Name: Roll Number:

ESO207: Data Structures and Algorithms

Programming Assignment 2

Due Date: 13th September 11:59pm, 2023

Total Number of Pages: 7 Total Points 100

Note :

- The questions have to be answered through a contest in Hackerrank. The contest has 4 challenges, each corresponding to a question. You have to submit your code through the contest.
- Link to Hackerrank contest: https://www.hackerrank.com/eso207-programming-assignment-2
- Additionally you must upload your solutions on Moodle as well. You need to upload 4 files corresponding to the 4 programs.
- Your codes will be checked for possible plagiarism of any sorts. If we find such cases, then we will possibly award an F grade.
- Allowed Languages for challenge code submission : C, C++
- Allowed libraries : stdio.h, stdlib.h for C and iostream, cstdlib, vector for C++

Question 1. (20 points) **Problem 1 - Number of Inversions**

You have an array A of size n. An inversion is defined as a pair (i, j), $1 \leq i < j \leq n$ such that $A[i] > A[j]$. You have to count the number of inversions in the array using the algorithm taught in lecture with time complexity $O(n \log n)$.

- **Input**

  First line of input consists of number of test cases T. Every test case has two lines first line contains a single integer N , next line has N integers $A_1, A_2, \ldots, A_N$

- **Output**

  For each test case, output the number of inversions for that array.

- **Constraints**
  - $1 \leq T \leq 10^3$
  - $1 \leq N \leq 10^5$
  - $-10^9 \leq A[i] \leq 10^9$

- **Sample Input**

```
2
5
1 4 3 -5 1
6
1 2 3 4 5 6
```

- **Sample Output**

```
6
0
```

Question 2. (20 points) **Problem 2 - Akshunya's Max Tree**

Akshunya has an array containing N **distinct** elements. He loves large numbers so he wants to order all the elements such that the largest elements are closest to him. So he follows this process to transform the array into a binary tree.

— He chooses the largest element of the array as the root.
— If the left subarray of this element is non-empty, he repeats this process on the left subarray to make the left subtree of this node (using the same process). If it is empty then the root has no left child.
— Similarly, if the right subarray of this element is non-empty, he repeats this process on the right subarray to make the right subtree of this node (using the same process). If it is empty then the root has no right child.

After the process is complete, he wants to know the depth of each node in the tree. Where the depth of a node is defined as the number of edges in the path from the root node to that node. Read the explanation for more understanding.

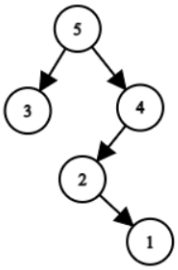
- **Input** First line of input consists of number of test cases T. Every test case has two lines first line contains a single integer N , next line has N integers $A_1, A_2, \ldots, A_N$
- **Output**
  For each test case, output n integers where i^{th} integer denotes the depth of i^{th} element of array, in the constructed tree.
- **Constraints**
  - $1 \leq T \leq 10^3$
  - $1 \leq N \leq 10^3$
  - $1 \leq A[i] \leq 10^5$
- **Sample Input**

```
2
5
3 5 2 1 4
7
2 10 5 21 9 12 1
```

- **Sample Output**

```
1 0 2 3 1
2 1 2 0 2 1 2
```

- **Explanation**
  In first test case, the Binary Tree would look like this.



Thus Depth of nodes would be-

Node	3	5	2	1	4
Depth	1	0	2	3	1

Question 3. (30 points) **Problem 3 - Stock BST**

Avi has just started investing in the stock market. But he has no knowledge of where to invest so he asks 2 of his best friends Bansal and Shrey. Bansal has an optimistic personality thus he always gives suggestions for buying stocks and Shrey has a pessimistic personality so he always gives advice of selling stocks. Now Avi wants to maintain his portfolio of stocks so he consults his friend Radha and she suggests that he maintain them in a Binary Search Tree.

For each of the next d days, he meets one person among Bansal, Shrey, or Radha. If he meets Bansal on i^{th} day, he advises him to buy the stock s_i and if Avi doesn't already have that stock in his portfolio then he buys it. If he meets Shrey on i^{th} day, he advises him to sell the stock s_i and if he has that stock in his portfolio, he sells that stock. If he meets Radha on i^{th} day she asks him to check if the stock s_i is present in his portfolio.

Help Avi maintain his stock portfolio and answer all the queries asked by Radha.

- **Input**
  First line of input contains the number of days D.
  The next d lines contains a character c and a number x.
  If c is B, then Avi met Bansal and got advice to buy stock x.
  If c is S, then Avi met Shrey and got advice to sell stock x.
  If c is R, then Avi met Radha and was asked if the stock x was present in his portfolio.
- **Output**
  For each of the days of type R x, print "YES" (without the quotes) if the stock x is present in the portfolio else print "NO" (without the quotes).
- **Constraints**
  - $1 \leq d \leq 10^5$
  - c = B | S | R
  - $1 \leq x \leq 10^5$
- **Sample Input**

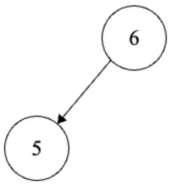
```
10
B 6
B 5
B 9
S 9
R 5
B 7
S 3
R 7
S 6
R 6
```

- **Sample Output**

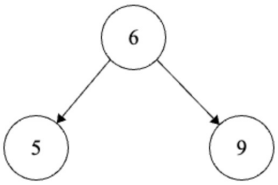
```
YES
YES
NO
```

- **Explanation**
  This would be the BST after-
  1st day:

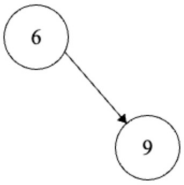
  2nd day:



  3rd day:



  4th day:



  Thus, on the 5th day, the stock x = 6 is present in the portfolio.

Question 4. (30 points) **Problem 4 - Positive Segments**

You have an array A of size n, containing -1 or 1 only, and s segments (not necessarily different). Each segment is defined by 2 integers l_i and r_i ($1 \leq l_i \leq r_i \leq n$) and represent the subarray $A_{l_i}, A_{l_i+1}, ..., A_{r_i}$ of the array A.

A segment $[l_i, r_i]$ is called positive if the sum of elements in the subarray is strictly greater than 0. That is $A_{l_i} + A_{l_i+1} + ... + A_{r_i} \geq 0$. For example, if $A = [-1, -1, 1, -1, 1]$, then the segments $[3, 5]$ and $[5, 5]$ are positive (sum of elements is greater than 0), but the segments $[1, 2]$ or $[1, 5]$ are not positive.

Now you have q queries. In each query, you are given an integer j ($1 \leq j \leq n$) such that you set $A_j = |A_j|$ (Absolute value of A_j). You have to find the minimum number of queries after which at least k ($\leq s$) of given segments become positive or tell it is impossible.

Note: It might happen that there already exists k positive segments.

- **Input**

  First line of input consists of number of test cases t. Every test case is defined as follows-
  - First line contains 2 integer n and s.
  - Next line contains n integers where i^{th} integer denotes A_i.
  - Following s lines contain 2 integers l_i and r_i defining i^{th} segment.
  - Next line contains 2 integers q and k.
  - Next line contains q integers where i^{th} integer denotes x corresponding to i^{th} query.

- **Output**

  For each test case, output a line containing the minimum number of queries needed for that case. If it is impossible in all the queries then output -1 in that case.

- **Constraints**
  - $1 \leq t \leq 10^2$
  - $1 \leq k \leq s \leq n \leq 10^5$
  - $a_i = -1$ or 1
  - $1 \leq q \leq 10^5$

- **Sample Input**

```
2
5 2
-1 -1 1 -1 -1
1 2
2 5
6 1
3 1 1 2 5 3
5 3
-1 -1 -1 -1 -1
1 3
2 2
1 1
3 2
2 4 5
```

- **Sample Output**

```
4
-1
```

- **Explanation**
  In the first test case, sum of elements of different segments after various queries-

Queries	Segment 1	Segment 2	Positive Segments
0	-2	-2	0
1	-2	-2	0
2	0	-2	0
3	0	-2	0
4	2	0	1
5	2	2	2
6	2	2	2

We have 1 positive segment after 4th query thus answer would be 4.
In the second test case, sum of elements of different segments after various queries-

Queries	Segment 1	Segment 2	Segment 3	Positive Segments
0	-3	-1	-1	0
1	-1	1	-1	1
2	-1	1	-1	1
3	-1	1	-1	1

There was no query after which we have at least 2 positive segments thus answer would be -1.