# Dry Run:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Step** | **left** | **height[left]** | **right** | **height[right]** | **Width** | **Height (min)** | **Area** | **Max Area** | **Move** |
| 1 | 0 | 1 | 8 | 7 | 8 | 1 | 1×8 = 8 | 8 | left++ (1 < 7) |
| 2 | 1 | 8 | 8 | 7 | 7 | 7 | 7×7 = 49 | 49 | right-- (8 > 7) |
| 3 | 1 | 8 | 7 | 3 | 6 | 3 | 3×6 = 18 | 49 | right-- |
| 4 | 1 | 8 | 6 | 8 | 5 | 8 | 8×5 = 40 | 49 | right-- |
| 5 | 1 | 8 | 5 | 4 | 4 | 4 | 4×4 = 16 | 49 | right-- |
| 6 | 1 | 8 | 4 | 5 | 3 | 5 | 5×3 = 15 | 49 | right-- |
| 7 | 1 | 8 | 3 | 2 | 2 | 2 | 2×2 = 4 | 49 | right-- |
| 8 | 1 | 8 | 2 | 6 | 1 | 6 | 6×1 = 6 | 49 | right-- |
| 9 | 1 | 8 | 1 | 8 | 0 | — | — | 49 | Stop |

# Solution:

public class Solution {

    public int maxArea(int[] height) {

        int left = 0, right = height.length - 1;

        int maxArea = 0;

        while (left < right) {

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

            int w = right - left;

            maxArea = Math.max(maxArea, h \* w);

            // Move the shorter wall inward

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

                left++;

            } else {

                right--;

            }

        }

        return maxArea;

    }

}