题意：有n个机器，m个任务。每个机器至多能完成一个任务。对于每个机器，有一个最大运行时间xi和等级yi，对于每个任务，也有一个运行时间xj和等级yj。只有当xi>=xj且yi>=yj的时候，机器i才能完成任务j，并获得500\*xj+2\*yj金钱。问最多能完成几个任务，当出现多种情况时，输出获得金钱最多的情况。

题解：

将任务已x从大到小排序（x相同时已y从大到小排序）。然后也用相同排序方法排序机器。开始遍历任务，找出所有xi（xi>=xj），从中选择yi最小的一个作为这个任务的运行机器。为什么这么贪心，因为若还存在任务(xk,yk)使得这个机器能被使用，但xj>=xk，所以获得金钱更多，优先选择j；若k不能使用这个机器，那么必定也就不存在其他机器能被使用，除非是新加入的机器，但新加入的必定不能完成任务j，所以完成任务保证了最多。  
  
从贪心的角度来说，已经知道去找符合时间的机器中等级最低的那个机器就好，自己做的时候没有想到用c[]数组去记录能够符合时间的各等级的机器数，而是每个任务都具体指定出来了一个机器，从而把时间浪费来标记的过程中，

#include <iostream>

#include <bitset>

#include<cstdio>

#include<queue>

#include<set>

#include<cstring>

#include<algorithm>

#define ll long long

using namespace std;

struct node

{

int x,y;

}machine[100010],task[100010];

bool cmp(node a,node b)

{

if(a.x==b.x)

return a.y>b.y;

return a.x>b.x;

}

bool cmp1(node a,node b)

{

return a.y<b.y;

}

int c[110];

int main()

{

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

//ios::sync\_with\_stdio(false);

//cin.tie(0);

int n,m;

while(cin>>n>>m)

{

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

cin>>machine[i].x>>machine[i].y;

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

cin>>task[i].x>>task[i].y;

sort(machine,machine+n,cmp);

sort(task,task+m,cmp);

ll ans(0),cnt(0);

memset(c,0,sizeof(c));

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

{

while(j<n && machine[j].x>=task[i].x)

{

c[machine[j].y]++;

j++;

}

for(int k=task[i].y;k<=100;k++)

if(c[k])

{

ans+=(500\*task[i].x+2\*task[i].y);

c[k]--;

cnt++;

break;

}

}

cout<<cnt<<' '<<ans<<"\n";

}

return 0;

}