Given *n* non-negative integers *a1*, *a2*, ..., *an*, where each represents a point at coordinate (*i*, *ai*). *n* vertical lines are drawn such that the two endpoints of line *i* is at (*i*, *ai*) and (*i*, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

**Note:**You may not slant the container and *n* is at least 2.



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 section) the container can contain is 49.

**Example:**

**Input:** [1,8,6,2,5,4,8,3,7]

**Output:** 49

蓄水两道墙求中间最大体积

可以用暴力解，

但是也可以用两个Pointer

**解题思路：**线之间形成的区域总是受限于短线的高度。此外，线越远，所获得的区域就越多。 我们使用两个指针，一个在开头，一个在数组末尾，构成行长度。此外，我们还保留了一个变量max来存储到目前为止获得的最大面积。在每步中，我们找到它们之间形成的区域，更新max并将指针指向另一端的指针。因为木桶原理，容积取决于行长度和最短高度的积，所以，两个端点高度较低的需要移动，**因为高度较高的移动不可能大于原来的两端点积，（因为每次长度都在固定少1，高的再变大还是被短的限制，短的变大才有可能变大）**。这样，每次都是高度低的移动，直到两指针相邻。

class Solution {

public int maxArea(int[] height) {

int start=0;

int end=height.length-1;

int max=0;

while(start!=end){

max=Math.max(max,(end-start)\*Math.min(height[start],height[end]));

if(height[start]<=height[end]) start++;

else end--;

}

return max;

}

}