

C++ Programming

1D Arrays Homework 3

Mostafa S. Ibrahim

Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher

PhD from Simon Fraser University - Canada

Bachelor / Msc from Cairo University - Egypt

Ex-(Software Engineer / ICPC World Finalist)

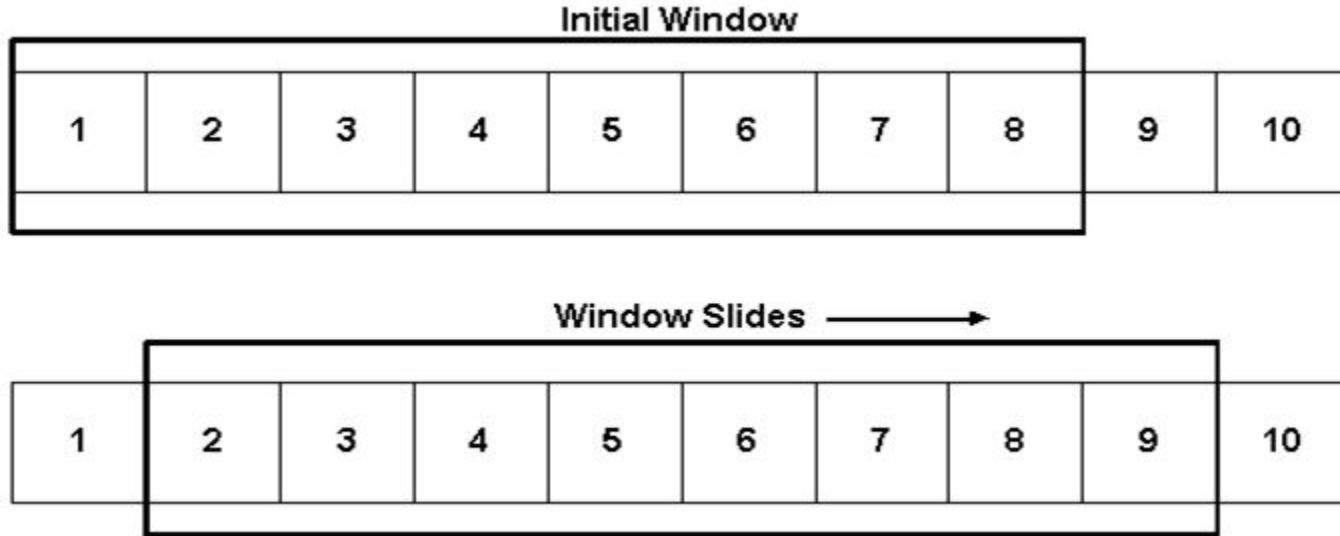


Problem #1: Recamán's sequence

- The first terms of this sequence are 0, 1, 3, 6, 2, **7**, ...
 - So last term **value** is 7 and its **index** is 5 (zero based)
 - The next value is either:
 - **LastValue - LastIndex - 1** if the following 2 conditions are satisfied:
 - value > 0 and It did not appear before
 - E.g. 7 (last value) - last index (5) - 1 = 7-5-1 = 1 (> 0 but already exists)
 - Or **LastValue + LastIndex + 1** = 7+5+1 = 13
- Read integer zero-based index ([1, 200]) and print the value of this index
 - E.g. (6 ⇒ 13), (9 ⇒ 21), (17 ⇒ 25)
- Don't use nested loops
- The series is: 0, 1, 3, 6, 2, 7, **13**, 20, 12, **21**, 11, 22, 10, 23, 9, 24, 8, **25**, 43

Background: Fixed Sliding Window

- Indicate a group of consecutive number. Fixed and variable size
 - You slide to next window



Background: Fixed Sliding Window

- Assume a list: **1 0 3** -4 2 -6 9
- Sliding window (sublist): 3
- Let's print all windows of length 3 and their sum
 - 1 0 3 \Rightarrow sum = 4
 - 0 3 -4 \Rightarrow sum = -1 [observe 0 3 are common]
 - 3 -4 2 \Rightarrow sum = 1
 - -4 2 -6 \Rightarrow sum = -8
 - 2 -6 9 \Rightarrow sum = 5
- Observe the relation between 2 consecutive windows:
 - They share all the elements except a change in the first / last element
- Variable sliding window: its size grows and shrinks

Problem #2: Fixed sliding window

- Read Integers K and N, (where $K \leq N$). then read $N < 200$ integers.
- Find **sub-array** (consecutive numbers) of K elements that has maximum sum
- Input 3 7 **1 0 3 -4 2 -6 9**
 - Let's list all sub-arrays of length 3
 - 1 0 3 \Rightarrow sum = 4
 - 0 3 -4 \Rightarrow sum = -1
 - 3 -4 2 \Rightarrow sum = 1
 - -4 2 -6 \Rightarrow sum = -8
 - 2 -6 9 \Rightarrow sum = 5
- Output: 4 6 5 (Sub-array from indices 4 to 6 has maximum sum of 5)
- Can you do it without nested loops? There are 2 ways.

Problem #3: Count increasing subarrays

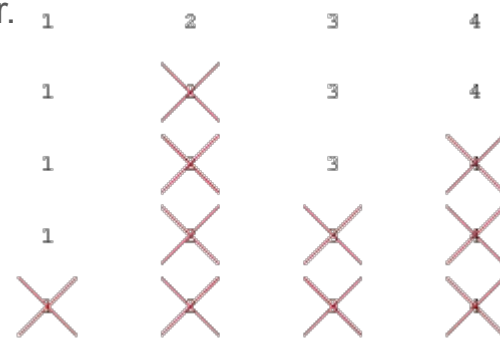
- Read an Integer N, then read N (< 200) integers.
- Output: **Count** how many **subarrays** are **increasing** in the array.
- E.g. If input is 1 2 3 4
 - We can find all sublists of length 1 \Rightarrow [1], [2], [3], [4]
 - All sublists of length 2 \Rightarrow [1, 2], [2, 3], [3, 4]
 - All sublists of length 3 \Rightarrow [1, 2, 3], [2, 3, 4]
 - All sublists of length 4 \Rightarrow [1, 2, 3, 4]
- Inputs \Rightarrow Outputs
 - 4 1 2 3 4 \Rightarrow 10 [10 sub-arrays from previous example, all are increasing]
 - 4 4 3 2 1 \Rightarrow 4 [only sub-arrays of length 1 can be considered]
 - 4 10 20 1 5 \Rightarrow 6
- Easy using 3 nested loops. Medium using 2 loops. Hard with 1 loop

Problem #4: Josephus problem

- Read integers N (< 200) and K (≤ 1000000). *Code for small K first*
 - Find the game winner for following game:
- There is a group of N people in Circle numbered 1, 2, ..., N
 - Someone is a master of the game.
 - He starts from Person #1. Count K . Then remove this person from the circle.
 - He keeps doing so till only 1 person remains. This is the winner.

Input 4 2

- Means we have people: 1, 2, 3, 4. Master starts at 1
- Count 2 persons (2 removed), start from 3
- Count 2 persons (4 removed), start from 1
- Count 2 persons (3 removed), 1 is winner



Output

- People removed in order: 2 4 3 1 [same answer for 10 2 why?]

Problem #4: Josephus problem

- Input \Rightarrow Output
 - 7 1 \Rightarrow 1 2 3 4 5 6 7
 - 7 2 \Rightarrow 2 4 6 1 5 3 7
 - 7 3 \Rightarrow 3 6 2 7 5 1 4
 - 7 4 \Rightarrow 4 1 6 5 7 3 2
 - 7 5 \Rightarrow 5 3 2 4 7 1 6
 - 7 6 \Rightarrow 6 5 7 2 1 4 3
 - 7 7 \Rightarrow 7 1 3 6 2 4 5
 - 7 14 \Rightarrow 7 2 6 3 5 4 1
 - 7 1000 \Rightarrow 6 3 2 1 4 7 5
 - 7 99999 \Rightarrow 4 7 5 2 1 3 6

Problem #5: longest subarray

- Read integer N (< 1000) then N read numbers each is either 0 or 1.
- Find the longest **subarray** with **number of zeros = numbers of ones**
 - You can easily implement it using 3 loops
 - Or with little thinking using 2 loops (even with no extra arrays)
 - Hard: You can implement it without any nested loops
- Inputs \Rightarrow outputs
 - 7 1 0 0 0 1 1 1 \Rightarrow 6 (e.g. 100011 or 000111)
 - 19 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 \Rightarrow 8 (e.g. 00101101)
- Reduction
 - How may this problem be reduced to another problem: longest subarray of zero sum?

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”