11. Container With Most Water

Created	@July 15, 2020 2:22 AM
Difficulty	Medium
LC Url	https://leetcode.com/problems/container-with-most-water/
Importance	****
Tag	Two pointers
Video	

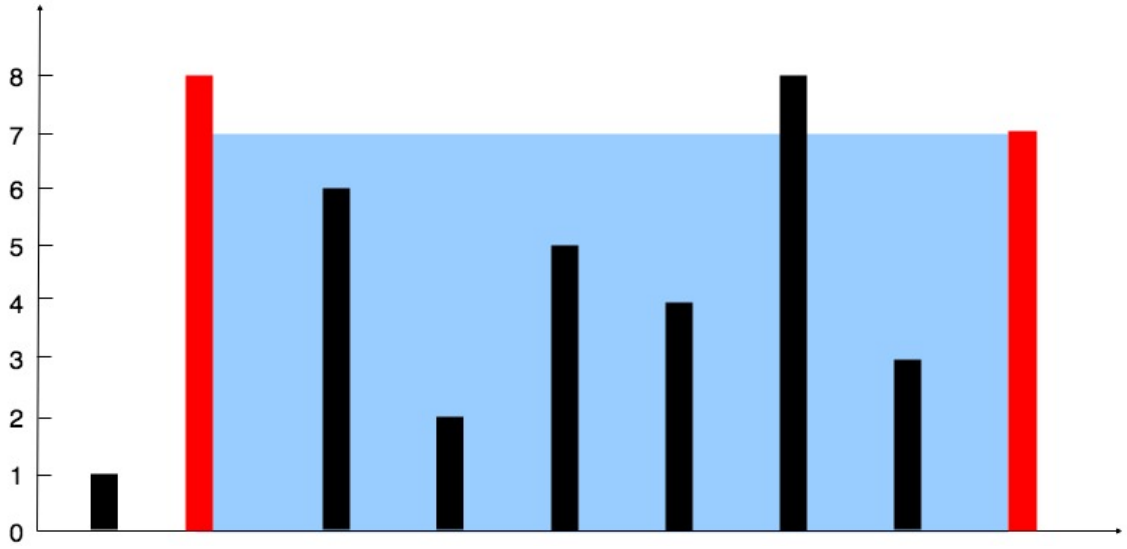
You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `i th` line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*.

Notice that you may not slant the container.

Example 1:




```
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 s
ection) the container can contain is 49.
```

Example 2:

```
Input: height = [1,1]
Output: 1
```

Constraints:

- `n == height.length`
- `2 <= n <= 10 5`
- `0 <= height[i] <= 104`

Solution

```
class Solution:
    def maxArea(self, height: List[int]) -> int:
        # 双指针
        # 链接：https://leetcode.cn/problems/container-with-most-water/solution/sheng-zui-duo-shui-de-rong-qi-by-leetcode-solution/
        left, right = 0, len(height) - 1
        ans = 0

        while left < right:
            area = min(height[left], height[right]) * (right - left)
            ans = max(ans, area)

            if height[left] <= height[right]:
                left += 1
            else:
                right -= 1
        return ans
```

复杂度分析

- 时间复杂度：$O(N)$，双指针总计最多遍历整个数组一次。
- 空间复杂度：$O(1)$，只需要额外的常数级别的空间。

力扣

https://leetcode.cn/problems/container-with-most-water/solution/sheng-zui-duo-shui-de-rong-qi-by-leetcode-solution/

https://www.bilibili.com/video/BV1a4411e7oh?p=8

```
class Solution {
    public int maxArea(int[] height) {
        if (height == null || height.length <2) return 0;
        int maxArea = 0;
        int left = 0, right = height.length - 1;
        while (left < right) {
            maxArea = Math.max(maxArea, (right - left) * Math.min(height[right], height[left]));
            if (height[right] < height[left]) {
                right--;
            } else {
                left++;
            }
        }
        return maxArea;
    }

    public static void main(String[] args) {
        Solution solver = new Solution();
        int[] arr = {1,8,6,2,5,4,8,3,7};
        System.out.println(solver.maxArea(arr));
    }
}
```