Problem Statement:

Given an integer array, heights, where heights[i] represents the height of the ith bar, choose any two bars to form a container. Return the maximum amount of water that a container can store.

Solution 1

```
def max_container(heights: list[int]) -> int:
    if not heights:
        return 0

max_volume = 0

# sum of water = height of lower edge (lh) * distance between edges (d)
left, right = 0, len(heights) - 1
while left < right:
    left_height, right_height = heights[left], heights[right]
    current_volume = min(left_height, right_height) * (right - left)
    max_volume = max(current_volume, max_volume)

if left_height <= right_height:
    left += 1
    else:
        right -= 1

return max_volume</pre>
```

Step-by-Step Breakdown

1. Input:

• heights: A list of integers (e.g., [1,7,2,5,4,7,3,6]).

2. Intermittent step 1:

- Edge Case Check:
 - If heights is empty, return 0.

3. Intermittent step 2:

- Initialize Two Pointers and a Maximum Volume Variable:
 - Set left to 0 and right to len(heights) 1.
 - Initialize max_volume to 0.
- Two-Pointer Iteration:
 - o While left < right:</p>

Calculate:

- 1. current_volume = min(heights[left], heights[right]) * (right left)
 - Update max_volume:
- 1. max_volume = max(max_volume, current_volume)
 - Move the pointers:
 - If left height is less than or equal to right height, increment left; otherwise, decrement right.

4. Output:

• Return max_volume, which holds the maximum water volume found.

5. Efficiency:

- Time Complexity: O(n) Similar single pass through the list.
- Space Complexity: O(1) Only constant extra space is used (no additional list).

Visual Flow Diagram

```
1.
                  Input: heights = [1,7,2,5,4,7,3,6]
 2.
 3.
 4.
                   Check if heights is empty
 5.
 6.
 8.
                   Yes
 9.
                                                No
10.
11.
                                     Initialize left = 0, right = len(heights)-1
12.
             Return 0
14.
15.
16.
17.
18.
                                   While left < right:
19.
20.
21.
22.
                                   Calculate current_volume
                   min(heights[left], heights[right]) * (right - left)
23.
24.
25.
26.
                          Update record of volume:
28.
                          max_volume = max(max_volume, current_volume)
29.
30.
31.
32.
33.
                               Adjust pointers:
                              If heights[left] <= heights[right]</pre>
34.
36.
                                 left++
37.
                               Else
38.
                                 right--
39.
40.
41.
42.
                                     Loop until left >= right
43.
44.
45.
                                   End loop: Return max_volume
46.
```