[Click on the logo to find **Problem Statement**]

Intuition

The given code utilizes a stack to determine the validity of a string containing various types of brackets (parentheses, square brackets, and curly braces). The algorithm iterates through each character in the input string. For every character encountered, it checks if the current character and the top of the stack form a valid pair of opening and closing brackets. If they do, it removes the top element from the stack, effectively pairing and canceling out the brackets. If not, the character is added to the stack.

The validity of the string is determined by whether the stack is empty at the end of the iteration. If it is, then all brackets found valid pairs and were canceled out; thus, the string is considered valid. Conversely, if the stack is not empty, there are unmatched brackets, making the string invalid.

Approach

1. **Initialization:**
    - Initialize a stack to keep track of the opening brackets encountered.
2. **Iteration through the String:**
    - Iterate through each character in the input string.
    - For each character:
        - Check if the stack is not empty and the current character, along with the top element of the stack, forms a valid pair of opening and closing brackets.
        - If a valid pair is found, pop the top element from the stack to cancel out the pair.
        - If not, push the current character onto the stack, indicating an unmatched opening bracket.
3. **Validation:**
    - After iterating through the entire string, check if the stack is empty.
    - If the stack is empty, all opening brackets found valid closing brackets, and the string is considered valid.
    - If the stack is not empty, there are unmatched opening brackets, making the string invalid.

4. **Result:**
    - Return `True` if the stack is empty (valid string) and `False` otherwise.

Complexity

- Time complexity: O(n)
- Space complexity: O(n)

Code

```python
class Solution:
    def isValid(self, s: str) -> bool:
        valid_brack = [('{', '}'), ('(', ')'), ('[', ']')]
        stack = []
        for i in s:
            if len(stack) != 0 and (stack[-1], i) in valid_brack:
                stack.pop()
            else:
                stack.append(i)

        return len(stack) == 0
```

If you want to see more solutions to coding problems, you can visit:

GitHub