##### 音乐会的等待

【题目来源】洛谷P1823：<https://www.luogu.org/problemnew/solution/P1823>

【题目描述】N个人正在排队进入一个音乐会。人们等得很无聊，于是他们开始转来转去，想在队伍里寻找自己的熟人。队列中任意两个人A和B，如果他们是相邻或他们之间没有人比A或B高，那么他们是可以互相看得见的。写一个程序计算出有多少对人可以互相看见。

【输入格式】第一行包含一个整数N（1<=N<=500000），表示队伍中共有N人。接下来的N行中，每行包含一个整数，表示人的高度，以毫微米(等于10的-9次方米)为单位，每个人的调度都小于2^31毫微米。这些高度分别表示队伍中人的身高。

【输出格式】输出仅有一行，包含一个数S，表示队伍中共有S对人可以互相看见。

【样例输入】

7

2

4

1

2

2

5

1

【样例输出】

10

【题目来源】

**【**题目分析**】**

针对序列8，8，6，2，1，4，7

考虑前5人，加入第6个人之后，显然第6人可以看见3、4、5，并且可以保证4、5不被6之后的人看见；

4、5 不被6之后的人看见，将这种没希望被后面看见的人记为灰色

我们可以发现，每次插入后，剩下的彩色标记序列是非递增的；每次插入前，标上灰色的人一定是彩色序列的最后几个。这让我们想起了一个常用的数据结构：栈。

如何统计呢？第6位插入栈之前，可以看见3、4、5，并且4、5矮于第6位；所以元素在入栈之前，统计栈顶比他矮的人和恰比他高的人的人数就是这个人向前看能看到的人数！

由平摊分析得，每个人进栈一次，出栈最多一次，所以时间复杂度是O（n）。

**【**参考代码**】**

#include<stack>

#include<iostream>

using namespace std;

int x,n,ans;

stack<int>s;

int main()

{

cin>>n;

while (n--)

{

cin>>x;int t=1; //t是有多少个跟x相等的(包括x它自己)，(我是先将相等也先弹掉，这样好判断还有没存在比x大一级的数(相等的也弹掉之后的栈顶)，之后再把t个相等的弄回栈中，因为t一开始=1，所以执行t次入栈的时候就将x自己也进栈了)

while (s.size() && x>=s.top()) //弹掉比它大或等于的同时更新答案

{

if (s.top()==x)t++; //有一个相等就+1(s)

ans++;s.pop();

}

if (s.size())ans++; //如果把矮的和相等的都弹掉之后还有比x大的就+1

while (t--)s.push(x); //把之前弹掉的相等的和x本身进栈

}

cout<<ans;

return 0;

}

【我的故事】

此题背景，对我的启发，等。