CS2705 : Programming and Data Structures

Assignment 1

August 22, 2022

Submission date: Aug 31, 5.00pm | Max. Marks : 8

Instructions

- The assignment is graded. All questions are compulsory and have to be solved individually.
- You are required to submit the code on repl.it by following the given instructions.
- You are expected to write code completely on your own. Use of unfair means found will be penalized and reported to appropriate higher level committees.

---

1. (3 points) There are m horizontal lines and n vertical lines drawn on the ground such that there are $m \times n$ cross-points. Students are standing at a subset of these cross-points. The lines are numbered from top to bottom and left to right, starting from 0. A window is a rectangle specified by integer co-ordinates which give the left-top corner and right-bottom corner of the window. Given a window, the goal is to output the number of cross-points in the window that are *not* occupied by students.

   Input format First line contains four integers in the following order, the value of m, the value of n, number of students s, and number of windows w, each separated by a white-space. Line 2 to line $s + 1$ contain the position of students one per line, that is, a cross-point, specified by a pair of row and column numbers, separated by a comma. Line $s + 2$ to $s + 2 + w - 1$ contain the set of windows one per line, that is, a pair of two cross-points, specified by the left-top and right-bottom corners, separated by a comma. Each corner point is given by a pair of indices, separated by a comma.

   Output format The output consists of w integers one per line. The integer on the i-th line of the output denotes the number of unoccupied cross-points in the i-th window in the input.

   Example

   Input:

   4 3 7 3

   0,0

   0,1

   1,2

   2,0

   2,2

   3,1

3,2

0,0,1,2

1,1,2,2

0,1,3,2

Output:

3

2

3

Explanation: The input can be visualized as an $m \times n$ matrix shown below such that the topmost row and the leftmost column is numbered 0, and 'X' represents a position where a student is standing, and '$-$' represents a position which is not occupied.

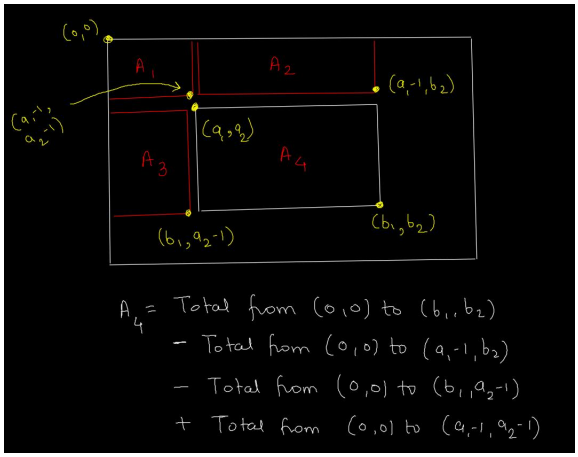
$X \quad X \quad -$
$- \quad - \quad X$
$X \quad - \quad X$
$- \quad X \quad X$

The expected output is simply the number of occurrences of '$-$' in each rectangular window.

(a) Implement an algorithm with running time $O(mnw)$.

(b) Implement an algorithm that runs faster, that is, in $O(mn + w)$ time using additional space. Follow the idea given below.

The modification to get $O(mn + w)$ time is to be able to answer for each window in $O(1)$ time. This can be done as follows. First let $cnt(i, j)$ indicate the number of empty cross-points for the window $(0, 0)$ to (i, j). Compute $cnt(i, j)$ for each pair of indices. Then count the number of empty cross-points for a rectangular window (a_1, a_2) to (b_1, b_2) by applying inclusion-exclusion principle. See the following diagram for reference.



2. (3 points) Given two doubly linked lists such that the second list is *attached* to the first one, your goal is to *fit in* the second list in the first. The fitting in has to be done at a specified position in the first linked list.

Input format The first line contains two integers which denote the length of the first and second list respectively. The next two lines contain the two doubly linked lists, one per line. Each linked list contains integers, given by a set of values in that order, separated by a white-space. The third line contains the position where the second list is to be *fit in* into the first. The position is a tuple (X, i) where $X \in \{L, R\}$ and i is a positive integer. Assume that the linked list nodes are numbered from

1. The integer i is guaranteed to be between 1 and the number of elements in the first linked list. The position (L, i) implies that the second doubly linked list has to be fit in the first linked list *before* the i-th element from the left. The position (R, i) implies that the second doubly linked list has to be fit in the first one *after* the i-th element from the right.

Output format The combined list printed in original and reverse order, one per line, wherein each value (node) is separated by a white-space.

Note: Implement a doubly linked list extending the singly linked list code discussed in the class. To print the list in the forward direction, use the standard print and access next pointers. To print the list in the reverse direction, use the prev pointers.

Also note that not using a doubly linked list (possibly using an array or a vector) and printing the correct output will fetch you zero credit.

Example-1
Input:
4 3
10 20 30 40
70 80 90
L 3
Output:
10 20 70 80 90 30 40
40 30 90 80 70 20 10

Example-2
Input:
4 3
10 20 30 40
70 80 90
R 3
Output:
10 70 80 90 20 30 40
40 30 20 90 80 70 10

3. (2 points) Given two sorted arrays of integers containing n_1 and n_2 elements each, your goal is to find the median of the combined sorted array. You are expected to do this *without* declaring another array of size $n_1 + n_2$. In particular, you cannot combine the two arrays into a single array, sort and output the median.

   The median is defined as *middle* element in the combined array if $n_1 + n_2$ is odd, otherwise the median is the average of the two *middle* elements in the combined array.

   Input format The first line contains the dimensions n_1 and n_2 of two arrays, separated by a white-space. The next two lines contain the array elements themselves, one array per line. Each array is a white-separated list of elements in it in that order.

   Output format The median value truncated up to 2 decimal places.

   Example

   Input:

   3 3

   2 5 9

   1 2 3

   Output

   2.50

   Explanation The combined (sorted) array is 1, 2, 2, 3, 5 ,9. Hence the median is $\frac{2+3}{2} = 2.5$.

4. (0 points) This is an ungraded question for your practice. Given an array of distinct integers and an integer x, find 3 integers in A that are closest to x in terms of their absolute difference with x.