时间限制：2秒

空间限制：65536K

为了找到自己满意的工作，牛牛收集了每种工作的难度和报酬。牛牛选工作的标准是在难度不超过自身能力值的情况下，牛牛选择报酬最高的工作。在牛牛选定了自己的工作后，牛牛的小伙伴们来找牛牛帮忙选工作，牛牛依然使用自己的标准来帮助小伙伴们。牛牛的小伙伴太多了，于是他只好把这个任务交给了你。

**输入描述:**

每个输入包含一个测试用例。  
每个测试用例的第一行包含两个正整数，分别表示工作的数量N(N<=100000)和小伙伴的数量M(M<=100000)。  
接下来的N行每行包含两个正整数，分别表示该项工作的难度Di(Di<=1000000000)和报酬Pi(Pi<=1000000000)。  
接下来的一行包含M个正整数，分别表示M个小伙伴的能力值Ai(Ai<=1000000000)。  
保证不存在两项工作的报酬相同。

**输出描述:**

对于每个小伙伴，在单独的一行输出一个正整数表示他能得到的最高报酬。一个工作可以被多个人选择。

**输入例子1:**

3 3

1 100

10 1000

1000000000 1001

9 10 1000000000

**输出例子1:**

100

1000

1001

对于难度一样的工作，当然是只会选那个报酬最大的

使用map将工作需要的能力和给的报酬一一对应，map中会对key值自动排序

将人的能力也插入到这张map中，

遍历map，使得能力值对应的报酬最大化

#include<stdio.h>

#include<iostream>

#include<cstring>

#include<set>

#include<map>

typedef long long ll;

using namespace std;

ll d[100010],v[100010],abi[100010];

int main()

{

//freopen("input.txt","r",stdin);

int n,m;

cin>>n>>m;

for(int i=0;i<n;i++)

cin>>d[i]>>v[i];

map<ll,ll>\_map;

for(int i=0;i<n;i++)

if(\_map[d[i]]<v[i])

\_map[d[i]]=v[i];

for(int i=0;i<m;i++)

{

cin>>abi[i];

\_map.insert(make\_pair(abi[i],0));////如果存在\_map[abi[i])]，返回false，如果不存在，就新建一个abilitys[i]，0

}

map<ll,ll>::iterator tmp=\_map.begin(),pre=\_map.begin();

tmp++;

for(;tmp!=\_map.end();tmp++,pre++)

{

if(tmp->second < pre->second)//只可能是新插入的人的情况，因为他们的second都赋值0了

tmp->second=pre->second;

}

for(int i=0;i<m;i++)

cout<<\_map[abi[i]]<<"\n";

return 0;

}