Codility_

CodeCheck Report: training7HC4C5-NQG

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
StoneWall Python	1 min	100%

Total score

Tasks Details

1. StoneWall

Easy

Cover "Manhattan skyline" using the minimum number of rectangles.

Task Score	Correctness	Performance
100%	100%	100%

Task description

You are going to build a stone wall. The wall should be straight and N meters long, and its thickness should be constant; however, it should have different heights in different places. The height of the wall is specified by an array H of N positive integers. H[I] is the height of the wall from I to I+1 meters to the right of its left end. In particular, H[0] is the height of the wall's left end and H[N−1] is the height of the wall's right end.

The wall should be built of cuboid stone blocks (that is, all sides of such blocks are rectangular). Your task is to

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

compute the minimum number of blocks needed to build the wall.

Write a function:

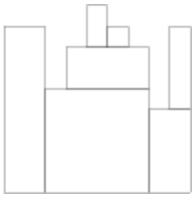
```
def solution(H)
```

that, given an array H of N positive integers specifying the height of the wall, returns the minimum number of blocks needed to build it.

For example, given array H containing N = 9 integers:

```
H[0] = 8    H[1] = 8    H[2] = 5
H[3] = 7    H[4] = 9    H[5] = 8
H[6] = 7    H[7] = 4    H[8] = 8
```

the function should return 7. The figure shows one possible arrangement of seven blocks.



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

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

Task timeline

09:09:50 09:10:32

Code: 09:10:31 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debuggin
2  # print("this is a debug message")
3
4  def solution(H):
5      # Implement your solution here
6      # pass
7      block_count = 0
8      stack = []
9
10     for h in H:
11         while stack and stack[-1] > h:
12             stack.pop()
13
14         if not stack or stack[-1] != h
15             stack.append(h)
16             block_count += 1
17
18     return block_count
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example	✔ OK
expand all	Correctness tests	
▶	simple1	✔ OK
▶	simple2	✔ OK
▶	simple3	✔ OK
▶	simple4	✔ OK
▶	boundary_cases	✔ OK
expand all	Performance tests	
▶	medium1	✔ OK
▶	medium2	✔ OK
▶	medium3	✔ OK
▶	medium4	✔ OK

▶	large_piramid	✔ OK
▶	large_increasing_decreasing	✔ OK
▶	large_up_to_20	✔ OK
▶	large_up_to_100	✔ OK
▶	large_max	✔ OK