

CodeCheck Report: trainingNCXZUG-XX3

[Check out Codility training tasks](#)

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
Nesting Python	3 min	100%

Total score



Tasks Details

Easy	1. Nesting Determine whether a given string of parentheses (single type) is properly nested.	Task Score	Correctness	Performance
		100%	100%	100%

Task description

A string *S* consisting of *N* characters is called *properly nested* if:

- *S* is empty;
- *S* has the form "(*U*)" where *U* is a properly nested string;
- *S* has the form "*VW*" where *V* and *W* are properly nested strings.

For example, string "((()))" is properly nested

Solution

Programming language used:	Python
Total time used:	3 minutes ?
Effective time used:	3 minutes ?
Notes:	not defined yet

but string "())" isn't.

Write a function:

```
def solution(S)
```

that, given a string S consisting of N characters, returns 1 if string S is properly nested and 0 otherwise.

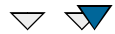
For example, given S = "(()())()", the function should return 1 and given S = "())", the function should return 0, as explained above.

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

- N is an integer within the range [0..1,000,000];
- string S is made only of the characters '(' and/or ')'.

Copyright 2009–2023 by Codility Limited. All Rights Reserved.
Unauthorized copying, publication or disclosure prohibited.

Task timeline



08:53:57

08:56:14

Code: 08:56:13 UTC, py, [show code in pop-up](#)
final, score: 100

```
1 # you can write to stdout for debuggin
2 # print("this is a debug message")
3
4 def solution(S):
5     # Implement your solution here
6     # pass
7     stack = []
8
9     for bracket in S:
10         if bracket in '(':
11             stack.append(bracket)
12
13         elif bracket in ')':
14             if not stack:
15                 return 0
16
17             top = stack.pop()
18             if (bracket == ')' and top
19                 return 0
20
21     return 1 if not stack else 0
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example1	✓ OK
	example test	
▶	example2	✓ OK
	example test2	
expand all	Correctness tests	
▶	negative_match	✓ OK
	invalid structure, but the number of parentheses matches	
▶	empty	✓ OK
	empty string	
▶	simple_grouped	✓ OK

simple grouped positive and
negative test, length=22

▶ small_random ✓ OK

expand all

Performance tests

▶ large1 ✓ OK

simple large positive and
negative test, 10K or 10K+1 ('s
followed by 10K')s

▶ large_full_ternary_tree ✓ OK

tree of the form T=(TTT) and
depth 11, length=177K+

▶ multiple_full_binary_trees ✓ OK

sequence of full trees of the
form T=(TT), depths [1..10..1],
with/without unmatched ')' at the
end, length=49K+

▶ broad_tree_with_deep_paths ✓ OK

string of the form (TTT...T) of
300 T's, each T being '((...))'
nested 200-fold, length=1 million