LeetCode 42 – Trapping Rain Water

Problem Statement:
Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Example 1:
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: 6 units of rainwater are trapped.

Example 2:
Input: height = [4,2,0,3,2,5]
Output: 9

Approach 1: Brute Force

Idea: For each bar, find the highest bar on the left and right. Water trapped at position i = min(left_max, right_max) - height[i].

Time Complexity: $O(n^2)$
Space Complexity: O(1)

Approach 2: Two-Pointer (Optimal)

Idea: Use two pointers (left, right) to traverse from both ends. Keep track of left_max and right_max while moving pointers inward. Water trapped is calculated using running maximums.

Time Complexity: O(n)
Space Complexity: O(1)

Approach	Time Complexity	Space Complexity	Description
Brute Force	$O(n^2)$	O(1)	Check left/right max for every index
Two Pointer	O(n)	O(1)	Use two pointers with running max

Example Dry Run (Two Pointer)

height = [4,2,0,3,2,5]
Total water trapped = 9 units.

Code Implementations

C++:
```
#include
using namespace std;
int trap(vector& height) {
int left = 0, right = height.size() - 1;
int left_max = 0, right_max = 0, water = 0;
while (left < right) {
```

```
if (height[left] < height[right]) {
if (height[left] >= left_max)
left_max = height[left];
else
water += left_max - height[left];
left++;
} else {
if (height[right] >= right_max)
right_max = height[right];
else
water += right_max - height[right];
right--;
}
}
return water;
}
```

Java:

```
class Solution {
public int trap(int[] height) {
int left = 0, right = height.length - 1;
int leftMax = 0, rightMax = 0, water = 0;
while (left < right) {
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;
}
}
```

Python:

```
def trap(height):
left, right = 0, len(height) - 1
left_max = right_max = 0
water = 0
while left < right:
if height[left] < height[right]:
if height[left] >= left_max:
left_max = height[left]
else:
water += left_max - height[left]
left += 1
else:
if height[right] >= right_max:
```

```
right_max = height[right]
else:
water += right_max - height[right]
right -= 1
return water
```