**# 🧱 Container With Most Water**

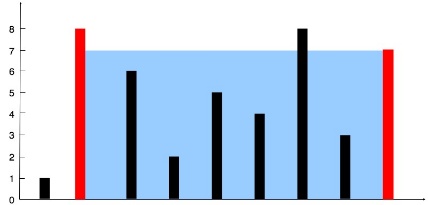
\*\*LeetCode #11\*\*

🔗 Problem link : [Container With Most Water - LeetCode](https://leetcode.com/problems/container-with-most-water/?envType=study-plan-v2&envId=leetcode-75)

**## 🧠 Problem Summary**

You are given an array `height` of `n` non-negative integers where each element represents the height of a vertical line on the x-axis.

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



**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.

**## 💡 Approach**

### ✅ Two-Pointer Technique

- Start with two pointers: one at the beginning and one at the end of the array.

- At each step:

- Calculate the area between the two lines.

- Move the pointer pointing to the \*\*shorter\*\* line inward (since moving the taller one won’t help increase area).

- Repeat until the pointers meet.

This works because the \*\*height is limited by the shorter line\*\*, and moving inward may allow a taller line and a larger area.

## 🧪 Example

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

Output: 49

Max area between heights at index 1 (8) and index 8 (7) → min(8,7) \* (8 - 1) = 7 \* 7 = 49

Important things to note:

1. Every vertical line is equally distance by 1 unit.
2. Two variables that define container size Lh, Rh: **Vol of water = min(Lh, Rh)\*Width**
3. One bar will always be included in full. Ie. Smaller of Lh, Rh.

✅ Code

public class Solution {

public int maxArea(int[] height) {

int left = 0;

int right = height.length - 1;

int maxArea = 0;

while (left < right) {

int h = Math.min(height[left], height[right]);

int width = right - left;

int area = h \* width;

maxArea = Math.max(maxArea, area);

if (height[left] < height[right]) {

left++;

} else {

right--;

}

}

return maxArea;

}

}

⏱️ Time & space complexity : O(N), O(1)