1260：拦截导弹(Noip1999)

题目描述

某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

输入导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数，导弹数不超过1000），计算这套系统最多能拦截多少导弹，如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

输入

输入导弹依次飞来的高度。

输出

第一行：最多能拦截的导弹数；

第二行：要拦截所有导弹最少要配备的系统数。

输入样例

389 207 155 300 299 170 158 65

输出样例

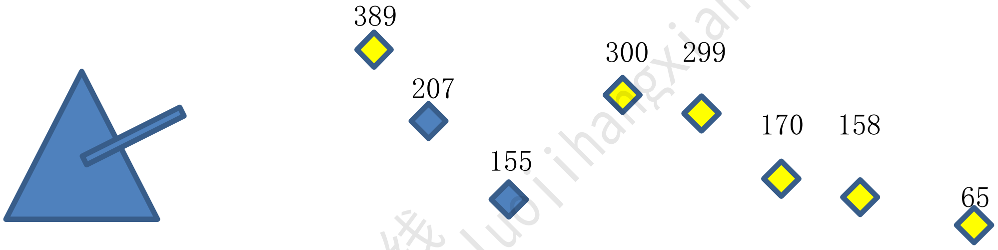
6
2

解析

先搞清楚题目的两个问题：

1、最多能拦截多少导弹？

根据题目的描述**“虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度”**，我们可以知道，这个系统最多的拦截数必然是一个最长不上升子序列。



将上述数字整理成表格

索引	1	2	3	4	5	6	7	8
高度	389	207	155	300	299	170	158	65
不上升	1	2	3	2	3	4	5	6

即上图中黄色的部分就是一个导弹系统能够拦截的最大导弹数量。

2、要拦截所有导弹最少要配备的系统数。

同理，通过题干的描述，我们明白导弹系统第一发能够达到任意的高度，但是后续的就只能比当前低。**因此，当第二发导弹高于第一发的时候，我们就必须启用一套全新的系统！**

所以，本问就变成了一个求**最大上升子序列**，那么为什么不是**最大不下降子序列呢**？这是因为当高度相同的时候，我们认为使用一套系统即可。

编码

```
#include <bits/stdc++.h>

using namespace std;
const int maxn = 1001;  //序列长度最长为1000
//a是原始数组
//up最长上升子序列，第二问
//down最长不上升子序列，第一问
int a[maxn], up[maxn], down[maxn];
int n = 0, downAns = 1, upAns = 1; //防止出现逆序数组
int main() {
    //读入输入的原始数据
   while (cin >> a[n]) {
        down[n] = up[n] = 1;
        n++;
    }
    //遍历数组中的每一个数字
   for (int i = 0; i < n; i++) {
        //开始向前寻找
       for (int j = 0; j < i; j++) {
            //找到第一个比他大的数字
          //不上升子序列
          if (a[j] >= a[i]) {
                down[i] = max(down[i], down[j] + 1);
                downAns = max(downAns, down[i]);
            }
            //上升子序列
          else {
                //每一个导弹最终都是要被打的，如果它后面有一个比它还高的导弹，
             //那么，打它的这套系统无论如何也不能再打那个导弹了。
             //因此，要求最大上升子序列。
             up[i] = max(up[i], up[j] + 1);
                upAns = max(upAns, up[i]);
            }
        }
    }
    printf("%d\n", downAns);
    printf("%d\n", upAns);
    return 0;
}
```