Codility_

CodeCheck Report: trainingP8GDGZ-67U

Test Name:

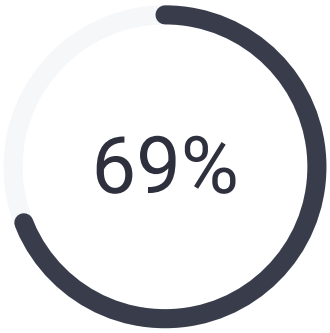
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
RectangleBuilderGreaterArea Python	4 min	69%

Total score



Tasks Details

Medium

1. RectangleBuilderGreaterArea

Count the distinct rectangle sizes, of area greater than or equal to X, that can be built out of a given set of segments.

Task Score	Correctness	Performance
69%	94%	22%

Task description

Halfling Woolly Proudhoof is an eminent sheep herder. He wants to build a pen (enclosure) for his new flock of sheep. The pen will be rectangular and built from exactly four pieces of fence (so, the pieces of fence forming the opposite sides of the pen must be of equal length). Woolly can choose these pieces out of N pieces of fence that are stored in his barn. To hold the entire flock, the area of the pen must be greater than or equal to a given threshold X.

Woolly is interested in the number of different ways in which he can build a pen. Pens are considered different if the sets of lengths of their sides are different. For example, a pen of size 1×4 is different from a pen of size 2×2 (although both have an area of 4), but pens of sizes 1×2 and 2×1 are considered the same.

Write a function:

```
def solution(A, X)
```

that, given an array A of N integers (containing the lengths of the available pieces of fence) and an integer X, returns the number of different ways of building a rectangular pen satisfying the above conditions. The function should return −1 if the result exceeds 1,000,000,000.

For example, given X = 5 and the following array A:

```
A[0] = 1
```

Solution

Programming language used:	Python
Total time used:	4 minutes
Effective time used:	4 minutes
Notes:	*not defined yet*

Task timeline

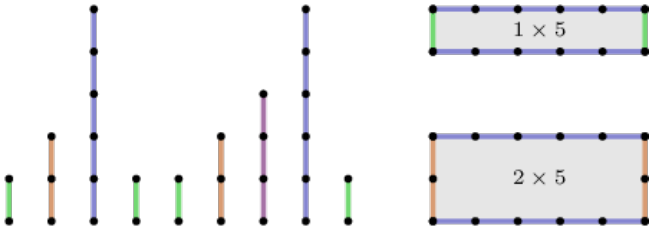
13:41:13 13:44:16

Code: 13:44:16 UTC, py, final, score: **69**

show code in pop-up

```
1  def solution(A, X):
2      # Initialize a dictionary to store the count
3      fence_count = {}
4      for fence in A:
```

```
A[1] = 2
A[2] = 5
A[3] = 1
A[4] = 1
A[5] = 2
A[6] = 3
A[7] = 5
A[8] = 1
```



the function should return 2. The figure above shows available pieces of fence (on the left) and possible to build pens (on the right). The pens are of sizes 1x5 and 2x5. Pens of sizes 1×1 and 1×2 can be built, but are too small in area. It is not possible to build pens of sizes 2×3 or 3×5, as there is only one piece of fence of length 3.

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

- N is an integer within the range [0..100,000];
- X is an integer within the range [1..1,000,000,000];
- each element of array A is an integer within the range [1..1,000,000,000].


```
5          fence_count[fence] = fence_count.get(fen
6
7      # Initialize the answer and the list of cand
8      answer = 0
9      candidate_fences = []
10
11     # For each fence length
12     for fence, count in fence_count.items():
13         # If there are more than one of this fen
14         if count > 1:
15             # Add it to the list of candidate fe
16             candidate_fences.append(fence)
17         # If there are more than three of this f
18         if count > 3 and fence * fence >= X:
19             # Increase the answer by one
20             answer += 1
21
22     # Sort the candidate fences
23     candidate_fences.sort()
24
25     # For each pair of candidate fences
26     for i in range(len(candidate_fences)):
27         for j in range(i + 1, len(candidate_fenc
28             # If a pen with these fence lengths
29             if candidate_fences[i] * candidate_f
30                 # Increase the answer by one
31                 answer += 1
32
33     # If the answer is greater than 1,000,000,00
34     if answer > 1000000000:
35         # Return -1
36         return -1
37
38     # Return the answer
39     return answer
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **O(N**2)**

expand all	Example tests	
▶	example1 example test	✔ OK
expand all	**Correctness tests**	
▶	extreme corner cases, N <= 5	✔ OK
▶	squares includes squares, N <= 20	✔ OK
▶	simple simple tests, N <= 20	✔ OK
▶	fib Fibonacci numbers, N <= 20	✔ OK
▶	small_repetitions a few repeated values, N <= 50	✔ OK
▶	small_random random values, N <= 50	✔ OK
▶	geometric_sequence powers of 2, N <= 100	✔ OK
expand all	**Correctness/performance tests**	
▶	medium_continuous continuous values, each appears 4 times, N <= 2,000	✔ OK

▶ large_repetitions many repeated values, N <= 100,000	✔ OK
▶ max continuous values, N <= 100,000	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.

expand all **Performance tests**

▶ medium_random random values, N <= 20,000	✘ TIMEOUT ERROR running time: 1.364 sec., time limit: 0.144 sec.
▶ large_continuous continuous values, each appears 4 times, N <= 80,000	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶ large_random random values, N <= 100,000	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.