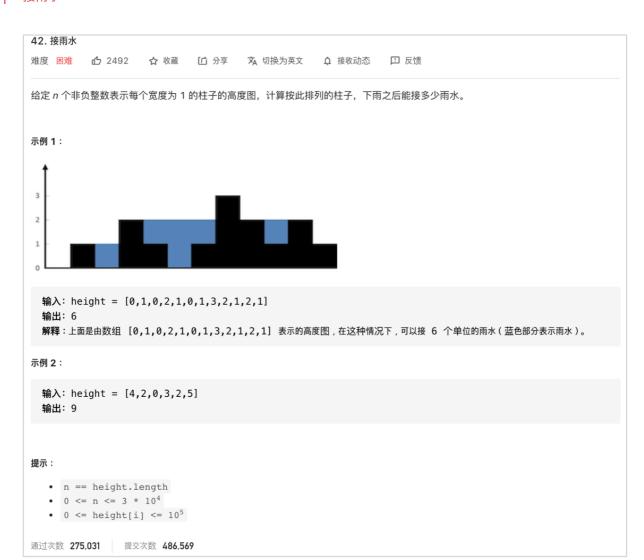
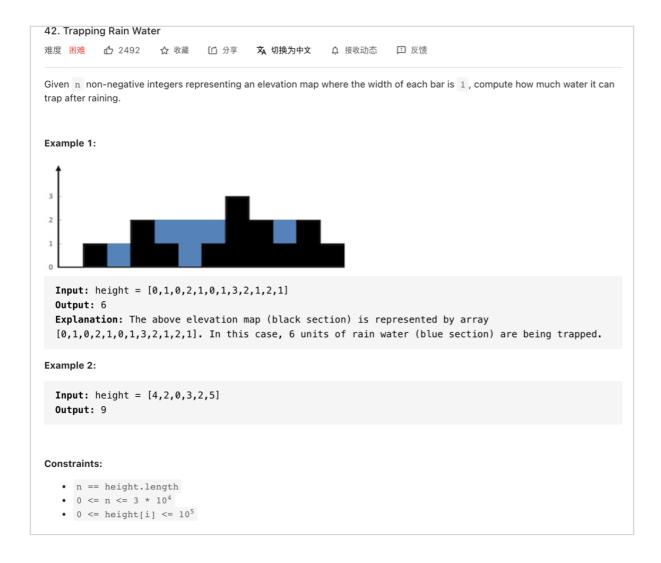
# 42.接雨水 (重要)

## 接雨水





## 思路1: 栈

- 1. 类似于括号匹配,蓝色雨水的部分,每次匹配出一对括号(这里是墙),就计算这两堵墙中的水;
- 2. 用栈保存每堵墙;
- 3. 当前高度小于栈顶高度,说明这里有会有积水,将墙的高度下标入栈,指针后移;
- 4. 当前高度大于栈顶高度,说明之前的积水到这里停下,出栈,计算当前墙和栈顶墙之间水的 多少;
- 5. 计算当前的高度和新栈的高度关系,重复第3步,直到当前墙的高度不大于栈顶高度或者栈空,然后把当前墙入栈,指针后移。

#### 代码

```
// Time : 2021 - 07 - 20
public int trap(int[] height) {
       int sum = 0;
       Stack<Integer> stack = new Stack<>();
       int current = 0;
       while(current < height.length){</pre>
           // 如果栈不空并且当前指向的高度大于栈顶高度就一直循环
           while(!stack.empty()&&height[current]>height[stack.peek()]){
               int h = height[stack.peek()]; // 取出要出栈道的元素
              stack.pop(); // 出栈道
              if(stack.empty()) // 栈空就出去
                  break;
              int distance = current - stack.peek() - 1; // 两堵墙之前的距离
              int min = Math.min(height[stack.peek()],height[current]);
              sum = sum + distance * (min - h);
           }
           stack.push(current); // 当前指向的墙入栈
           current++; // 指针后移
       }
       return sum;
   }
```

### 复杂度分析

- 时间复杂度 O(n): 虽然 while 循环里嵌套了一个 while 循环,但是考虑到每个元素最多访问两次,入栈一次和出栈一次,所以时间复杂度为 O(n).
- 空间复杂度 O(n): 需要给栈分配 n 个空间.

#Leetcode/Stack