Codility_

CodeCheck Report: trainingYN26JU-BSP

Test Name:

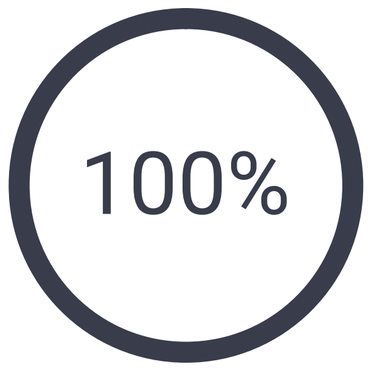
Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
MinPerimeterRectangle ⚠ Python	12 min	100%

Total score



Tasks Details

Easy

1. MinPerimeterRectangle

Find the minimal perimeter of any rectangle whose area equals N.

Task Score	Correctness	Performance
100%	100%	100%

Task description

An integer N is given, representing the area of some rectangle.

The *area* of a rectangle whose sides are of length A and B is A * B, and the *perimeter* is 2 * (A + B).

The goal is to find the minimal perimeter of any rectangle whose area equals N. The sides of this rectangle should be only integers.

For example, given integer N = 30, rectangles of area 30 are:

- (1, 30), with a perimeter of 62,
- (2, 15), with a perimeter of 34,
- (3, 10), with a perimeter of 26,
- (5, 6), with a perimeter of 22.

Write a function:

```
def solution(N)
```

that, given an integer N, returns the minimal perimeter of any rectangle whose area is exactly equal to N.

For example, given an integer N = 30, the function should return 22, as explained above.

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

- N is an integer within the range [1..1,000,000,000].

Solution

Programming language used:	Python
Total time used:	12 minutes
Effective time used:	12 minutes
Notes:	*not defined yet*

Task timeline

03:52:39 04:03:57

Code: 04:03:56 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging purposes, e.g.
2  # print("this is a debug message")
3
4  def solution(N):
5      # write your code in Python 3.6
6      if N == 0:
7          return 0
8      elif N == 1:
9          return 2*(1+1)
10     elif N == 2:
11         return 2*(1+2)
12     else:
13         if (N**0.5)%1 == 0:
14             return 4*int(N**0.5)
15         for i in range(int(N**0.5)+1, 0, -1):
16             if N%i == 0:
17                 return 2*(N//i+N//(N//i))
18         return 0
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✔ OK
expand all	**Correctness tests**	
▶	extreme_min N = 1 test	✔ OK
▶	simple1 N = 36 test	✔ OK
▶	simple2 N = 48 test	✔ OK
▶	simple3 N = 101 test	✔ OK
▶	small N = 1,234 test	✔ OK
expand all	**Performance tests**	
▶	medium N = 4,564,320 test	✔ OK
▶	prime1 N = 15,486,451 test	✔ OK
▶	square N = 100,000,000 test	✔ OK
▶	prime2 N = 982,451,653 test	✔ OK
▶	extreme_max N = 1,000,000,000 test	✔ OK