Taotao Picks Apples

一. 考察内容：

前缀和 单调栈

二. 题目分析：

[题目大意]

一个长度为n的序列，从前到后从序列中挑出一些数，满足挑出的数严格大于前一个挑出的数，特殊的，第一个数必须挑，有一些对序列的修改操作，类似将第i个数修改为c，对于每一个询问，求出修改之后会挑出多少个数，其中每个询问之间是独立的。

[写题思路]

考虑离线求解，对于每一个操作的答案，我们可以看成两部分：操作点i（包括i）之前挑的个数，（不一定挑i），和i之后（不包括i）所挑的个数，最终的答案就是这两部分之和。

考虑如何求解前一部分，显然的，我们将所有操作按照操作点从小到大排序，从前到后扫描一遍，更新每个操作的第一部分即可，对于一个操作点为i的操作，如果c≤之前挑出的数，则该操作的第一部分值为f[i-1]，其中f[i]表示原序列前i个点挑出的个数，如果c>之前挑出的数，那么该操作第一部分的值为f[i-1]+1，同时，我们对于每个操作维护一个v，表示该操作第一部分选择的最后一个数的大小，以供求第二部分使用。

考虑用单调栈求解第二部分，单调栈中使得当前入栈的元素按照从栈底到栈顶，从大到小的单调顺序排列，我们从后往前逆向扫描整个序列，当扫描到i时，我们更新操作点为i的所有操作，用二分查找的方法找到栈中当前比c大且离c最近的一个节点j，第二部分的值就是f[j]。

最后每个操作的答案就是这两部分的和。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 08 21
*文件类型：源代码文件
*题目来源：HDU
*当前状态：已通过
*备忘录：单调栈 离散化 前缀和
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 200010
int n,m,a[MAXN],ls[MAXN],_ls,f[MAXN],ans[MAXN];
struct A
{
    int id,x,v;
    bool operator < (A a) const
    {
```

```
        return x==a.x?v<a.v:x<a.x;
    }
}b[MAXN];
struct DDDL
{
    int sta[MAXN],top;
    void push(int x)
    {
        while(top>=0&&a[x]>=a[sta[top]])--top;
        sta[++top]=x;
    }
    void update(int x)
    {
        while(top>=0&&x>=a[sta[top]])--top;
    }
    int query()
    {
        return top>=0?sta[top]:0;
    }
    int find(int x)
    {
        // printf("# ");
        // for(int i=0;i<=top;++i)printf("%d ",sta[i]);
        // printf("#\n");
        int l=0,r=top,ans=0;
        while(l<=r)
        {
            int m=(l+r)>>1;
            if(a[sta[m]]>x)ans=sta[m],l=m+1;
            else r=m-1;
        }
        return ans;
    }
    void reset()    {top=-1;}
}sta;
/* Variable explain:
n:元素个数
m:询问个数
a[i]:原数组
ls[i]:离散化
_ls:离散化标记
vis[i]:是否选择第i个数
f[i]:选择第i个数时后面能摘到的苹果（包括i）
ans[i]:第i个询问的答案
*/
int find(int x)
{
    int l=1,r=_ls;
    while(1)
    {
        int m=(l+r)>>1;
        if(ls[m]==x)return m;
        if(ls[m]>x)r=m-1;
        else l=m+1;
    }
}
void read()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;++i)scanf("%d",&a[i]),ls[i]=a[i];
    for(int i=1;i<=m;++i)scanf("%d%d",&b[i].x,&b[i].v),ls[n+i]=b[i].v,b[i].id=i;
    sort(ls+1,ls+n+m+1);
    sort(b+1,b+m+1);
    _ls=1;
    for(int i=2;i<=n+m;++i)
        if(ls[i]!=ls[i-1])ls[++_ls]=ls[i];
    for(int i=1;i<=n;++i)a[i]=find(a[i]);
    for(int i=1;i<=m;++i)b[i].v=find(b[i].v);
    return;
}
```

```
int main()
{
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    int TT;
    scanf("%d",&TT);
    while(TT--)
    {
        read();
        int num=0,last=0,T=1;
        for(int i=1;i<=n;++i)
        {
            while(T<=m&&b[T].x==i)//<=
            {
                if(a[last]<b[T].v)ans[b[T].id]=num+1;
                else ans[b[T].id]=num,b[T].v=a[last];
                ++T;
                // printf("%d\n",ans[b[T].id]);
            }
            if(a[last]<a[i])++num,last=i;
        }
        T=m,last=0;
        sta.reset();
        for(int i=n;i;--i)
        {
            while(T&&b[T].x==i)//>=
            {
                // printf("%d %d\n",T,b[T].v);
                ans[b[T].id]+=f[sta.find(b[T].v)];//返回大于等于v的第一个元素的位置
                --T;
            }
            sta.update(a[i]);
            f[i]=f[sta.query()]+1;
            sta.push(i);
            // printf("%d ",f[i]);
        }
        for(int i=1;i<=m;++i)printf("%d\n",ans[i]);
    }

    return 0;
}
```

<题目跳转> <查看代码>