Computer Science C73
Scarborough Campus

October 5, 2022
University of Toronto

Homework Assignment #5
Due: October 19, 2022, 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.*

• *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. (15 marks) Let $A[1..n]$ be a non-empty array of numbers, which may be negative, zero, or positive, and are not necessarily integers. We want to find the maximum number p that is the product of

all the numbers in subarray $A[i..j]$ of A, over all i, j such that $1 \leq i \leq j \leq n$. (If $i = j$, the product of all the numbers in $A[i..j]$ is, by defintion, the number $A[i]$.)

Use dynamic programming to give an $O(n)$ algorithm that solves this problem. (Assume that basic arithmetic operations and comparisons involving the numbers in A take $O(1)$ time each.) Describe your algorithm in pseudocode, explain why it is correct, and analyze its running time.

Hint: As a warm-up exercise, solve the related problem of finding the maximum ***sum*** (instead of product) of all elements in some subarray $A[i..j]$ of A. Do not submit your answer to this warm-up exercise.

Question 2. (10 marks) (DPV Exercise 6.6, slightly rephrased.) The table below defines a "binary operation" on three symbols a, b, and c. Note that this operation is neither commutative nor associative.

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

Give a polynomial-time dynamic programming algorithm that, given a string of symbols over $\{a, b, c\}$, determines whether it is possible to parenthesize the string so that the value of the resulting expression is a. If so, your algorithm should return true; otherwise it should return false.

For example, on input $bbbbac$ the algorithm should return true, since $((b(bb))(ba))c = a$; while on input aaa it should return false, since $a(aa) = b$ and $(aa)a = c$.

Describe your algorithm in pseudocode, explain why it is correct, and analyze its running time.

Question 3. (15 marks) A digital image is a two-dimensional array $P[1..m, 1..n]$. Each entry (i, j) of P is called a ***pixel*** and contains a number, encoding colour. For example, an 8-bit grayscale encodes 256 different intensities of light, representating variations of gray: 0 is total absence of light (black) and 255 is total presence of light (white). Suppose we want to resize our $m \times n$ image P to an $m \times (n-1)$ image with minimal loss of visual information, as perceived by the human eye. Deleting a column of pixels is not necessarily the best choice as the column might contain many "edges" of objects that are crucial in interpreting the image. Here is a better way of doing it.

A (vertical) ***seam*** of P is a sequence of pixels $S = (1, j_1), (2, j_2), \ldots, (m, j_m)$, one per row, so that the columns of successive pixels differ by at most one: for each $i \in [1..m]$, $1 \leq j_i \leq n$; and for each $i \in [2..m]$, $|j_i - j_{i-1}| \leq 1$. (The figure below shows in red a vertical seam, and a similarly defined horizontal seam that we will not discuss here.)

Figure 1: Vertical and horizontal seams of an image

We want to reduce the number of columns of the image by one by removing a seam whose deletion is least noticeable. Pixels containing values very similar to those of their neighbours' are less noticeable when removed. Thus we define the "cost" of removing pixel (i, j) as

$$C(i,j) = \begin{cases} \infty, & \text{if } j = 1 \text{ or } j = n \\ |P[i,j] - P[i,j-1]| + |P[i,j] - P[i,j+1]|, & \text{otherwise} \end{cases}$$

(The first clause makes it prohibitively expensive to remove pixels at the vertical edges of the image.) The cost of removing the seam $S = (1, j_1), (2, j_2), \ldots, (m, j_m)$ is defined as $C(S) = \sum_{i=1}^{m} C(i, j_i)$.

The vertical seam depicted on the image of the previous page is a minimum-cost seam. Note how it tends to skirt object boundaries, such as the rock at the bottom, by seeking pixels whose horizontal neighbours are similar. (The same comment applies to the horizontal seam shown in that image, except that we now we seek pixels whose vertical neighbours are similar.)

Give a polynomial time dynamic programming algorithm that, given an image P, returns a minimum-cost seam of P. Note that the algorithm should return the seam itself, not its cost. Describe your algorithm in pseudocode, explain why it is correct, and analyze its running time.

Note: The idea outlined in Question 3 is a simplified version of a technique used for image resizing, for example so that it can be displayed on a different device. There are various techniques used to this end, such as "letterboxing", cropping, or rescaling along the horizontal or vertical axis. Letterboxing, however, introduces unsightly black bars on the edges of the output; cropping may eliminate important elements of the image; and rescaling may create unnaturally elongated or flattened images. "Seam carving", invented by the graphics researchers Shai Avidan and Ariel Shamir and now implemented in Photoshop, is an alternative technique for image resizing that repeatedly removes (or adds) minimum-cost (vertical or horizontal) seams. In many instances it avoids the pitfalls of letterboxing, cropping, and rescaling, and produces realistic facsimilies of the original image. At the core of this technique is the identification of a minimum-cost seam as described in Question 3, except that Avidan and Shamir use more sophisticated cost functions. In Figure 2 you can see, from left to right, an original image and then that image resized by cropping, rescaling, and seam carving.

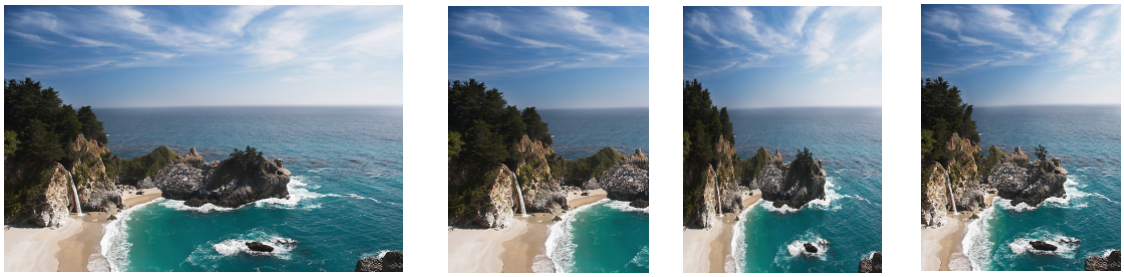

Figure 2: An original image resized by cropping, rescaling, and seam carving

For more information on this technique you can read the paper that introduced this resizing technique, and from which the figures on this and the previous page are taken; or watch a video presentation of the paper. (Consulting these sources does not constitute a violation of the course's homework collaboration policy.)