有效的括号isValid

利用堆栈stack与hashmap进行判断，思路如下：

1.判断是否为空串。
2.判断s.length是否是偶数（奇数肯定错误）
3.判断括号的顺序，
（1）首先把括号放到HashMap中，key放左括号，value放右括号；
（2）遍历字符串s，把左括号push到堆栈，右括号与最接近的左括号（栈顶的元素）匹配，成功继续循环，并pop栈顶左括号，失败则false；
（3）注意堆栈中左括号的数量。

代码如下：

```
public boolean isValid(String s) {
    if(s.length()==0)return true;
    if(s.length() % 2 != 0)return false;
    Stack<Character> stack=new Stack<Character>();
    HashMap<Character,Character> map=new HashMap<Character,Character>();
    map.put('(', ')');
    map.put('[', ']');
    map.put('{', '}');
    for(Character c:s.toCharArray()) {
        if(map.containsKey(c)) { //左括号压入堆栈
            stack.push(c);
        }else {
            if(stack.isEmpty())return false; //右括号匹配
            char current=stack.pop();
            if(map.get(current) != c)return false;
        }
    }

    return stack.isEmpty();
}
```

奇偶链表

给定一个单链表，把所有的奇数节点和偶数节点分别排在一起。请注意，这里的奇数节点和偶数节点指的是节点编号的奇偶性，而不是节点的值的奇偶性。

请尝试使用原地算法完成。你的算法的空间复杂度应为 O(1)，时间复杂度应为 O(nodes)，nodes 为节点总数。

```
class Solution {
public:
    ListNode* oddEvenList(ListNode* head) {
        if(head == nullptr || head->next == nullptr)
            return head;
        ListNode* oddhead = head;//奇链表
        ListNode* evenhead = head->next;//偶链表
        ListNode* pHead = evenhead;//偶链表头部，便于等会的指向
        while(oddhead->next != nullptr && evenhead->next != nullptr)//有一个为空则终止循环
        {
```

```
            oddhead->next = oddhead->next->next;//奇链表往后走
            oddhead = oddhead->next;//改变位置
            evenhead->next = evenhead->next->next;偶奇链表往后走
            evenhead = evenhead->next;//改变位置
        }
        oddhead->next = pHead;//奇链表尾部指向偶链表头部
        return head;//返回头
    }
};
```

接雨水

方法 1：暴力

```
int trap(vector<int>& height)
{
    int ans = 0;
    int size = height.size();
    for (int i = 1; i < size - 1; i++) {
        int max_left = 0, max_right = 0;
        for (int j = i; j >= 0; j--) { //Search the left part for max bar size
            max_left = max(max_left, height[j]);
        }
        for (int j = i; j < size; j++) { //Search the right part for max bar
size
            max_right = max(max_right, height[j]);
        }
        ans += min(max_left, max_right) - height[i];
    }
    return ans;
}
```