<https://leetcode.com/problems/container-with-most-water/>

**Container With Most Water**

**You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).**

**Find two lines that together with the x-axis form a container, such that the container contains the most water.**

**Return the maximum amount of water a container can store.**

**Notice that you may not slant the container.**

**Example:**

Input: height = [1,8,6,2,5,4,8,3,7]

Output: 49

Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

Constraints:

n == height.length

2 <= n <= 105

0 <= height[i] <= 104

**Method 1: (Brute Force)**

For every pair find area and compare the areas.

Time Complexity: O(n2)

Space Complexity: O(1)

**Method 2: (Two pointer)**

**Intuition:**

ex: 6, 8, 2, 1, 5, 4, 8, 3, 7  
i points to 6, j points to 7  
First pair we consider is (6,7) and so, area=min(6,7)x(j-i)=6x8=48  
Any other pair formed with 6 will always give an area with value less than (6x8)=48.

Why? : min(6,any\_value) has a maximum of only 6  
(j-i) will be lesser for all values other than 7.  
i.e., area=((not greater than 6) x (less than 8)=less than (6x8)

So we can ignore these pairs and increment i to point to 8  
and take next pair (8,7)  
Similarly decrement j when array[j] has lesser value

Time Complexity: O(n) *[for every round of loop either increment index or decrement exactly once]*

Space Complexity: O(1) *[constant space]*

int maxArea(vector<int>& height) {

        int i=0, j=height.size()-1;

        int area=0;

        while(i<j){

            int l=height[i];

            int r=height[j];

            area=max(area, (min(l,r)\*(j-i)));

            if(l<=r) i++;

            else j--;

        }

        return area;

    }