1. 题目
2. 题干

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。

1. 示例

**输入：**[1,8,6,2,5,4,8,3,7]

**输出：**49

1. 题解
2. 思路

从题干来看，这是要找到两条垂直线，使得这两条线所围成的长方形的面积最大。因此，我们需要尽量地寻找比较长的两条线。

因为要寻找动态的两条线，因此我们可以联想到双指针法，先将两个指针分别指向数列的首端和尾端，通过比较两个指针所指的数字的大小来来移动指针。

因为我们要寻找的是两条垂直线之间的最大面积，即两条垂直线的较小值乘以两条垂直线间的距离。

如果左指针指向的数大于或等于右指针指向的数，因此此时的面积等于右指针指向的数乘以两数的间距。则将右指针向左移动一个，左指针不动。如果我们选择将左指针右移一位，右指针不动，则此时两条线的间距减1，且不论新的左指针指向的数大于或者小于右指针指向的数，两个数中较小的数一定是小于或等于右指针所指向的数的，因此面积一定小于之前的面积，与题目要求相悖。

同理，如果左指针指向的数小于右指针指向的数，则把左指针右移一位，右指针不动。

总而言之，比较两个指针所指向的数，并移动指向较小的数的指针，方向是向着另一个指针。

1. 代码实现
2. C++

#include<vector>

#include<iostream>

#include<algorithm>

using namespace std;

class Solution {

public:

int maxArea(vector<int>& height) {

int l = 0;

int r = height.size() - 1;

int ans = 0;

while (l < r) {

//此时的面积

int area = min(height[l], height[r]) \* (r - l);

ans = max(ans, area);

if (height[l] <= height[r])

l++;

else

--r;

}

return ans;

}

};

1. Java

public class Solution {

    public int maxArea(int[] height){

        int l = 0;

        int r = height.length-1;

        int ans = 0;

        while(l < r){

            int area = Math.min(height[l], height[r]) \* (r-l);

            ans = Math.max(ans, area);

            if(height[l] <= height[r]){

                ++l;

            }

            else{

                --r;

            }

        }

        return ans;

    }

}