Valid parentheses

20. Valid Parentheses

Easy 20546 1279 Add to List Share

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
```

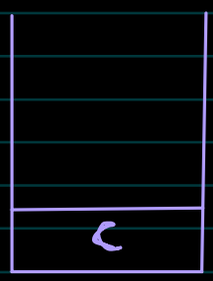
Approach :-

It's very simple we will be using stack which will store opening Brackets.

s = "( ) [ ]"

ch = s[0]

If ch holds opening bracket we will push it to stack

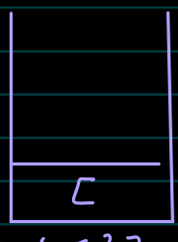


ch = s[1]

Now ch holds closing bracket then compare it with stack.top() if they are correct order simply pop stack and move to next element.

ch = s[2]

ch = s[3] if (ch == ']' ||



stack.top() == '[') → pop()

else return false;

After loop ends if our stack remains empty return True else false

loop(i → 0 → s.size()){

ch = s[i]

if (opening bracket) push();

else if (check if in correct order)
pop()

else False

}

if (empty()) True

else False.

```
class Solution {
public:
    bool isValid(string s) {
        stack<char> stk;
        for(auto ch: s){
            if(ch=='(' || ch=='[' || ch=='{') stk.push(ch);
            else{
                if(!stk.empty()){
                    if( (ch==')' && stk.top()=='(') || (ch==']' && stk.top()=='[') || (ch=='}' && stk.top()=='{') )
                        stk.pop();//Correct Order
                    }
                    else{
                        return false;
                    }
                }
                else return false;
            }
        }
        if(stk.empty()) return true;
        else return false;
    }
};
```

T.C - O(n) n: length of s

S.C - O(n)