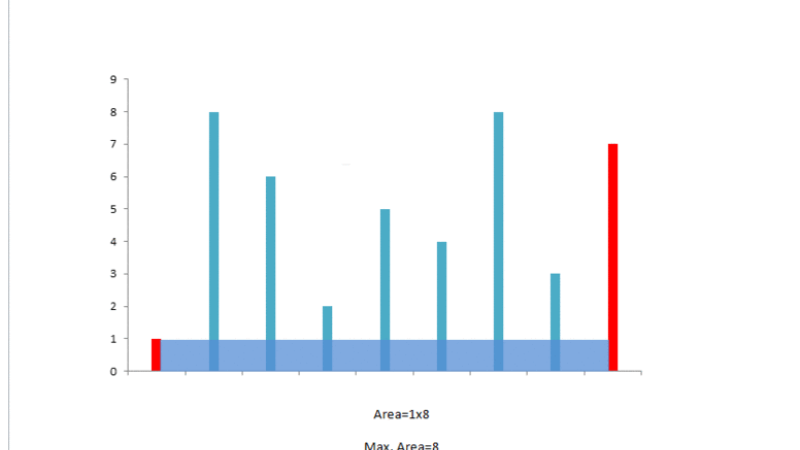
题目：

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.

题目意思如图所示：

所求为正方形面积，面积肯定是有ai中较小的数值决定的，两个边界的长度也有影响。



首先我们想到的肯定是嵌套循环，把每个可能出现的正方形面积都求一遍。然后选取其中最大的一个面积的数值输出。这种做法是最直观的做法，但是时间复杂度很大为O（n\*n）

思路就不详细写了，代码如下：

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++){

if(Math.abs((i-j))\*(Math.min(height[i],height[j]))>res)

res=Math.abs((i-j))\*(Math.min(height[i],height[j]));

}

}

return res;

}

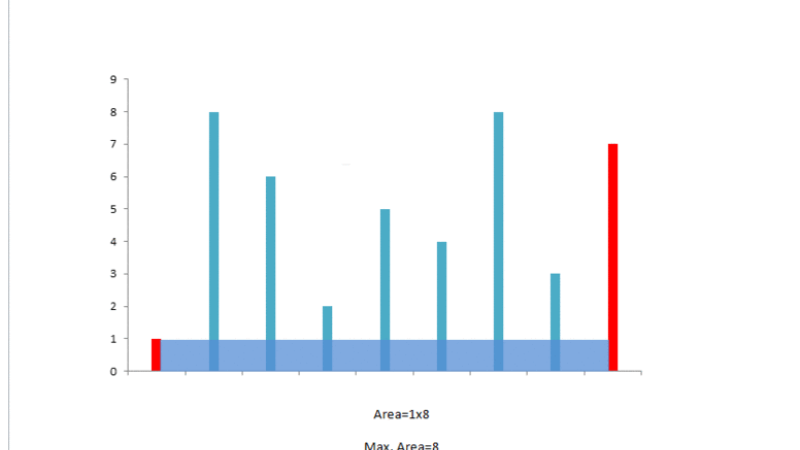
下面介绍答案给出的时间复杂度很少的方法。

思路：

设置两头向中间一起走，这样就减少一次循环。设置一个从头开始走的指针，然后在设置一个从末尾开始走的指针。L为从头开始走的指针，R为从结尾开始走的指针，两个指针根据条件相互移动，当L>R的时候走完一遍结束循环。那么根据什么条件移动指针呢？

最重要的一个条件是正方形的宽，因为正方形的宽变化比较大，其次是正方形的长，每次循环只减少1个单位。那么我们应该把移动条件定为宽的变化。宽怎么变呢？

在看看图：



决定宽是多少的是较短的宽，如图，条件是a[L]<a[R]，L=a1，R=a9。指针该如何动呢？L（左）指针动还是R（右）指针动呢？我们都试一试。当条件是啊a1<a9的时候，这时候的面积为(9-1)\*a1，左指针动的话，面积的长为8减少了一个单位，然后a2和a9比，如果a2<a1，面积为a2\*8，面积减少；如果a2>a9，面积为a9\*8，面积增大。要是R指针动的话呢面积为（9-1）\*a1，面积是缩小的。就是说在a1<a9的时候，如果动a9的话，面积是缩小的。你想长减少一个单位，如果a1<a8，宽还是a1，面积减少；如果a1>a8，宽减少，长也减少，面积肯定减少更多。所以当条件为a[L]<a[R]时移动指针，如果移动L指针面积有可能增大也有可能减小不确定；但是移动R指针就一定减少。

然后看看题目要求找到面积最大的，当条件为a[L]<a[R]时移动指针，一定是移动左指针才有可能增大，然后记录最大的那个面积输出。如果题目是找到面积最小的，当条件为a[L]<a[R]时移动指针，是移动右指针，因为面积会一直变小。

当a[L]<a[R]时，L++，此时的面积为：Math.min(a[L],a[R])\*(R-L)，然后跟每次循环的面积比找出最大的。时间复杂度为O（n）

代码如下:

public int maxArea\_A(int[] height){

int res=0;

int L =0,R=height.length-1;

while(R<L){

res=Math.max(res,(R-L)\*Math.min(height[L],height[R]));

if(height[L]<height[R])

L++;

else

R--;

}

return res;

}

心得，当遇到这种需要遍历两次数组的问题，最简单暴力的方法为嵌套循环。但是时间复杂度比较大，所以要想办法简化。之前做的，给你一个值，从数组中找哪俩个数值加一起得这个值，第一想法便是把所有得数字加起来。其实你可以简化成结果减去一个数得出得值在数组中进行查找，如果不明白向上翻题。本次题目，没有任何支柱简化，就只能遍历，所以就需要想简化遍历。两头同时开始找，就是简化了遍历得时间复杂度。