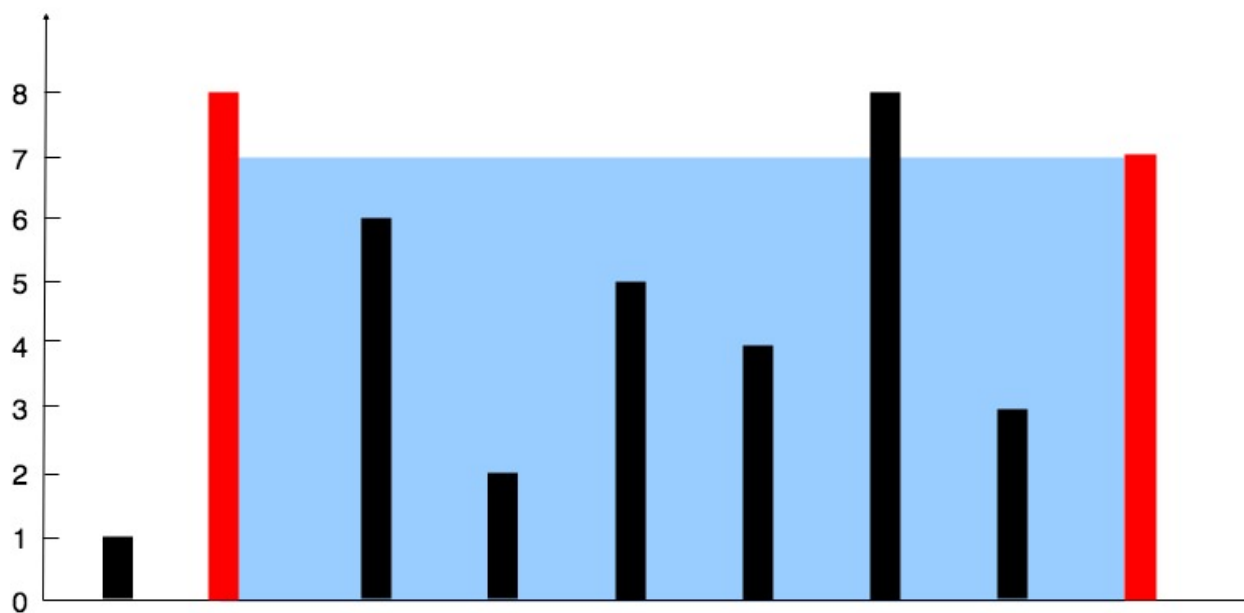


## 25 盛水最多的容器

更新时间: 2019-09-09 10:33:31



“

自信和希望是青年的特权。

——大仲马

”

### 刷题内容

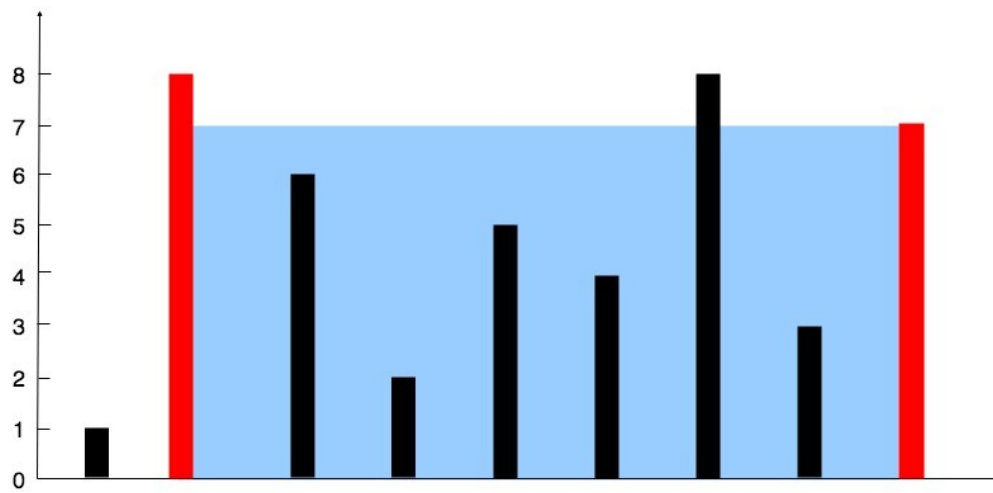
难度: Medium

题目链接: <https://leetcode-cn.com/problems/container-with-most-water/>

### 题目描述

给定  $n$  个非负整数  $a_1, a_2, \dots, a_n$ ，每个数代表坐标中的一个点  $(i, a_i)$ 。在坐标内画  $n$  条垂直线，垂直线  $i$  的两个端点分别为  $(i, a_i)$  和  $(i, 0)$ 。找出其中的两条线，使得它们与  $x$  轴共同构成的容器可以容纳最多的水。

说明：你不能倾斜容器，且  $n$  的值至少为 2。



图中垂直线代表输入数组 `[1,8,6,2,5,4,8,3,7]`。在此情况下，容器能够容纳水（表示为蓝色部分）的最大值为 **49**。

示例:

输入: `[1,8,6,2,5,4,8,3,7]`

输出: **49**

### ## 解题方案

#### 思路 : 时间复杂度:  $O(N^2)$  空间复杂度:  $O(1)$

对于任意的  $i < j$ , 由于  $a_i$  和  $a_j$  组成的 container 的面积:  $S(i, j) = \min(a_i, a_j) * (j - i)$

下面的 height 指的是对应 index 上的高度。当  $height[left] < height[right]$  时, 对任何  $left < j < right$  来说:

1.  $\min(height[left], height[j]) \leq height[left]$
2.  $height[left] = \min(height[left], height[right])$
2.  $j - left < right - left$

所以  $S(left, right) = \min(height[left], height[right]) * (right - left) > S(left, j) = \min(height[left], height[j]) * (j - left)$ 。这就排除了所有以 left 为左边界的组合, 因此需要右移 left。

因此:

-  $height[left] < height[right]$  需要右移 left。

- 同理, 当  $height[left] > height[right]$  时, 需要左移 right。

- 而当  $height[left] = height[right]$  时, 需要同时移动 left 和 right。

思路整理:  $left = 0, right = n - 1$

1.  $height[left] < height[right], left++$
2.  $height[left] > height[right], right--$
3.  $height[left] = height[right], left++, right--$

终止条件:  $left > right$

下面来看具体代码实现:

\*\*\*Python beats 82.89%\*\*\*

```
python
class Solution:
    def maxArea(self, height: List[int]) -> int:
        left, right, most_water = 0, len(height)-1, 0
        while left <= right:
            water = (right - left) * min(height[left], height[right]) # 目前这个区间能装的水有多少
            most_water = max(water, most_water)
            if height[left] < height[right]: # 此时left需要右移
                left += 1
            elif height[left] > height[right]: # 此时right需要左移
                right -= 1
            else: # 此时left右移, right左移
                left += 1
                right -= 1
        return most_water
```

**Java beats 54.04%**

```

class Solution {
public int maxArea(int[] height) {
    int left = 0;
    int right = height.length - 1;
    // 目前这个区间能装的水有多少
    int ret = min(height[left], height[right]) * (right - left);
    while (left < right) {
        if (height[left] < height[right]) {
            left++; // 此时left需要右移
        } else {
            right--; // 此时right需要左移
        }
        ret = max(ret, min(height[left], height[right]) * (right - left));
    }
    return ret;
}

public int max(int a, int b) {
    return a > b ? a : b;
}

public int min(int a, int b) {
    return a < b ? a : b;
}
}

```

**Go beats 77.51%**

```

func maxArea(height []int) int {
    left, right, most_water := 0, len(height)-1, float64(0)
    for left <= right {
        water := float64(right - left) * math.Min(float64(height[left]), float64(height[right])) // 目前这个区间能装的水有多少
        most_water = math.Max(water, most_water)
        if height[left] < height[right] { // 此时left需要右移
            left += 1
        } else if height[left] > height[right] { // 此时right需要左移
            right -= 1
        } else { // 此时left右移, right左移
            left += 1
            right -= 1
        }
    }
    return int(most_water)
}

```

**C++ beats 92.06%**

```

class Solution {
public:
    int max(int a, int b) {
        if (a > b) {
            return a;
        } else {
            return b;
        }
    }
    int min(int a, int b) {
        if (a < b) {
            return a;
        } else {
            return b;
        }
    }
    int maxArea(vector<int>& height) {
        int left = 0, right = height.size() - 1;
        // 目前这个区间能装的水有多少
        int ret = min(height[left], height[right]) * (right - left);
        while (left < right) {
            if (height[left] < height[right]) {
                left++; // 此时left需要右移
            } else {
                right--; // 此时right需要左移
            }
            ret = max(ret, min(height[left], height[right]) * (right - left));
        }
        return ret;
    }
};

```

## 小结

看到问题之后不要先着急写代码，可以把观察到的现象用数学公式表达出来。在动手写代码的时候参考你刚才写的公式来写代码，这样你的思路会更加的清晰。最后吧代码和思路大致整理一下，在脑海中行程一个大致的模板，这样的话以后遇到相似的题目也方便代码的复用。

}