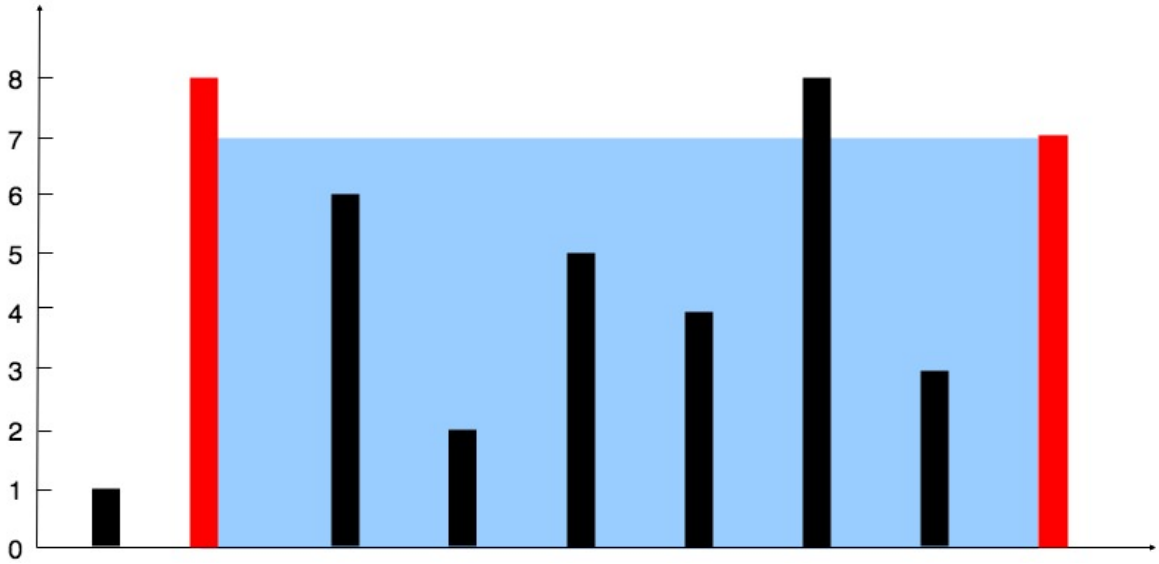
题目描述

11. 盛最多水的容器

难度:中等

给你 *n* 个非负整数 *a*1，*a*2，...，*a*n，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。



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。

代码实现

```
//借用一下别人的想法；
/*
对O(n)的算法写一下自己的理解，一开始两个指针一个指向开头一个指向结尾，此时容器的底是最大的，
接下来随着指针向内移动，会造成容器的底变小，在这种情况下想要让容器盛水变多，就只有在容器的高上下功夫。
那我们该如何决策哪个指针移动呢？我们能够发现不管是左指针向右移动一位，还是右指针向左移动一位，容器的底都是一样的，
都比原来减少了 1。这种情况下我们想要让指针移动后的容器面积增大，就要使移动后的容器的高尽量大，
所以我们选择指针所指的高较小的那个指针进行移动，这样我们就保留了容器较高的那条边，放弃了较小的那条边，
以获得有更高的边的机会。
*/
class Solution {
public:
    int maxArea(vector<int>& height) {
        int i = 0, j = height.size()-1, ans = 0;
        while (i != j)
        {
```

```
            ans = max(ans, (j - i)*min(height[i], height[j]));
            if (height[i] < height[j])
                i++;
            else
                j--;
        }
        return ans;
    }
};
```

Python实现：

```
class Solution:
    def maxArea(self, height: List[int]) -> int:
        i=0
        j=len(height)-1  # 计算List长度大小用len()函数
        ans=0
        while i!=j:
            ans=max(ans,min(height[i],height[j])*(j-i))
            # 下进行指针移动
            if height[i]<height[j]:
                i+=1
            else:
                j-=1
        return ans
```