本日学习内容

1. 二叉树基础知识
2. leetcode hot 100补全哈希，双指针，滑动窗口部分

题目一：128. 最长连续序列

给定一个未排序的整数数组 `nums` ，找出数字连续的最长序列（不要求序列元素在原数组中连续）的长度。

请你设计并实现时间复杂度为 `O(n)` 的算法解决此问题。

示例 1：

```
输入：nums = [100,4,200,1,3,2]
输出：4
解释：最长数字连续序列是 [1, 2, 3, 4]。它的长度为 4。
```

示例 2：

```
输入：nums = [0,3,7,2,5,8,4,6,0,1]
输出：9
```

示例 3：

```
输入：nums = [1,0,1,2]
输出：3
```

提示：

- `0 <= nums.length <= 105`
- `-109 <= nums[i] <= 109`

思路

1. **哈希集合去重**：使用哈希集合存储所有数字，去除重复值。
2. **查找序列起点**：遍历集合中的每个数字，如果当前数字的前一个数字不在集合中，说明当前数字可能是一个连续序列的起点。
3. **扩展序列**：从起点开始，依次检查下一个连续数字是否在集合中，统计当前序列长度。
4. **更新最大值**：每次找到一个序列后，更新全局最长序列长度。

代码

```cpp
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
        unordered_set<int> us;
        for (const int& num : nums) {
            us.insert(num);
        }
        int longestStreak = 0;
        for (const int& num : us) {
            if (!us.count(num - 1)) {
                int currentNum = num;
                int currentStreak = 1;
                while (us.count(currentNum + 1)) {
                    currentNum += 1;
                    currentStreak += 1;
                }
                longestStreak = max(longestStreak, currentStreak);
            }
        }
        return longestStreak;
    }
};
```

题目二：42. 接雨水

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：

- `n == height.length`
- `1 <= n <= 2 * 104`
- `0 <= height[i] <= 105`

思路：

使用栈来跟踪可能形成低洼的柱子。当遇到比栈顶高的柱子时，计算栈顶位置的雨水量。

遍历处理每个柱子

当前柱子高度 > 栈顶柱子高度时：

栈顶元素作为**凹槽底部** 新栈顶元素即为左边界 当前柱子为右边界 计算当前凹槽区域的雨水量

代码：

```cpp
class Solution {
public:
    int trap(vector<int>& height) {
        stack<int> st;
        int n = height.size();
        int total = 0;

        for (int i = 0; i < n; i++) {
            while (!st.empty() && height[i] > height[st.top()]) {
                int bottom = st.top();
                st.pop();
                if (st.empty()) {
                    break;
                }
                int l = st.top();
                int width = i - l - 1;
                int waterHeight = min(height[i], height[l]) - height[bottom];
                total += width * waterHeight;
            }
            st.push(i);
        }
        return total;
    }
};
```

本日遇到的问题

1. 刚接触单调栈还是有些不理解
2. 当遇到二分法的特征（数量级大，有序等）时不能及时想到用二分优化算法

明日学习内容

1. 开始学习二叉树的遍历
2. 继续leetcode hot 100