closed

a

Session
ID: certQ2WFKG-WHNKXKZFGEASR2AR
Time limit: 120 min.

Status: closed
Started on: 2014-01-14 18:07 UTC

Score:

60

of 100

★★★ 1. CountBoundedSlices

score: 60 of 100

Calculate the number of slices in which (maximum - minimum <= K).

Task description

An integer K and a non-empty zero-indexed array A consisting of N integers are given.
A pair of integers (P, Q), such that 0 ≤ P ≤ Q < N, is called a *slice* of array A.
A *bounded_slice* is a slice in which the difference between the maximum and minimum values in the slice is less than or equal to K. More precisely it is a slice, such that max(A[P], A[P + 1], ..., A[Q]) − min(A[P], A[P + 1], ..., A[Q]) ≤ K.
The goal is to calculate the number of bounded_slices.
For example, consider K = 2 and array A such that:

```
A[0] = 3
A[1] = 5
A[2] = 7
A[3] = 6
A[4] = 3
```

There are exactly nine bounded_slices: (0, 0), (0, 1), (1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3), (4, 4).
Write a function:

```
class Solution { public int solution(int K,
int[] A); }
```

that, given an integer K and a non-empty zero-indexed array A of N integers, returns the number of bounded_slices of array A.
If the number of bounded_slices is greater than 1,000,000,000, the function should return 1,000,000,000.
For example, given:

```
A[0] = 3
A[1] = 5
A[2] = 7
A[3] = 6
A[4] = 3
```

the function should return 9, as explained above.
Assume that:

- N is an integer within the range [1..100,000];
- K is an integer within the range [0..1,000,000,000];
- each element of array A is an integer within the range [−1,000,000,000..1,000,000,000].

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Solution

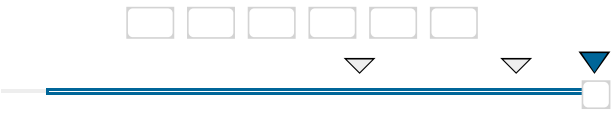
Programming language used: Java

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: correct functionality, problems with scalability

Task timeline



18:07:48 18:08:18

Code: 18:08:18 UTC, java, final, score: 60.00

Analysis

Detected time complexity:
O(N ** 3)

test	time	result
example example test	0.290 s.	**OK**
single single element	0.290 s.	**OK**
double two elements	0.300 s.	**OK**
small_functional small functional tests	0.310 s.	**OK**
small_random small random sequences length = ~100	0.300 s.	**OK**
small_random2 small random sequences length = ~100	0.300 s.	**OK**
medium_random chaotic medium		

sequences length = ~3,000	0.340 s.	**OK**
large_range large range test, length = ~100,000	2.250 s.	**TIMEOUT ERROR** running time: >2.25 sec., time limit: 0.98 sec.
large_random random large sequences length = ~100,000	4.060 s.	**TIMEOUT ERROR** running time: >4.06 sec., time limit: 1.06 sec.
large_answer test with large answer	2.820 s.	**TIMEOUT ERROR** running time: >2.82 sec., time limit: 1.02 sec.
large_extreme all maximal value = ~100,000	2.890 s.	**TIMEOUT ERROR** running time: >2.89 sec., time limit: 1.10 sec.

Get acc