Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoUFHJQR-FYS
Time limit: 120 min.

Status: closed

Created on: 2014-03-15 02:44 UTC
Started on: 2014-03-15 02:44 UTC
Finished on: 2014-03-15 02:56 UTC

Congratulations ×

You have completed Codility demo.

Tweet this!
I scored 100% in #py on @Codility!
http://codility.com/demo/take-sample-test/brackets/

Sign up for our newsletter!

Like us on Facebook!

Test score

100%

out of 100 points

Tasks in test

EASY

1. Brackets

Determine whether a given string of parentheses is properly nested.

score: 100 of 100

Task description

A string S consisting of N characters is considered to be *properly nested* if any of the following conditions is true:

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

For example, the string "{[()()]}" is properly nested but "([)()]" is not.
Write a function:

```
def solution(S)
```

that, given a string S consisting of N characters, returns 1 if 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.
Assume that:

- N is an integer within the range [0..200,000];
- string S consists only of the following characters: "(", "{", "[", "]", "}" and/or ")".

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N) (not counting the storage required for input arguments).

Solution

Programming language used: Python

Total time used: 13 minutes

Effective time used: 13 minutes

Notes: correct functionality and scalability

Task timeline

02:44:39 02:56:57

Code: 02:56:57 UTC, py, final, score: **100.00**

```
01. def solution(S):
02.     # write your code in Python 2.6
03.     stack = []
04.     for s in S:
05.         if s in ('(', '{', '['):
06.             stack.append(s)
07.         elif stack and ((s == ')' and stack[-1] ==
            '(')\
08.             or (s == '}' and stack[-1] == '{')\
09.             or (s == ']' and stack[-1] == '[')):
10.             stack.pop()
11.         else:
12.             return 0
13.
14.     return 1 if not stack else 0
```

Analysis

Detected time complexity:

O(N)

test	time	result
example1 example test 1	0.050 s.	OK
example2 example test 2	0.050 s.	OK
negative_match invalid structures	0.050 s.	OK
empty empty string	0.050 s.	OK
simple_grouped simple grouped positive and negative test, length=22	0.050 s.	OK
large1 simple large positive test, 100K ('s followed by 100K)'s +)(	0.050 s.	OK
large2 simple large negative test, 10K+1 ('s followed by 10K)'s +)(+ ()	0.050 s.	OK
large_full_ternary_tree tree of the form T=(TTT) and depth 11, length=177K+	0.110 s.	OK
multiple_full_binary_trees sequence of full trees of the form T=(TT), depths [1..10..1], with/without some brackets at the end, length=49K+	0.050 s.	OK
broad_tree_with_deep_paths string of the form [TTT...T] of 300 T's, each T being '{{{...}}}' nested 200-fold, length=120K+	0.090 s.	OK

Training center