Valid Parentheses

Given a string containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

The brackets must close in the correct order, `"()"` and `"()[]{}"` are all valid but `"(]"` and `"([)]"` are not.

Solution 1

```
public class Solution {
    public boolean isValid(String s) {
        Stack<Character> stack = new Stack<Character>();
        // Iterate through string until empty
        for(int i = 0; i<s.length(); i++) {
            // Push any open parentheses onto stack
            if(s.charAt(i) == '(' || s.charAt(i) == '[' || s.charAt(i) == '{')
                stack.push(s.charAt(i));
            // Check stack for corresponding closing parentheses, false if not va
lid
            else if(s.charAt(i) == ')' && !stack.empty() && stack.peek() == '(')
                stack.pop();
            else if(s.charAt(i) == ']' && !stack.empty() && stack.peek() == '[')
                stack.pop();
            else if(s.charAt(i) == '}' && !stack.empty() && stack.peek() == '{')
                stack.pop();
            else
                return false;
        }
        // return true if no open parentheses left in stack
        return stack.empty();
    }
}
```

written by KyleAsaff original link here

Solution 2

```
class Solution:
    # @return a boolean
    def isValid(self, s):
        stack = []
        dict = {"]":"[", "}":"{", ")":"("}
        for char in s:
            if char in dict.values():
                stack.append(char)
            elif char in dict.keys():
                if stack == [] or dict[char] != stack.pop():
                    return False
            else:
                return False
        return stack == []
```

It's quite obvious.

written by xiaoying10101 original link here

Solution 3

```
public class Solution {
    public boolean isValid(String s) {
        int length;

        do {
            length = s.length();
            s = s.replace("()", "").replace("{}", "").replace("[]", "");
        } while(length != s.length());

        return s.length() == 0;
    }
}
```

In this solution you essentially can remove parentheses that you know are valid until the string is empty. If the string is not empty, that means that the parentheses were malformed.

written by lough.r original link here