codility

Congratulations

You have completed a Codility training test.

Tweet this!
I took a programming test on @Codility!
https://codility.com/demo/take-sample-test/flags/

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingECXJ78-5MZ
Time limit: 120 min.

Status: closed

Created on: 2018-01-21 02:52 UTC
Started on: 2018-01-21 02:52 UTC
Finished on: 2018-01-21 02:53 UTC

Tasks in test	Correctness	Performance	Task score
1 Flags Submitted in: C++	75%	14%	46%

Test score

46%

46 out of 100 points

MEDIUM

1. Flags

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

score: 46 of 100

Task description

A non-empty zero-indexed 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. The goal is to set the maximum number of flags on the peaks, according to certain rules.

Solution

Programming language used: C++

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: *not defined yet*

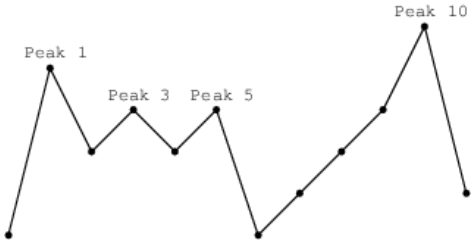
Task timeline

02:52:45 02:53:11

Code: 02:53:11 UTC, cpp, final, score: **46**

show code in pop-up

```
1  // you can use includes, for example:
2  // #include <algorithm>
3
4  // you can write to stdout for debugging purposes, e.g.
5  // cout << "this is a debug message" << endl;
6
7  int solution(vector<int> &A) {
8          int ret = 0;
9          vector<int> vPeaks;
10         int size = A.size();
11
12         //get the peak positions first, store them in a vector v
13         for (int i = 0; i < size; i++)
14         {
15                 //handle index 0
16                 if (0 == i)
```



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:

```
int solution(vector<int> &A);
```

that, given a non-empty zero-indexed 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.

Assume that:

- 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].

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).


```
17              {
18                      if (size > 1)
19                      {
20                              if (A[0] > A[1])
21                                      vPeaks.push_back(0);
22                      }
23              }//handle the last element
24              else if (i == size - 1)
25              {
26                      if (A[i] > A[i - 1])
27                              vPeaks.push_back(i);
28              }//rest of the vector
29              else
30              {
31                      if (A[i] > A[i - 1] && A[i] > A[i + 1])
32                              vPeaks.push_back(i);
33              }
34      }
35
36
37      int posMaxFlags = vPeaks.size();
38      int peakSize = vPeaks.size();
39      //int firstFlag = 0;
40      //int firstPeak = vPeaks[0];
41      int nextFlag;
42      int nextPeak;
43      //int currentFlag = firstFlag;
44      int flags;
45      for (flags = posMaxFlags; flags > 0; flags--)
46      {
47              int moves = flags -1;
48              int currentFlag = 0;
49              while (moves > 0 /*&& currentFlag < peakSize*/)
50              {
51                      nextPeak = vPeaks[currentFlag] + flags;
52                      //nextFlag = currentFlag + flags;
53                      //if (nextFlag >= peakSize)
54                      //      break;
55                      int j;
56                      for (j = currentFlag + 1; j < peakSize; j++)
57                      {
58                              if (vPeaks[j] >= nextPeak)
59                              {
60                                      currentFlag = j;
61                                      break;
62                              }
63                      }
64                      if (j >= peakSize)
65                              break;
66                      moves--;
67              }
68              if (0 == moves)
69              {
70                      ret = flags;
71                      break;
72              }
73      }
74      ret = flags;
75
76      return ret;
77  }
```

Analysis summary

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

For example, for the input `[3, 2, 1]` the solution returned a wrong answer (got 1 expected 0).

Analysis

expand all	Example tests	
▸	example example test	✓ OK
expand all	Correctness tests	
▸	single extreme min test	✓ OK
▸	triple three elements	✗ WRONG ANSWER got 1 expected 0
▸	extreme_without_peaks test without peaks	✗ WRONG ANSWER got 1 expected 0

▸ 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	✗ TIMEOUT ERROR	running time: 0.74 sec., time limit: 0.10 sec.
▸ large_little_peaks large test with 20-800 peaks	✗ WRONG ANSWER	got 21 expected 20
▸ large_many_peaks large test with 10,000 - 25,000 peaks	✗ TIMEOUT ERROR	running time: 0.42 sec., time limit: 0.10 sec.
▸ large_anti_slow large test anti slow solutions	✗ TIMEOUT ERROR	running time: >6.00 sec., time limit: 0.10 sec.
▸ large_anti_slow2 large test anti slow solutions	✗ TIMEOUT ERROR	running time: 0.59 sec., time limit: 0.10 sec.
▸ extreme_max extreme test, maximal number of elements	✗ TIMEOUT ERROR	running time: 1.05 sec., time limit: 0.10 sec.
▸ extreme_max2 extreme test, maximal number of elements	✓ OK	

Training center