Codility_

CodeCheck Report: trainingRNRG7J-KH8

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task		Time spent	Score
Flags Java 8	⚠	29 min	100%

Total score

100%

Tasks Details

1. Flags

Medium

Find the maximum number of flags that can be set on mountain peaks.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given.

A *peak* is an array element which is larger than its neighbours. More precisely, it is an index P such that 0 < P < N − 1 and A[P − 1] < A[P] > A[P + 1].

For example, the following array A:

```
    A[0] = 1
    A[1] = 5
    A[2] = 3
    A[3] = 4
    A[4] = 3
    A[5] = 4
    A[6] = 1
    A[7] = 2
    A[8] = 3
    A[9] = 4
    A[10] = 6
    A[11] = 2
```

has exactly four peaks: elements 1, 3, 5 and 10.

You are going on a trip to a range of mountains whose relative heights are represented by array A, as shown in a figure below. You have to choose how many flags you should take with you.

Solution

Programming language used:	Java 8
Total time used:	29 minutes
Effective time used:	29 minutes
Notes:	*not defined yet*

Task timeline

07:19:29 07:48:11

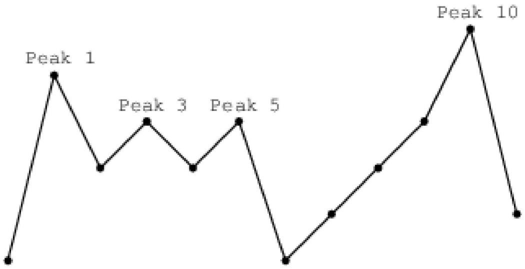
Code: 07:48:10 UTC, java, final, score: **100**

show code in pop-up

```
// you can also use imports, for example:
// import java.util.*;

```

The goal is to set the maximum number of flags on the peaks, according to certain rules.



Flags can only be set on peaks. What's more, if you take K flags, then the distance between any two flags should be greater than or equal to K. The distance between indices P and Q is the absolute value |P − Q|.

For example, given the mountain range represented by array A, above, with N = 12, if you take:

- two flags, you can set them on peaks 1 and 5;
- three flags, you can set them on peaks 1, 5 and 10;
- four flags, you can set only three flags, on peaks 1, 5 and 10.

You can therefore set a maximum of three flags in this case.

Write a function:

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

that, given a non-empty array A of N integers, returns the maximum number of flags that can be set on the peaks of the array.

For example, the following array A:

```
A[0] = 1
A[1] = 5
A[2] = 3
A[3] = 4
A[4] = 3
A[5] = 4
A[6] = 1
A[7] = 2
A[8] = 3
A[9] = 4
A[10] = 6
A[11] = 2
```

the function should return 3, as explained above.

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

- N is an integer within the range [1..400,000];
- each element of array A is an integer within the range [0..1,000,000,000].

```
// you can write to stdout for debugging purposes,
// System.out.println("this is a debug message");

import java.util.Arrays;
import java.lang.Integer;
import java.util.ArrayList;
import java.util.List;
class Solution {
    public int solution(int[] A) {
        ArrayList<Integer> array = new ArrayList<I
                for (int i = 1; i < A.length - 1;
                        if (A[i - 1] < A[i] && A[i
                                array.add(i);
                        }
                }
            if (array.size() == 1 || array.size()
                        return array.size();
            }
        int sf = 1;
        int ef = array.size();
        int result = 1;
        while (sf <= ef) {
            int flag = (sf + ef) / 2;
            boolean suc = false;
            int used = 0;
            int mark = array.get(0);
            for (int i = 0; i < array.size(); i++)
                if (array.get(i) >= mark) {
                    used++;
                    mark = array.get(i) + flag;
                                        if (used =
                                                su
                                                br
                                        }
                }
            }
            if (suc) {
                result = flag;
                sf = flag + 1;
            }else {
                ef = flag - 1;
            }
        }
       return result;
     }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all Example tests

▶ example ✓ OK
example test

expand all Correctness tests

▶ single ✓ OK
extreme min test

▶ triple ✓ OK
three elements

▶ extreme_without_peaks ✓ OK
test without peaks

▶

	simple1 first simple test	✓ OK
▶	simple2 second simple test	✓ OK
▶	medium_many_peaks medium test with 100 peaks	✓ OK
▶	medium_random chaotic medium sequences, length = ~10,000	✓ OK
▶	packed_peaks possible to set floor(sqrt(N))+1 flags	✓ OK
expand all	**Performance tests**	
▶	large_random chaotic large sequences, length = ~100,000	✓ OK
▶	large_little_peaks large test with 20-800 peaks	✓ OK
▶	large_many_peaks large test with 10,000 - 25,000 peaks	✓ OK
▶	large_anti_slow large test anti slow solutions	✓ OK
▶	large_anti_slow2 large test anti slow solutions	✓ OK
▶	extreme_max extreme test, maximal number of elements	✓ OK
▶	extreme_max2 extreme test, maximal number of elements	✓ OK