

题目：

题目描述

通过

笔记

题解

提交记录

42. 接雨水

困难 相关企业 题解

给定 n 个非负整数表示每个宽度为 1 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。

示例 1:



输入: height = [0,1,0,2,1,0,1,3,2,1,2,1]
输出: 6
解释: 上面是由数组 [0,1,0,2,1,0,1,3,2,1,2,1] 表示的高度图，在这种情况下，可以接 6 个单位的雨水（蓝色部分表示雨水）。

示例 2:

输入: height = [4,2,0,3,2,5]
输出: 9

提示:

5.2K 2.6K

</> 代码

C++

```
1 class Solution {
2 public:
3     int trap(vector<int>& height) {
4         if (height.empty()) return 0;
5         int n = height.size();
6         int leftMax = 0, rightMax = 0;
7         int left = 0, right = n - 1;
8         int ans = 0;
9         while (left < right) {
10             if (height[left] < height[right]) {
11                 if (height[left] > leftMax) leftMax = height[left];
12                 else ans += leftMax - height[left];
13             } else {
14                 if (height[right] > rightMax) rightMax = height[right];
15                 else ans += rightMax - height[right];
16             }
17             if (left < right) left++;
18             if (left < right) right--;
19         }
20         return ans;
21     }
22 }
```

已存储

测试用例

通过 执行

Case 1

输入

height = [0,1,0,2]

输出

6

预期结果

解题思路一：

联系之前做的题，有一道为求当前数组某个数左边右边的最大值，此题可以借鉴其思路，即我们也分为左右部分。观察数据我们发现，雨水的数量与最大值有关，只要当前值小于最大值时，当前下标与旁边下标的雨水积累一定是差值。即我们设定数组从左边和右边分别开始，依次设定左边最大值和右边最大值，当左值大时，右值小于左值，对于右值而言，左值是一个暂时的最大值，则右值只要小于右边的最大值一定有雨水积累（右值的最大值作为边界，即最大值右边的值要么为空，要么已经计算完毕了，左值类似）

换句话说，左值最大值每次计算最大值右边的小值积累的雨水，右值最大值每次计算左边的小值积累的雨水，当左值等于右值时，说明所有元素都已经遍历过依次，则当前雨水记录完毕

```
class Solution {
public:
    int trap(vector<int>& height) {
```

```

    if (height.empty()) return 0;
    //把数组拆开成为左右两部分
    int left = 0, right = height.size() - 1;
    int left_max = 0, right_max = 0;
    int water = 0;
    //相遇时说明遍历完整个数组了
    while (left < right) {
        //谁的MAX最大决定从另外一边开始计算，因为是从次高值开始写入的
        if (height[left] < height[right]) {
            //更新max
            if (height[left] >= left_max) {
                //左最值左一定计算完毕
                left_max = height[left];
            }
            else {
                water += left_max - height[left];
            }
            ++left;
        }
        else {
            if (height[right] >= right_max) {
                //右最值右一定计算完毕
                right_max = height[right];
            }
            else {
                water += right_max - height[right];
            }
            --right;
        }
    }

    return water;
}
};

```

解题思路二：

利用单调栈来解决这个问题的核心思路是，使用栈来存储柱子的索引，从而能方便地找到左右边界的高度。具体步骤如下：

1. 初始化一个空栈，`stack`，以及一个变量 `water` 来存储总接水量。
2. 遍历数组 `height`对每一个元素 `height[i]`：
 1. 当栈不为空且当前高度大于栈顶柱子的高度时：
 - 弹出栈顶元素，记为 `top`。
 - 如果栈为空，则跳出循环。
 - 计算左边界柱子的高度 `left_bound` 为当前栈顶元素的高度。
 - 计算当前柱子和左边界柱子之间的宽度 `distance`。
 - 计算可以接的水的高度 `h` 为左右边界柱子高度的较小值减去 `top` 的高度。
 - 更新总接水量 `water`。
 2. 当前索引 `i` 入栈。
3. 返回总接水量 `water`

```
class Solution {
public:
    int trap(vector<int>& height)
    {
        int ans = 0;
        //单调递减栈
        stack<int> st;
        for (int i = 0; i < height.size(); i++)
        {
            //当前栈非空且栈顶元素小于数组元素
            while (!st.empty() && height[st.top()] < height[i])
            {
                int cur = st.top();
                st.pop();
                //栈空则跳出
                if (st.empty()) break;
                int l = st.top();
                int r = i; //当前元素
                int h = min(height[r], height[l]) - height[cur];
                ans += (r - l - 1) * h;
            }
            //入栈元素一定大于等于原栈顶
            st.push(i);
        }
        return ans;
    }
};
```