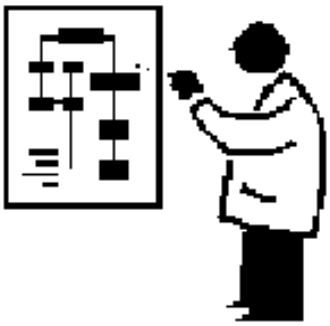
6193 Component Testing

The engineers at ACM Corp. have just developed some new components. They plan to spend the next two months thoroughly reviewing and testing these new components. The components are categorized into several different classes, depending on their complexity and importance. Components in different classes may require different number of reviewers, whereas components in the same class always require the same number of reviewers.



There are also several different job titles at ACM Corp. Each engineer holds a single job title. All engineers holding a given job title have the same limit on the number of components that they can review. Note that an engineer can be assigned to review any collection of components and will be able to complete the task, regardless of which classes the components belong to. An engineer may review some components of the same class, and others from different classes, but an engineer cannot review the same component more than once.

Can the engineers complete their goal and finish testing all components in two months?

Input

There will be multiple test cases in the input.

The first line of each test case contains two integers n ($1 \leq n \leq 10,000$) and m ($1 \leq m \leq 10,000$), where n is the number of component classes and m is the number of engineer job titles.

Each of the next n lines contains two integers j ($1 \leq j \leq 100,000$) and c ($0 \leq c \leq 100,000$), indicating that there are j components in this class and that each component requires at least c different reviewers.

Then each of the next m lines each contains two integers k ($1 \leq k \leq 100,000$) and d ($0 \leq d \leq 100,000$), indicating that there are k engineers with this job title and that each engineer may be assigned to review at most d components.

The input will end with a line with two `0`'s.

Output

For each test case, print a single line containing '`1`' if it is possible for the engineers to finish testing all of the components and '`0`' otherwise.

Sample Input

```
3 2
```

```
2 3
1 2
2 1
2 2
2 3
5 2
1 1
1 3
1 1
1 3
1 1
1 20
1 4
0 0
```

Sample Output

```
1
0
```