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 1:**

A picture containing bar chart

Description automatically generated

**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.

**Example 2:**

**Input:** height = [1,1]

**Output:** 1

🡺

We need to find the area first, and this is calculated using width \* height. In our example we can see that width can be calculated using difference between element indexed in the array. And we need to take a minimum heigh between two values, otherwise water overflow will be there.

e.g height = [1,8,6,2,5,4,8,3,7]   
if we check (1, 8) then we get 🡪 min(1, 8) this will give us height. And to calculate the width we get (1-0) [index of 8th and index of 1st in the height array].

So area we get 🡪 (1) \* 1 = 1;

Note – we are taking minimum of because we want a container which will store the water. We do not want a container to overflow.

Ways to solve 🡺

1. Brute force 🡪 O(n2)
2. Linear time 🡪 O(n)

* Brute force, we need to iterate from 0th ( i ) index in the outer loop and for inner loop we need to iterate from ( i+1th ) index. But here we will be comparing single element with every element in the array. Hence, resultant increase in time complexity
* Pseudo code

maxCapacity = -1;

For i = 0 to I < n-1

For j = i+1 to j < n

Capacity = min(arr[i], arr[j]) \* (j-i);

maxCapacity = Max(maxCapacity, Capacity);

return maxCapacity;

🡺 Linear Time

If we start iterating an array from both the ends. From start and last, and keep checking the maximum elements from both the ends. If any element is smaller then increment/ decrement the counter, but before that calculate the area.

You will find the code for this approach below