Contest

Content

— Super Stream Engineers
— Toggle Case
— Positive in Range

$\Bigg\}$ Scan through

$\xrightarrow{}$ assignment.

Contest	Re-attempt	Final
{ 100	80 }	max (100, 80)

Surprise $\longrightarrow$ Revision Class Intermediate.

23 - Dec

Avg PSP

70.96 $\longrightarrow$ 75 %

SuperStream Engineers

Problem Description

You've just been hired as a network engineer at SuperStream, a leading video streaming service. One of your first tasks is to optimize the number of video data packets sent to users based on their internet connectivity.

When a user hits "play," video data is transmitted in packets. If their device acknowledges these packets quickly, it means they have a strong connection and can receive more packets simultaneously for smoother streaming. If acknowledgments lag, fewer packets should be sent to prevent buffering.

Given an array **A**, where each entry represents the acknowledgment time (in milliseconds) for individual packets, and two integers **B** and **C**, can you determine if there's a continuous sequence of **B** packets with an average acknowledgment time less than or equal to **C** milliseconds? If so, it's a green signal (integer **1**) to send more packets. Otherwise, it's time to throttle back (integer **0**).

Note: For average, take the **floor** of *(sum/total number of elements)*. → integer division

B packets

limit in millisecs.

Problem Constraints

$1 <= N <= 10^5$

$1 <= A[i] <= 10^9$

$1 <= B <= N$

$1 <= C <= 10^9$

Input Format

First argument A is an array of integers.

The remaining arguments B and C are integers

Output Format

Return 1 if such a subarray exist and 0 otherwise (0/ 1)

Example Input

Input 1:

```
 A = [30, 25, 40, 35, 20, 45, 50, 55, 22, 18, 15],
 B = 3,
 C = 30
```

31

33

18

since 18 avg of 3 packets time < limit 30 → Green signal 1

Input 2:

```
 A = [4, 2, 2, 5, 1]
 B = 4
 C = 1
```

3

2

ans = 0

Example Explanation

Explanation 1:

```
Average of [30, 25, 40] = 31.67 milliseconds
Average of [25, 40, 35] = 33.33 milliseconds
Average of [40, 35, 20] = 31.67 milliseconds
Average of [35, 20, 45] = 33.33 milliseconds
Average of [20, 45, 50] = 38.33 milliseconds
Average of [45, 50, 55] = 50 milliseconds
Average of [50, 55, 22] = 42.33 milliseconds
Average of [55, 22, 18] = 31.67 milliseconds
Average of [22, 18, 15] = 18.33 milliseconds

From the data, we see that the sequence [22, 18, 15] has an average acknowledgment time of 18.33 milliseconds,
which is less than C = 30 milliseconds. Thus, Jake's device meets the criteria, and SuperStream's server can ramp
up the data packets to Jake's device for an enhanced streaming experience. Hence the answer is 1.
```

Bruteforce

```
int solve ( A, B, C) {
    for ( i = 0 ; i < N ; i++) {
        s = i
        e = i + B - 1
        if (e >= N)  break
        time = 0
        for ( j = s ; j <= e ; j++) {
            time += A[j]
        }
        avgtime = time/B
        if ( avgtime <= C ) { return 1 }
```

[s, e] = B
e - s + 1 = B
e - i + 1 = B
e = B + i - 1

TC : O(N B)

Prefix sum to optimize

```
    }
}  return 0
```

avg = total

i-B   i

A = ~~30~~ 25 40 35 20 45 50 55 22 18 15

Run a loop from 0 to B-1

95

if $\frac{total}{B}$ <= C return $\underline{\underline{1}}$

```
time = 0
for i ⟶ 0 to B-1 {
    time += A[i]
}
```

TC : O(N)

```
if (time/B) <= C  return 1

for i ⟶ B to N-1 {
    time += A[i]
    time -= A[i-B]
    if (time/B) <= C  return 1
}
return 0
```

Toggle Case

Problem Description

You are given a character string **A** having length **N**, consisting of only lowercase and uppercase latin letters.

You have to toggle case of each character of string **A**. For e.g 'A' is changed to 'a', 'e' is changed to 'E', etc.

Problem Constraints

1 <= **N** <= 10^5 $\longrightarrow$ O(N)

A[i] ∈ ['a'-'z', 'A'-'Z']

Input Format

First and only argument is a character string **A**.

Output Format

Example Input

Input 1:

```
 A = "Hello"
```

$\longrightarrow$
$$\begin{tikzcd}
H \arrow[d] & e \arrow[d] & l \arrow[d] & l \arrow[d] & o \arrow[d] \\
h & E & L & L & O
\end{tikzcd}$$

Input 2:

```
 A = "tHiSiSaStRiNg"
```

Example Output

Output 1:

```
 hELLO
```

Output 2:

```
 ThIsIsAsTrInG
```

Pseudocode

Given string s

```
for i ⟶ 0 to N-1 {   // N = length of s
    ch = s[i]
    // condition of lowercase
    if (ch >= 'a' && ch <= 'z') {
        s[i] = (char) (ch - 32)
    }
    else {
        s[i] = (char) (ch + 32)
    }
}
```

'A' ↓ 65   'a' ↓ 97

TC: O(N)

SC: O(1)

Positive in Range

Problem Description

You are working on a project to analyze profit for a given set of days. You have been given an array **A** with profit for **N** days. You also have **Q** queries represented by a 2D array **B** of size **Qx2**. Each query consists of two integers **B[i][0]** and **B[i][1]**.

For every query, your task is to find the count of non-negative profit in the range from **A[B[i][0]]** to **A[B[i][1]]**.

$>= 0$

Problem Constraints

|A| = N
|B| = Q
$1 <= N, Q <= 10^5$
$-10^9 <= A[i] <= 10^9$
0 <= B[i][0] <= B[i][1] <= N - 1

Input Format

First arguemnt A, is an array
Second argument B, is a matrix

Output Format

Return an array.

Example Input

Input 1:

```
A = [1, -1, 0]  ⟶ 1 0 1
B = [[0, 2],  ⟶ 2
     [1, 2]]  ⟶ 1
```

Input 2:

```
A = [-1, -2]  ⟶ 0 0
B = [[0, 0],  ⟶ 0
     [1, 1]]  ⟶ 0
```

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ 1 & -1 & 0 \end{array}$$

$\uparrow$

l	r	
0	2	2
1	2	1

-1 -2

0

0

Example Output

Output 1:

```
[2, 1]
```

Output 2:

```
[0, 0]
```

Example Explanation

For Input 1:

```
Consider 0-based indexing:
Number of non-negative elements from [0, 2] is 2.
Number of non-negative elements from [1, 2] is 1.
```

For Input 2:

```
Number of non-negative elements from [0, 0] is 0.
Number of non-negative elements from [1, 1] is 0.
```

Bruteforce

		0	1	2	3	4	5	6
A	=	-5	2	10	-5	-10	0	6
A	=	0	1	1	0	0	1	1

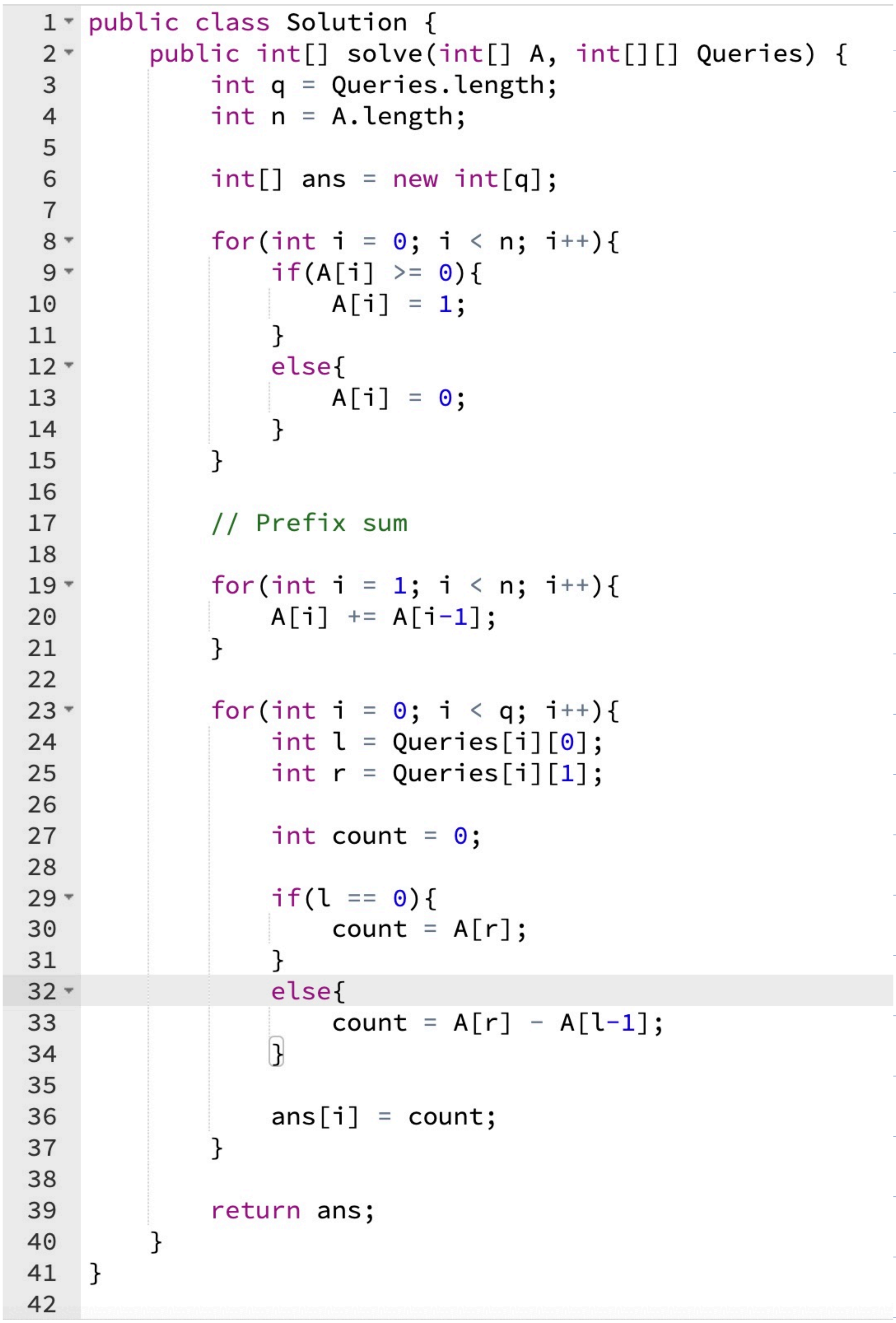
l	r	
0	6	4 $\longrightarrow$ loop throug l - r and sum up.
2	4	1

. . . . .

		0	1	2	3	4	5	6
A	=	0	1	1	0	0	1	1
PS	=	0	1	2	2	2	3	4

l	r		
0	6	PS [6]	0 6
2	4	PS [4] - PS [1]	

sum (l . . . . r) $\longrightarrow$ PS [r] - PS [l-1]



```java
public class Solution {
    public int[] solve(int[] A, int[][] Queries) {
        int q = Queries.length;
        int n = A.length;

        int[] ans = new int[q];

        for(int i = 0; i < n; i++){
            if(A[i] >= 0){
                A[i] = 1;
            }
            else{
                A[i] = 0;
            }
        }

        // Prefix sum

        for(int i = 1; i < n; i++){
            A[i] += A[i-1];
        }

        for(int i = 0; i < q; i++){
            int l = Queries[i][0];
            int r = Queries[i][1];

            int count = 0;

            if(l == 0){
                count = A[r];
            }
            else{
                count = A[r] - A[l-1];
            }

            ans[i] = count;
        }

        return ans;
    }
}
```