

closed

## Demo ticket

Score:

# 100

of 100

**Session**  
**ID:** demoF9Z49T-KF5  
**Time limit:** 120 min.

**Status:** closed  
**Started on:** 2014-01-12 15:19 UTC

### ★ 1. Brackets

score: 100 of 100



Determine whether a given string of parentheses is properly nested.

#### 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:

```
class Solution { public int solution(String 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).

Copyright 2009–2014 by Codility Limited. All Rights Reserved.  
Unauthorized copying, publication or disclosure prohibited.

#### Solution

**Programming language used:** C#

**Total time used:** 1 minutes

**Effective time used:** 1 minutes

**Notes:** correct functionality and scalability

#### Task timeline



**Code:** 15:20:31 UTC, cs, final, score: 100.00

```
01. using System;
02. using System.Collections.Generic;
03.
04. class Solution {
05.     public int solution(string S)
06.     {
07.         // write your code in C# with
08.         // .NET 2.0
09.         if (S.Length > 200000) throw
10.             new
11.             ArgumentOutOfRangeException();
12.         if (String.IsNullOrEmpty(S))
13.             return 1;
14.         var stack = new Stack<char>
15.             (S.Length);
16.         foreach (var value in S)
17.         {
18.             switch (value)
19.             {
20.                 case '{':
21.                 case '[':
22.                 case '(':
23.                     stack.Push(value);
24.                     break;
25.                 case '}':
26.                     if(stack.Count == 0
27.                        || stack.Pop() !=
28.                        '{')
29.                         return 0;
30.                     break;
31.                 case ']':
32.                     if (stack.Count == 0
33.                        || stack.Pop() !=
34.                        '[')
35.                         return 0;
36.                     break;
37.                 case ')':
38.                     if (stack.Count == 0
39.                        || stack.Pop() !=
40.                        '(')
41.                         return 0;
42.                     break;
43.             }
44.         }
45.         return 1;
46.     }
47. }
```

```
29.         if (stack.Count == 0
30.             || stack.Pop() !=
31.             '(')
32.             return 0;
33.         break;
34.     default:
35.         return 0;
36.     }
37. }
38. if (stack.Count != 0)
39.     return 0;
40. return 1;
41. }
42. }
```

Analysis

Detected time complexity:

O(N)

test	time	result
example1 example test 1	0.080 s.	OK
example2 example test 2	0.080 s.	OK
negative_match invalid structures	0.080 s.	OK
empty empty string	0.080 s.	OK
simple_grouped simple grouped positive and negative test, length=22	0.080 s.	OK
large1 simple large positive test, 100K '('s followed by 100K ')'s + )	0.080 s.	OK
large2 simple large negative test, 10K+1 '('s followed by 10K ')'s + )( + )	0.080 s.	OK
large_full_ternary_tree tree of the form T=(TTT) and depth 11, length=177K+	0.090 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+	0.080 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+	0.080 s.	OK