Untitled8

November 23, 2020

1 20. Valid Parentheses

1.1 Description

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

An input string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.

In [ ]:
```
# The solution is done via Stack

    # Solution 1

    class Solution:
        def isValid(self, s: str) -> bool:
            from queue import LifoQueue
             # Initializing a stack
            stack = LifoQueue(maxsize = 10000)
            # Create a dictionary
            dict = {
                        ")":"(",
                        "}":"{",
                        "]":"[",
                    }

            if(len(s) % 2 != 0):
                # The number of brackets must be even
                return False
            else:

                # s is not null by definition
                for i in s:
                    if(i in dict.values()):
                        # Input open bracket in a stack
                        stack.put(i)
```

```
                elif(i in dict.keys()):
                    # pop from the stack
                    if(stack.empty() == True):
                        return False
                    else:
                        item = stack.get()
                        if(item == dict.get(i)):
                            # matched correctly
                            continue
                        else:
                            # the sequence is not correct
                            return False
                else:
                    # unexpected character
                    return False
            # If we are here, than everything was correct
            if(stack.empty()):
                return True
            else:
                return False
```

```
In [ ]: # The most elegant solution
        class Solution(object):
            def isValid(self, s):
                while "()" in s or "{}" in s or '[]' in s:
                    s = s.replace("()", "").replace('{}', "").replace('[]', "")
                return s == ''
```

```
In [ ]: # The fastest solution
        class Solution(object):
            def isValid(self, s):
                if(len(s) % 2 != 0):
                    # The number of brackets must be even
                    return False
                else:
                    bracket_map = {"(": ")", "[": "]",  "{": "}"}
                    stack = []
                    for i in s:
                        if i in bracket_map.keys():
                            stack.append(i)
                        elif stack and i == bracket_map.get(stack[-1]):
                                stack.pop()
                        else:
                            return False
                    return stack == []
```

1.2 Resume

1. Simple array can be efficiently used to reproduce stack.

2. Always check border cases (like len(s) % 2 != 0) this significantly reduces the time.
3. The elegant solution efficiently uses strings functions, worth to think about an object available functions first