leetcode42.接雨水——困难

笔记本:	leetcode刷题		
创建时间:	2021/8/12 23:54	**更新时间:**	2021/8/17 23:52
作者:	Zard		

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。

知识点——双指针

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

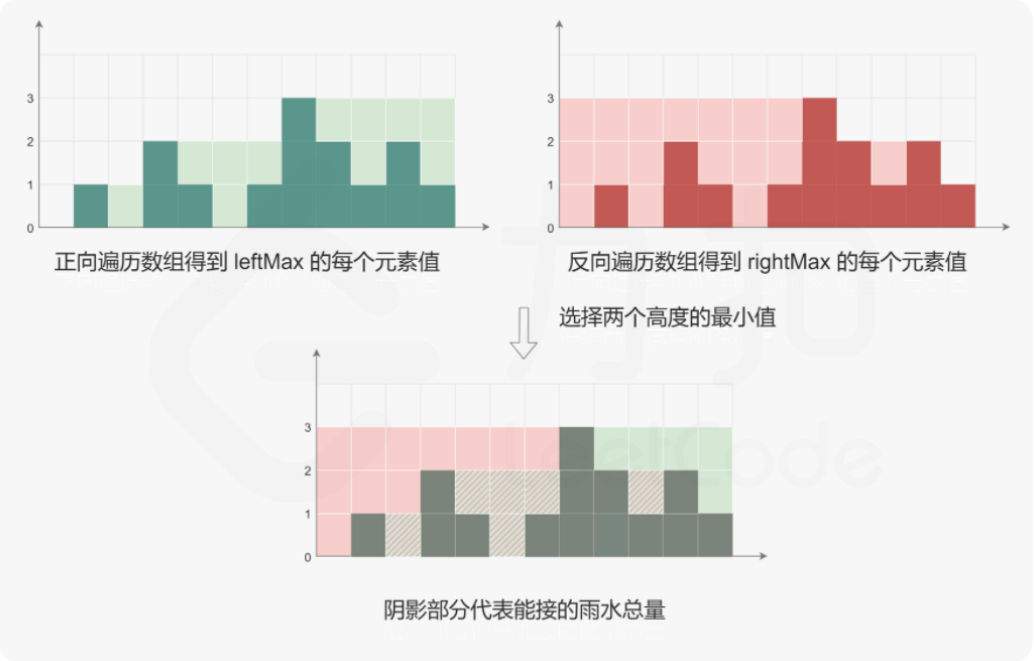
提示：
n == height.length
0 <= n <= 3 * 10.^4
0 <= height[i] <= 10.^5

方法一：动态规划
对于下标i，下雨后水能到达的最大高度等于下标i两边的最大高度的最小值，下标i处能接的雨水量等于下标i处的水能到达的最大高度减去height[i].
朴素的做法是对于数组height中的每个元素，分别向左和向右扫描并记录左边和右边的最大高度，然后计算每个下标位置能接的雨水量。假设数组height的长度为n，该做法需要对每个下标位置使用O(n)的时间向两边扫描并得到最大高度，因此总时间复杂度是O(n.^2);
使用动态规划的方法，可以在O(n)的时间内预处理得到每个位置两边的额最大高度。
创建两个长度为n的数组leftMax和rightMax。对于0<=i<n,leftMax[i]表示下标i及其左边的位置中，height的最大高度，rightMax[i]表示下标i及其右边的位置中，height的最大高度。显然，leftMax[0]=height[0],rightMax[n-1]=height[n-1]。两个数组的其余元素的计算如下：

- 当1<=i<=n-1时，leftMax[i]=max(leftMax[i-1],height[i]);
- 当0<=i<=n-2时，rightMax[i]=max(rightMax[i+1],height[i])。

因此可以正向遍历数组height得到数组leftMax的每个元素值，反向遍历数组height得到数组rightMax的每个元素值。

在得到数组leftMax和rightMax的每个元素值之后，对于0<=i<n,下标i处能接的雨水量等于min(leftMax[i],rightMax[i])-height[i]。遍历每个下标位置即可得到能接的雨水总量。




```
class Solution {
public:
    int trap(vector<int>& height) {
        int n=height.size();
        if(n==0)
            return 0;
        vector<int> leftMax(n);
        leftMax[0]=height[0];
        for(int i=1;i<n;i++)
            leftMax[i]=max(leftMax[i-1],height[i]);
        vector<int> rightMax(n);
        rightMax[n-1]=height[n-1];
        for(int i=n-2;i>=0;i--)
            rightMax[i]=max(rightMax[i+1],height[i]);
        int res=0;
        for(int i=0;i<n;i++)
            res+=min(leftMax[i],rightMax[i])-height[i];
        return res;
    }
};
```

方法三：双指针

动态规划的做法中，需要维护两个数组leftMax和rightMax，因此空间复杂度是O(n)。是否可以将空间复杂度降到O(1)?

注意到下标i处能接的雨水量由leftMax和rightMax，初始时left=0，right=n-1,leftMax=0,rightMax=0。指针left只会向右移动，指针right只会向左移动，在移动指针的过程中维护两个变量leftMax和rightMax的值。

当两个指针没有相遇时，进行如下操作：

- 使用height[left]和height[right]的值更新leftMax和rightMax的值；
- 如果height[left]<height[right],则必有leftMax<rightMax,下标left处能接的雨水量等于leftMax-height[left],将下标left处能接的雨水量加到能接的雨水总量，然后left加1（即向右移动一位）；
- 如果height[left]>=height[right],则必有leftMax>=rightMax,下标right处能接的雨水量等于rightMax-height[right],将下标right处能接的雨水量加到能接的雨水总量，然后将right减1（即向左移动一位）。

当两个指针相遇时，即可得到能接的雨水总量。

下面用一个例子height=[0,1,0,2,1,0,1,3,2,1,2,1]来帮助理解双指针的做法。

```
class Solution {
```

```
public:
    int trap(vector<int>& height) {
        int res=0;
        int left=0,right=height.size()-1;
        int leftMax=0,rightMax=0;
        while(left<right){
            leftMax=max(leftMax,height[left]);
            rightMax=max(rightMax,height[right]);
            if(height[left]<height[right]){
                res+=leftMax-height[left];
                left++;
            }else{
                res+=rightMax-height[right];
                right--;
            }
        }
        return res;
    }
};
```