Codility_

CodeCheck Report: trainingW2FA5U-WST

Test Name:

Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
StoneWall ⚠ Python	10 min	100%

Total score

Tasks Details

Easy

1. StoneWall

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

Write a function:

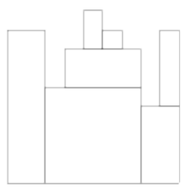
```
def solution(H)
```

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.



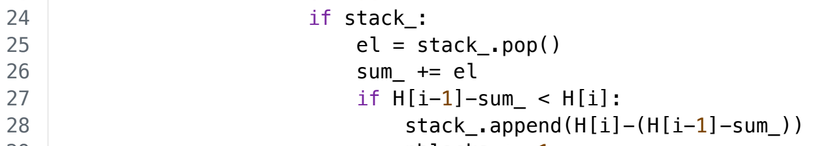
Write an **efficient** algorithm for the following assumptions:

- 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].

Solution

Programming language used:	Python
Total time used:	10 minutes
Effective time used:	10 minutes
Notes:	*not defined yet*

Task timeline



07:23:16 07:33:15

Code: 07:33:15 UTC, py, final, score: **100**

show code in pop-up

```python
# you can write to stdout for debugging purposes, e.g.
# print("this is a debug message")

def solution(H):
    # write your code in Python 3.6

    N = len(H)
    if not H:
        return None
    if min(H) < 1:
        return None
    if N < 3:
        return N

    nblocks = 1
    stack_ = []
    for i in range(len(H)):
        if i > 0:
            if H[i] == H[i-1]:
                pass
            elif H[i] < H[i-1]:
                sum_ = 0
                while True:
                    if stack_:
                        el = stack_.pop()
                        sum_ += el
                        if H[i-1]-sum_ < H[i]:
                            stack_.append(H[i]-(H[i-1]-sum_))
                            nblocks += 1
                            break
                        elif H[i-1]-sum_ == H[i]:
                            break
                        else:
                            pass
                    else:
                        break
                if not stack_:
                    stack_.append(H[i])
                    nblocks += 1
            elif H[i] > H[i-1]:
                nblocks += 1
                stack_.append(H[i]-H[i-1])
            else:
                pass
        else:
            stack_.append(H[0])

    return nblocks
```

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