HDU4325：Flowers（flowers.cpp）

（时间限制：2 秒　　内存限制：64MB）

http://acm.hdu.edu.cn/showproblem.php?pid=4325

【题目描述】

所周知，不同种类的花开花的时间和持续时间不同。现在有一个花园里种满了花，园丁想知道在特定的时间里花园里会开多少朵花，但是花园里的花实在太多了，所以请你帮帮他。

【输入格式】

第一行包含一个整数 t（1≤t≤10），表示测试数据的组数。

对于每组数据，第一行包含两个整数 N 和 M，其中 N（1≤N≤10^5）表示花的数量，M（1≤M≤10^5）表示询问的次数。

接下来 N 行，每行包含两个整数 S_i 和 T_i（1≤S_i≤T_i≤10^9），表示第 i 朵花将在时间区间 [S_i, T_i] 开放。

接下来 M 行，每行包含一个整数 T_i，表示第 i 次询问的时刻。

【输出格式】

对于第 i 组数据：

第一行先输出“Case #i：”　（冒号是英文的）；

接下来输出 M 行，每一行包含一个整数，表示在时刻 T_i 时，开花的花朵数量。

【输入样例】

```
2
1 1
5 10
4
2 3
1 4
4 8
1
4
6
```

【输出样例】

```
Case #1:
0
Case #2:
1
2
1
```

【分析】

题意：有 n 朵花，每一朵花只会在固定的时间范围内绽放，m 次询问，让你求在某个时刻有多少朵花是绽放的。

思路：这题可以看成是树状数组的区间更新，给出每一朵花开放的时间范围，就相当于做一次区间更新，这个范围内的每个点加 1，直接使用树状数组区间更新的模板就可以。

但是花开放范围的数据特别大（$1\sim10^9$），无法直接开数组储存每一个点。那怎么办呢，这就需要用到数据的离散化。

我们可以发现，虽然花开放范围极大，但是总共只有 n 朵花，n 最大是 10^5，每一个范围由两个数 l，r 组成，再加上最多 10^5 个查询数，也就意味着一组数据最多出现 $3*10^5$ 个不同的数据，数组完全能够存下。而且我们判断花是否在某一时刻开放，只需要知道数与数之间的相对大小，而不需要在意具体数值（某一时刻在[l,r]之间，即花是开的），所以，这就满足了离散化的条件。

我们对每一朵花开放的范围进行离散化，也对查询的时刻进行离散化，这样，就不会改变他们的相对大小。然后使用树状数组区间更新+单点查询即可。

离散化的方法：将所有出现的数存入数组，去除重复出现的数，再进行排序，排序后这个数在数组中对应的下标，即为其离散化之后的数值。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;

const int MAXN=3e5+21;   //3*N
int getNum[MAXN]          //离散化数组，之所以是 3*N 的空间，是因为要存 N 朵花开放的起、始时间、查询时间
int li[MAXN/3],ri[MAXN/3],c[MAXN]; //花开的起始时间、结束时间、树状数组

int lowbit(int x)
{
    return x&(-x);
}

void add(int x,int num,int len)  //区间更新
{
    for(int i=x;i≤len;i+=lowbit(i))
    {
        c[i] += num;
    }
}

ll query(int x)  //单点查询
{
    ll ans=0;
    for(int i=x;i>0;i-=lowbit(i))
    {
        ans+=c[i];
    }
    return ans;
}

int bSerach(int num,int len)    //在数组 getNum 中找出大于等于 num 的数，并返回其下标+1（则树状数组下标从 1 开始）
{
    return lower_bound(getNum,getNum+len,num)-getNum+1;
}
```

```
int main()
{
    int t,n,m,cnt=0;
    cin>>t;
    while(t--)
    {
        memset(c,0,sizeof(c));
        memset(getNum,0,sizeof(getNum));
        scanf("%d%d",&n,&m);
        int cnt1=0;
        for(int i=0; i<n; ++i)
        {
            scanf("%d%d",&li[i],&ri[i]);
            getNum[cnt1++]=li[i];            //将花开放范围出现的数据存入数组
            getNum[cnt1++]=ri[i];
        }
        int ques[MAXN/3];                    //存储查询时间
        for(int i=0;i<m;++i)
        {
            scanf("%d",&ques[i]);
            getNum[cnt1++]=ques[i];           //将查询出现的数据存入数组
        }
        sort(getNum,getNum+cnt1);             //给所有出现的数排序

        int cnt2=1;
        for(int i=1;i<cnt1;++i)
        {
            if(getNum[i]!=getNum[i-1])        // getNum[0]～getNum[cnt2-1]存的是去重后的数
                getNum[cnt2++]=getNum[i];
        }

        for(int i=0;i<n;++i)  //树状数组区间更新
        {
            add(bSerach(li[i],cnt2),1,cnt2);     //位置 bSerach(li[i],cnt2)～位置 cnt2 均增加 1
            add(bSerach(ri[i],cnt2)+1,-1,cnt2); //位置 bSerach(lr[i]+1,cnt2)～位置 cnt2 均增加-1
        }
        printf("Case #%d:\n",++cnt);
        for(int i=0;i<m;++i)
        {
            cout<<query(bSerach(ques[i],cnt2))<<endl; //树状数组单点查询
        }                    //bSerach(ques[i],cnt2)是查找 ques[i]在离散化数组中的位置
    }
    return 0;
}
```