 3, 5, 7$; $2, 3, 7, 5$; $2, 7, 5, 3$; and $7, 5, 3, 2$ are single-peak sequences. In contrast, $5, 2, 3, 7$; $3, 5, 2, 7$; and $3, 2, 7, 5$ are not single-peak sequences.

Give a polynomial-time dynamic programming algorithm, which takes as input a non-empty sequence of ***distinct*** integers represented as an array $A[1..n]$ with $n \geq 1$, and returns the ***length*** of a longest single-peak subsequence of A. (Recall that the elements of a subsequence of A are not necessarily consecutive elements of A, but must appear in the same order as in A.) Briefly explain why your algorithm is correct. What is its running time?

[Question to think about (but not to turn in as part of your answer): Modify your algorithm to compute a longest single-peak subsequence of A (not just its length).]