codility

Congratulations

You have completed a Codility training test.

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

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingQHN43N-W5H
Time limit: 120 min.

Status: closed

Created on: 2016-09-12 19:42 UTC
Started on: 2016-09-12 19:42 UTC
Finished on: 2016-09-12 20:03 UTC

Tasks in test	Correctness	Performance	Task score
1 Nesting Submitted in: Java	100%	100%	100%

Test score

100%

100 out of 100 points

EASY

1. Nesting

Determine whether given string of parentheses is properly nested.

score: 100 of 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 but string "`())`" isn't.

Write a function:

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

Assume that:

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

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(1) (not counting the storage required for input arguments).

Solution

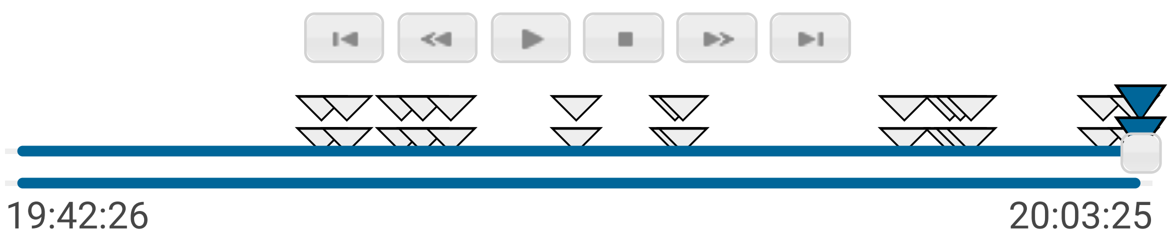
Programming language used: Java

Total time used: 21 minutes

Effective time used: 21 minutes

Notes: *not defined yet*

Task timeline



Code: 20:03:25 UTC, java, final, score: 100

show code in pop-up

```
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

import java.util.Stack;

class Solution {
    private final int fail = 0;
    private final int success = 1;

```

```java
    public int solution(String S) {
        Stack stack = new Stack();
        char[] brackets = S.toCharArray();

        for (char c : brackets) {
            if (c == '(')
                stack.push(c);
            else if (c == ')') {
                if (stack.empty())
                    return fail;
                stack.pop();
            }
        }

        return (stack.empty()) ? success : fail;
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:
O(N)

expand all	Example tests
▶ example1 example test	✔ OK
▶ example2 example test2	✔ OK

expand all	Correctness tests
▶ negative_match invalid structure, but the number of parentheses matches	✔ OK
▶ empty empty string	✔ OK
▶ simple_grouped simple grouped positive and negative test, length=22	✔ OK
▶ small_random	✔ OK

expand all	Performance tests
▶ large1 simple large positive and negative test, 10K or 10K+1 ('s followed by 10K)'s	✔ OK
▶ large_full_ternary_tree tree of the form T=(TTT) and depth 11, length=177K+	✔ OK
▶ multiple_full_binary_trees sequence of full trees of the form T=(TT), depths [1..10..1], with/without unmatched ')' at the end, length=49K+	✔ OK
▶ broad_tree_with_deep_paths string of the form (TTT...T) of 300 T's, each T being '(((...)))' nested 200-fold, length=1 million	✔ OK

Training center