Codility_

## CodeCheck Report: trainingPY2EJH-RCF

Test Name:

Check out Codility training tasks

Summary Timeline

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| CountBoundedSlices<br>Python | 7 min | 80% |

### Total score


## Tasks Details

1. CountBoundedSlices

Medium

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

| Task Score | Correctness | Performance |
|---|---|---|
| 80% | 100% | 60% |

### Task description

An integer K and a non-empty 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:

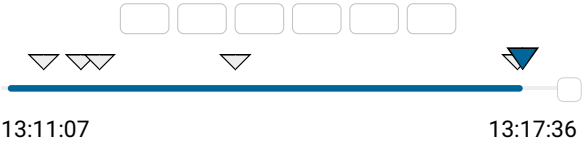
```
A[0] = 3
A[1] = 5
A[2] = 7
A[3] = 6
A[4] = 3
```

There are exactly nine bounded slices: (0, 0), (0, 1), (1, 1), (1, 2),

### Solution

| | |
|---|---|
| Programming language used: | Python |
| Total time used: | 7 minutes |
| Effective time used: | 7 minutes |
| Notes: | *not defined yet* |

### Task timeline



(1, 3), (2, 2), (2, 3), (3, 3), (4, 4).

Write a function:

```
def solution(K, A)
```

that, given an integer K and a non-empty 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.

Write an **efficient** algorithm for the following assumptions:

- 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].



Code: 13:17:35 UTC, py, final, score: **80** show code in pop-up

```python
# you can write to stdout for debugging purp
# print("this is a debug message")

def solution(K, A):
    # Implement your solution here

    def triangular(i):
        return (i * (i + 1)) // 2


    i = 0
    result = 0

    while i < len(A):
        lower = A[i]
        upper = A[i]
        countBackw = 0
        countForw = 0

        j = i - 1
        while j >= 0:
            if A[j] < lower:
                if upper - A[j] > K:
                    break
                else:
                    lower = A[j]
            elif A[j] > upper:
                if A[j] - lower > K:
                    break
                else:
                    upper = A[j]
            countBackw += 1
            j -= 1

        j = i
        while j < len(A):
            if A[j] < lower:
                if upper - A[j] > K:
                    break
                else:
                    lower = A[j]
            elif A[j] > upper:
                if A[j] - lower > K:
                    break
                else:
                    upper = A[j]
            countForw += 1
            j += 1

        result -= triangular(countBackw)
        result += triangular(countForw + cou
        i += countForw

    return result
```

## Analysis summary

The following issues have been detected: wrong answers, timeout errors.

## Analysis

Detected time complexity: **O(N)**

| expand all | **Example tests** |
|---|---|

| | | |
|---|---|---|
| ▶ | example<br>example test | ✔ OK |

expand all **Correctness tests**

| | | |
|---|---|---|
| ▶ | single<br>single element | ✔ OK |
| ▶ | double<br>two elements | ✔ OK |
| ▶ | small_functional<br>small functional tests | ✔ OK |
| ▶ | small_random<br>small random sequences length = ~100 | ✔ OK |
| ▶ | small_random2<br>small random sequences length = ~100 | ✔ OK |

expand all **Performance tests**

| | | |
|---|---|---|
| ▶ | medium_random<br>chaotic medium sequences length = ~3,000 | ✔ OK |
| ▶ | large_range<br>large range test, length = ~100,000 | ✘ TIMEOUT ERROR<br>Killed. Hard limit reached: 6.000 sec. |
| ▶ | large_random<br>random large sequences length = ~100,000 | ✔ OK |
| ▶ | large_answer<br>test with large answer | ✔ OK |
| ▶ | large_extreme<br>all maximal value = ~100,000 | ✘ WRONG ANSWER<br>got 5000050000<br>expected 1000000000 |