42-接雨水

题述

42. 接雨水

难度 困难 3312

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：

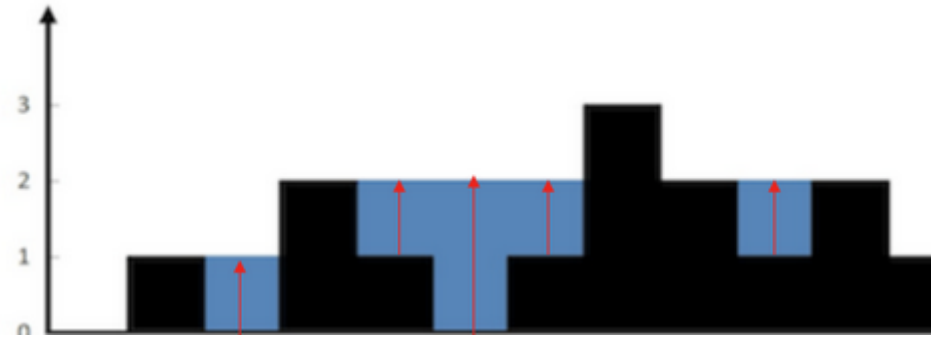
```
输入：height = [4,2,0,3,2,5]
输出：9
```

思路

双指针

先明确是按照行计算还是按照列来计算

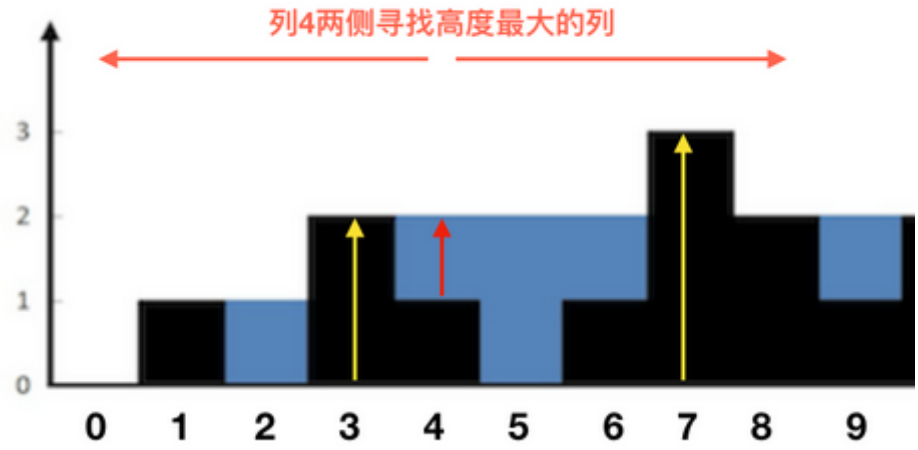
推荐按照列来计算



如果按照列来计算的话，宽度一定是1了，再把每一列的雨水的高度求出来就可以了。

每一列雨水的高度，取决于，该列 左侧最高的柱子和右侧最高的柱子中最矮的那个柱子的高度。

可能不理解，这样想：举例：求列4的雨水高度



列4 左侧最高的柱子是列3，高度为2（以下用lHeight表示）。

列4 右侧最高的柱子是列7，高度为3（以下用rHeight表示）。

列4 柱子的高度为1（以下用height表示）

那么列4的雨水高度为 列3和列7的高度最小值减列4高度，即： min(lHeight, rHeight) - height。

一样的方法，只要从头遍历一遍所有的列，然后求出每一列雨水的体积，相加之后就是总雨水的体积了。

动态规划

我们前面推导出一个公式：每一列雨水的高度，取决于，该列 左侧最高的柱子和右侧最高的柱子中最矮的那个柱子的高度。

每一列雨水的高度 = 该列 左侧最高的柱子和右侧最高的柱子中最矮的那个柱子的高度 - 当前列的高度

当前列雨水面积：min(左边柱子的最高高度，记录右边柱子的最高高度) - 当前柱子高度。

把每一个位置的左边最高高度记录在一个数组上（maxLeft），右边最高高度记录在一个数组上（maxRight）。这样就避免了重复计算，这就用到了动态规划。

递推公式：

即从左向右遍历：maxLeft[i] = max(height[i], maxLeft[i - 1]);

从右向左遍历：maxRight[i] = max(height[i], maxRight[i + 1]);

题解

Python

双指针

```python
class Solution:
    def trap(self, height: List[int]) -> int:
        # 双指针解法
        result = 0
        for i in range(len(height)):
            if i == 0 or i == len(height)-1:
                continue
            lHeight = height[i-1]
            rHeight = height[i+1]
            for j in range(i-1):
                if height[j] > lHeight:
                    lHeight = height[j]

            for k in range(i+2,len(height)):
                if height[k] > rHeight:
                    rHeight = height[k]
            rest = min(lHeight,rHeight) - height[i]
            if rest > 0:
                result += rest
        return result
```

动态规划

```python
class Solution:
    def trap(self, height: List[int]) -> int:
        # 动态规划
        # 使用两个数组来记录左右的最大高度

        # 初始化 dp teble
        leftHeight = [0] * len(height)
        rightHegiht = [0] * len(height)

        leftHeight[0] = height[0]
        for i in range(1,len(height)):
            leftHeight[i] = max(leftHeight[i-1],height[i])

        rightHegiht[-1] = height[-1]
        for i in range(len(height) - 2 , -1, -1):
            rightHegiht[i] = max(rightHegiht[i+1] , height[i])

        result = 0
        for i in range(0,len(height)):
            sumH = min(leftHeight[i],rightHegiht[i]) - height[i]
            result += sumH
        return result
```

思考