1. 题目
2. 题干

给定一个只包括 '('，')'，'{'，'}'，'['，']' 的字符串，判断字符串是否有效。

有效字符串需满足：

左括号必须用相同类型的右括号闭合。

左括号必须以正确的顺序闭合。

注意空字符串可被认为是有效字符串。

1. 示例

示例 1:

输入: "()"

输出: true

示例 2:

输入: "()[]{}"

输出: true

示例 3:

输入: "(]"

输出: false

示例 4:

输入: "([)]"

输出: false

示例 5:

输入: "{[]}"

输出: true

1. 题解
2. 思路

这是一个判断括号有效性的题目，每一对括号必须是对应完整的，且每个左括号的后面必须跟着对应的右括号或者是另一对左右括号；如果后面跟的是另一对括号的右括号，那么括号就失去了有效性。

因此，我们采用栈的思想，循环字符串，如果碰到右括号，那么判断栈的头部是否是对应的左括号，或者栈是否为空？如果栈为空，或者栈的头部不是对应的左括号，那么字符串是错误的，返回false。否则，弹出左括号。注意，右括号是不入栈的，它只是用来判断。

如果循环碰到左括号，那么将左括号入栈。

最后，循环结束，如果栈是空的，那么说明左右括号都是配对的，字符串中的左括号都已经全部出栈了，字符串是正确的；否则，字符串是错误的。

1. 代码实现
2. C++

#include<string>

#include<unordered\_map>

#include<stack>

using namespace std;

class Solution {

public:

bool isValid(string s) {

int n = s.size();

//如果s的长度为奇数，则直接返回false

if (n % 2 == 1)

return false;

//设置左右括号的映射表，右括号为键，左括号为值，因为是通过右括号来找左括号

unordered\_map<char, char> pairs = {

{')', '('},

{']', '['},

{'}', '{'}

};

//栈

stack<char> stk;

//循环字符串

for (char ch : s) {

//找到了右括号

if (pairs.count(ch)) {

//如果栈为空或者栈顶不是对应的左括号

if (stk.empty() || stk.top() != pairs[ch]) {

return false;

}

//否则，配对成功，弹出栈顶

stk.pop();

}

//循环到了左括号

else {

//左括号入栈

stk.push(ch);

}

}

//循环结束，判断此时的栈是否非空，若为空则字符串正确

return stk.empty();

}

};

1. Java

import java.util.Deque;  
import java.util.HashMap;  
import java.util.LinkedList;  
import java.util.Map;  
  
import static javax.swing.UIManager.*put*;  
  
public class Valid20 {  
 public boolean isValid(String s){  
 int n = s.length();  
 if(n % 2 == 1)  
 return false;  
  
 Map<Character, Character> pairs = new HashMap<>() {{  
 put(')', '(');  
 put(']', '[');  
 put('}', '{');  
 }};  
  
 Deque<Character> stack = new LinkedList<Character>();  
 for(int i = 0; i < n; ++i){  
 char ch = s.charAt(i);  
 if(pairs.containsKey(ch)){  
 if(stack.isEmpty() || stack.peek() != pairs.get(ch))  
 return false;  
 stack.pop();  
 }  
 else{  
 stack.push(ch);  
 }  
 }  
 return stack.isEmpty();  
 }  
}