<https://leetcode.com/problems/valid-parentheses>

**Valid Parentheses**

**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.**

**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 <= 104

s consists of parentheses only '()[]{}'.

**Method 1: (Brute Force)**

For

Time Complexity: O()

Space Complexity: O()

**Method 2:**

Use map with <closing brackets, opening brackets> as key, value pair;

helps to obtain corresponding opening bracket whenever a closing bracket is encountered.

For each character in the string :

--->if stack is empty push()

--->if char is opening bracket push()

--->if char is closing bracket, check if TOS(top of the stack) contains corresponding opening bracket and pop it.

Empty stack implies valid string.

Time Complexity: O(n) *[]*

Space Complexity: O(n) *[]*

 bool isValid(string s) {

        stack<int> m;

        map<char, char> br;

        br[')'] = '(';

        br['}'] = '{';

        br[']'] = '[';

        for(char c:s){

            if(!m.empty() && br.count(c)){

                if(br[c]==m.top()) m.pop();

                else return false;

            }

            else m.push(c);

        }

        if(m.empty())

            return true;

        else return false;

    }