Container With Most Water

2nd Review	@October 28, 2025
# Attempts	3
Date Solved	@October 27, 2025
Difficulty	Medium
Status	Solved
Topic/Pattern	Two Pointers

LINK →

Problem

- You are given an integer array `height` where each element represents the height of a vertical line drawn on the x-axis.
- Any two lines together with the x-axis form a container.
- Return the **maximum amount of water** the container can store.

The container's capacity is determined by:

> area = min(height[i], height[j]) * (j - i)

Example

Input	Output	Reason
`[1,7,2,5,4,7,3,6]`	`36`	Max area between lines at indices 1 and 7 → min(7,6) × (7−1) = 6×6 = 36
`[2,2,2]`	`4`	Lines at 0 and 2 → min(2,2) × (2−0) = 4

Approaches

1. **Brute Force — Check Every Pair**

```
def max_area(height):
    max_water = 0
    n = len(height)

    for i in range(n):
        for j in range(i + 1, n):
            # Width between bars
            width = j - i
            # Height is limited by the shorter line
            area = min(height[i], height[j]) * width
            # Update max area
            max_water = max(max_water, area)

    return max_water
```

- **Time:** O(n²) — check all pairs of lines
- **Space:** O(1) — only uses constant extra space
- **Notes:** Simple but very slow for large inputs (TLE for big arrays)

---

2. **Two-Pointer Approach (Optimal)**

```
def max_area(height):
    i, j = 0, len(height) - 1
    max_water = 0

    while i < j:
        # Calculate current area
        width = j - i
        h = min(height[i], height[j])
        area = h * width
        max_water = max(max_water, area)
```

```
        # Move pointer from the shorter line inward
        if height[i] < height[j]:
            i += 1
        else:
            j -= 1

    return max_water
```

- **Time:** O(n) — each line is checked once
- **Space:** O(1) — no extra space used
- **Notes:**
  - Intuition: moving the smaller height inward might find a taller line and increase area.
  - Moving the taller line inward never helps because the limiting height doesn't increase.

Summary

Approach	Time	Space	Notes
Brute Force	$O(n^2)$	O(1)	Checks every pair; very slow
Two-Pointer	O(n)	O(1)	Efficient and elegant

Edge Cases

Input	Output	Reason
`[1,1]`	`1`	Only two lines → min(1,1)×(1)=1
`[1,8,6,2,5,4,8,3,7]`	`49`	Max area between lines 1 and 8 (8×7=56 → but min height limits it to 7×7=49)
`[5]`	`0`	Not enough lines
`[]`	`0`	Empty input

Mistakes

- Made a mistake while moving the pointer.
- Used long ass code instead of min(),max() function.

Tip

- The **two-pointer pattern** is common in array problems involving distance and comparison.
- Always remember:
  - Area = `min(height[i], height[j]) * (j - i)`
  - Move the pointer pointing to the **smaller height**.
- Great practice for mastering the **Greedy + Two-Pointer** technique.