codility

Training center

ck out Codility training tasks

Congratulations

You have completed a Codility demo.

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

Sign up for our newsletter!

Like us on Facebook!

Demo ticket

Session

ID: demoQPT3U8-YAE
Time limit: 120 min.

Status: closed

Created on: 2014-08-27 02:51 UTC
Started on: 2014-08-27 02:51 UTC
Finished on: 2014-08-27 02:58 UTC

Tasks in test	Correctness	Performance	Task score
1 | {} Brackets	100%	100%	100%

Test score

100%

100 out of 100 points

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

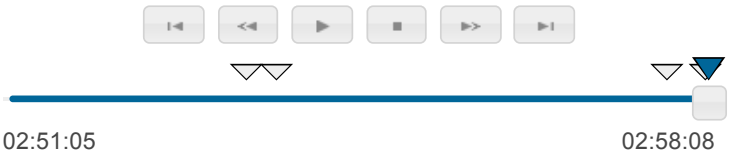
Programming language used: Python

Total time used: 8 minutes

Effective time used: 8 minutes

Notes: correct functionality and scalability

Task timeline



02:51:05 02:58:08

Code: 02:58:08 UTC, py, final, score: **100.00**

```
1  # you can use print for debugging purposes, e.g.
2  # print "this is a debug message"
3
4  def solution(S):
5      stack = []
6      matched = {}
7      matched[')'] = '('
8      matched['}'] = '{'
9      matched[']'] = '['
```

```
10
11      for v in S:
12          if v == '(' or v == '{' or v == '[':
13              stack.append(v)
14          else:
15              if stack:
16                  top = stack.pop()
17                  if matched[v] != top:
18                      return 0
19              else:
20                  return 0
21
22      if stack:
23          return 0
24
25      return 1
```

Analysis

Detected time complexity:

O(N)

test	time	result
Example tests		
example1 example test 1	0.064 s	**OK**
example2 example test 2	0.064 s	**OK**
Correctness tests		
negative_match invalid structures	0.064 s	**OK**
empty empty string	0.064 s	**OK**
simple_grouped simple grouped positive and negative test, length=22	0.064 s	**OK**
Performance tests		
large1 simple large positive test, 100K ('s followed by 100K)'s +)(	0.068 s	**OK**
large2 simple large negative test, 10K+1 ('s followed by 10K)'s +)(+ ()	0.064 s	**OK**
large_full_ternary_tree tree of the form T=(TTT) and depth 11, length=177K+	0.124 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+	0.064 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+	0.108 s	**OK**

Training center