我的思路，先计算左半边，然后再计算右半边

```cpp
class Solution {
public:
    int trap(vector<int>& height) {
        int N = height.size();
        if (N == 0) return 0;

        int water = 0;
        int sum = 0;

        // 使用 std::max_element 找到最大值的迭代器
        auto maxIt = max_element(height.begin(), height.end());
        int maxValue = *maxIt;
        int maxIndex = distance(height.begin(), maxIt);

        // 计算左侧区域
        int i = 0;
        while (i < maxIndex) {
            int step = 0;
            while (i + step + 1 < maxIndex && height[i] ≤ height[i + step + 1]) {
                step++;
            }
            sum += step * height[i];
            i += step;
        }

        // 计算右侧区域
        int j = N - 1;
        while (j > maxIndex) {
            int step = 0;
            while (j - step - 1 > maxIndex && height[j] ≤ height[j - step - 1]) {
                step++;
            }
            sum += step * height[j];
            j -= step;
        }

        // 计算总高度
        int sum_height = accumulate(height.begin(), height.end(), 0);
```

```
39
40          // 计算接水量
41          water = sum - (sum_height - maxValue);
42          return water;
43      }
44  };
```

使用动态规划优化时间复杂度

```
1   class Solution {
2   public:
3       int trap(vector<int>& height) {
4           int N = height.size();
5           if (N == 0) return 0;
6
7           vector<int> leftMax(N);
8           vector<int> rightMax(N);
9
10          // 计算左侧最大高度
11          leftMax[0] = height[0];
12          for (int i = 1; i < N; i++) {
13              leftMax[i] = max(leftMax[i - 1], height[i]);
14          }
15
16          // 计算右侧最大高度
17          rightMax[N - 1] = height[N - 1];
18          for (int i = N - 2; i >= 0; i--) {
19              rightMax[i] = max(rightMax[i + 1], height[i]);
20          }
21
22          // 计算接雨水的总量
23          int water = 0;
24          for (int i = 0; i < N; i++) {
25              water += min(leftMax[i], rightMax[i]) - height[i];
26          }
27
28          return water;
29      }
30  };
```

双指针解法

```
1   class Solution {
2   public:
3       int trap(vector<int>& height) {
4           int N = height.size();
```

```
        if (N == 0) return 0;

        int left = 0, right = N - 1;
        int leftMax = 0, rightMax = 0;
        int water = 0;

        while (left <= right) {
            // 更新左右边界的最大高度
            if (height[left] < height[right]) {
                if (height[left] >= leftMax) {
                    leftMax = height[left];
                } else {
                    water += leftMax - height[left];
                }
                left++;
            } else {
                if (height[right] >= rightMax) {
                    rightMax = height[right];
                } else {
                    water += rightMax - height[right];
                }
                right--;
            }
        }

        return water;
    }
};
```