LeetCode第11号问题：盛水最多的容器

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步个人博客：www.zhangxiaoshuai.fun*

本题选自leetcode的第11题，medium级别，目前通过率：61.3%

题目描述：

给你n个非负整数a1，a2，...，an，每个数代表坐标中的一个点(i,ai)。在坐标内画n条垂直线，
垂直线i的两个端点分别为(i,ai)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。
示例：
输入：[1,8,6,2,5,4,8,3,7]
输出：49

我们都应该听说过**木桶原理**，一个木桶可以装入多少水取决于最短的那块板；而这道题也可以与木桶装水的问题对应上。 很容易的可以得到---->**容器可以容纳水的容量=两条垂直线中最短的那条*两条线之间的距离** 现在的情况是，有很多条线，让你计算两两之间能装的最多的水，其实暴力法之间就能解决这个问题，但是它的时间复杂度也达到了**O(n^2)**

ok，那我们先试试用**暴力法**来解 决问题：

1.暴力法

直接上代码：

```
public int maxArea(int[] height) {
    int res = 0;
    for(int i = 0;i < height.length;i++){
      for(int j = i+1;j < height.length;j++){
        int temp = Math.min(height[i],height[j]) * (j-i);
        res = Math.max(res,temp);
      }
    }
    return res;
}
```

暴力法是可以通过测试的，但是可以看到**程序执行用时**并不理想

```
执行用时 :440 ms, 在所有 Java 提交中击败了17.44% 的用户
内存消耗 :39.9 MB, 在所有 Java 提交中击败了37.86%的用户
```

2.双指针

思路：使用两个指针（**resource**和**last**）分别指向数组的第一个元素和最后一个元素，然后我们计算这两条"线"之间能容纳的水的容量，并更新最大容量（初始值为0）；接着我们需要将指向元素值小的那个指针前移一步，然后重复上面的步骤，直到**resource = last**循环截止。

GIF动画演示：

来看看代码：

```
public int maxArea(int[] height) {
    int resource = 0;
    int last = height.length - 1;
    int res = 0;
    while (resource < last) {
        if (height[resource] >= height[last]) {
            res = Math.max(res, (last - resource) * height[last]);
            last--;
        } else {
            res = Math.max(res, (last - resource) * height[resource]);
            resource++;
        }
    }
    return res;
}
```

可以很明显的看到，虽然内存消耗两者是差不多的，但是双指针的速度比暴力解法的速度可是高出好多倍。

时间复杂度：**O(n)** 空间复杂度：**O(1)**

```
执行用时 :3 ms, 在所有 Java 提交中击败了92.69% 的用户
内存消耗 :40.3 MB, 在所有 Java 提交中击败了7.86%的用户
```

视频演示