# Stone wall

c0dility
WE TEST CODERS

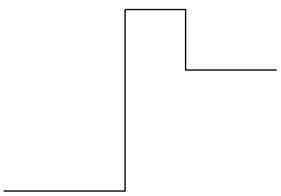
It's time to show you how the Codility challenge code-named Sigma can be solved. You can still give it a try, but no certificate will be granted.

This problem is a variant of a problem known as *rectilinear skyline problem* or *Manhattan skyline problem*, where a city skyline is given, each building contributes a rectangle to the skyline, and the question is: what is the minimum number of buildings required to produce the given skyline? The skyline corresponds to the shape of the wall and buildings correspond to stone blocks.

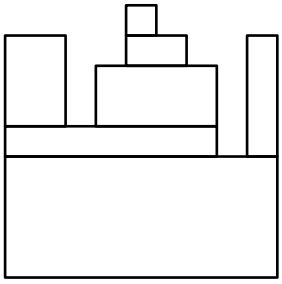
At a first glance, problems seem different. On one hand, buildings have to stand on the ground and rectangles corresponding to them can overlap. On the other hand, stone blocks cannot overlap and one block can be placed on top of another. However, as we will see, both problems can be solved using the same method.

The intuition is that by extending each stone block to the ground, we obtain a set of buildings forming the given skyline. The converse observation is less obvious.

The number of stone blocks (or buildings) is not larger than the number of horizontal edges in the skyline that are above the ground. Simply, For each such edge we can put a separate building or a stone block standing on the ground. Tu use fewer stone blocks, one stone block must be adjacent to several horizontal edges of the skyline. When is it possible? Clearly, if the two horizontal edges are adjacent to the same stone block, they must be at the same height. Moreover, the stone wall between the two edges cannot be lower than them. It turns out, that these two conditions are sufficient.



The peak shown on the above figure should be made of a single stone block. But how tall should be this stone block? It turns out, that making its lower edge level with the higher of the two neighboring horizontal edges is always a good choice. Using such a greedy strategy, any two horizontal edges of the skyline satisfying the conditions specified above are adjacent to the same stone block. Applying this greedy strategy to the example data yields the following solution:



Here is a solution implementing this greedy strategy in Python:

**1: Stone walls — $O(N)$**

```
def stone_wall(H):
    N = len(H)
    stones = 0
    stack = [0] * N
    stack_num = 0

    for i in range(N):
        while stack_num > 0 and stack[stack_num - 1] > H[i]:
            stack_num -= 1
        if stack_num > 0 and stack[stack_num - 1] == H[i]:
            pass
        else:
            stones += 1
            stack[stack_num] = H[i]
            stack_num += 1
    return stones
```

This solution scans the skyline from left to right. It keeps on a stack a sequence of horizontal edges, such that their heights form an increasing sequence. Whenever a peak is passed, it is eliminated by generating some stone blocks covering it. Each horizontal edge can be pushed onto the stack and popped from the stack once. Hence, time complexity of this solution is $O(N)$.