

To solve the problem where you need to find the maximum area of water that can be contained between two vertical lines represented by an array, you can use a two-pointer approach. Here's a step-by-step explanation and implementation in Python:

Problem Explanation

Given an array of heights, each element represents the height of a vertical line drawn at each index. The area of water that can be trapped between any two lines is determined by the shorter line's height multiplied by the distance between the two lines.

Approach

1. Initialize Pointers: Start with one pointer at the beginning and one at the end of the array.
2. Calculate Area: Calculate the area between the two lines pointed by the two pointers.
3. Update Maximum Area: Keep track of the maximum area found.
4. Move the Pointers: Move the pointer pointing to the shorter line inward, as moving the taller line cannot increase the area.
5. Repeat: Continue this process until the two pointers meet.

Python Code

Here's a Python function to implement the above logic:

```
def maxArea(height):
    left = 0
    right = len(height) - 1
    max_area = 0

    while left < right:
        h = min(height[left], height[right])
        w = right - left
        max_area = max(max_area, h * w)

        # Move the pointer of the smaller height
        if height[left] < height[right]:
            left += 1
        else:
            right -= 1

    return max_area

# Example usage:
example1 = [1, 8, 6, 2, 5, 4, 8, 3, 7]
example2 = [1, 1]
```

```
print(maxArea(example1)) # Output: 49  
print(maxArea(example2)) # Output: 1
```

Explanation of the Code

- We initialize two pointers (left and right).
- We calculate the current height (h) as the minimum of the two heights at the pointers.
- We compute the width (w) as the distance between the two pointers.
- The area is then determined by multiplying height and width.
- We update max_area whenever we find a larger area.
- We move the pointer of the shorter vertical line to try and find a potentially larger area.

Complexity

- Time Complexity: $O(n)$, where n is the number of elements in the height array, as we traverse the array only once.
- Space Complexity: $O(1)$, as we are using a constant amount of space for the pointers and the max_area variable.