**Problem Name:** Trapping Rain water

**Topics:**

**Companies:**

**Level:** Easy

**Language:** C++

**Problem Statement**:

**Input Format:**

**Output Format:**

**Constraints:**

**Examples:**

**Brute force Solution:**

**Explanation:**

In each iteration we will compute the water we can store vertically (exactly over/above the current bar)  
So, we will check for spillings from both ends.  
We can only hold as much water till either one of the bar spills first, which will be min(left\_max\_bar , right\_max\_bar)  
But a corner case, what if the min(left\_max\_bar , right\_max\_bar) < height[i]? We will end up adding negative water to the total trapped water. In short, we won't be able to hold any water above the current bar, hence this bars contribution would be 0.

The only major concern is to be able to find left\_max\_bar and right\_max\_bar within sufficient time.  
We can always do a linear scan to the left and to the right to find those values, but it would take O(n) time, which is expensive from the standpoint of overall complexity.  
Hence, we will stick to dp.

We can preprocess the left\_max\_bars for each index i in the following manner-  
left\_max[i] = max(left\_max[i - 1] , height[i])  
And for right\_max\_bars we would do-  
right\_max[i] = max(right\_max[i+1] , height[i])

Now, we have a tool to compute left\_max\_bar and right\_max\_bar for each iteration at our disposal.

**Code:**

**Time Complexity**: O(n2)

**Space Complexity: O(1)**

**Optimized Solution:**

**Explanation:** Here is my idea: instead of calculating area by height\*width, we can think it in a cumulative way. In other words, sum water amount of each bin(width=1).  
Search from left to right and maintain a max height of left and right separately, which is like a one-side wall of partial container. Fix the higher one and flow water from the lower part. For example, if current height of left is lower, we fill water in the left bin. Until left meets right, we filled the whole container.

**Code:**

**Time Complexity**: O(n)

**Space Complexity:** O(1)