20. Valid Parentheses

Created	@July 16, 2020 12:26 AM
Difficulty	Easy
LC Url	
Importance	****
Tag	Stack
Video	

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

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

Example 1:

```
Input: s = "()"
Output: true
```

Example 2:

```
Input: s = "()[]{}"
Output: true
```

Example 3:

```
Input: s = "(]"
Output: false
```

Constraints:

- `1 <= s.length <= 10` `4`
- `s` consists of parentheses only `'()[]{}'`.

Solution

```
class Solution:
    def isValid(self, s: str) -> bool:
        Map = {")": "(", "]": "[", "}": "{"}
        stack = []

        for c in s:
            if c not in Map:
                stack.append(c)
                continue
            if not stack or stack[-1] != Map[c]:
                return False
            stack.pop()

        return not stack
```

```
class Solution:
    def isValid(self, s: str) -> bool:
        stack = []
        for i in range(len(s)):
            char_i = s[i]
            if char_i in ['(', '[', '{']:
                stack.append(char_i)
            elif char_i in [')', ']', '}']:
                if not stack:
                    return False

                cur = stack.pop()
                if (cur == '(' and char_i != ')') \
                        or (cur == '[' and char_i != ']') \
                        or (cur == '}' and char_i != '}'):
                    return False
        if not stack:
            return True
        return False
```

```
class Solution {
    public boolean isValid(String s) {
        Stack<Character> mark = new Stack<Character>();
        for (int i = 0; i < s.length(); i++) {
            char char_i = s.charAt(i);
            if (char_i == '(' || char_i == '[' || char_i == '{') {
                mark.push(char_i);
            } else if (char_i == ')' || char_i == ']' || char_i == '}'){
                if (mark.isEmpty()) return false;
                char cur = mark.pop();

                if (cur == '(' && char_i != ')') return false;
                if (cur == '[' && char_i != ']') return false;
                if (cur == '{' && char_i != '}') return false;
            }
        }
        if (mark.isEmpty()) return true;
        return false;
    }
}
```