863 Process Scheduling

In batch programming, there are many tasks that need to be completed in minimal overall time. Accurate information about the total runtime and interdependencies of processes is often available to the scheduler, allowing optimal schedulings to be found. This problem considers processes that can run on different CPUs at the same time, but which are interdependent. That is to say, some processes cannot start until others have completed. Scheduling processes optimally prevents CPUs from being idle unnecessarily.

Your task is to find a way of using n processors to run p processes, minimizing the number of time slices until they are all finished. Some processes may be dependent on processes that depend on other processes, but there will be no dependency loops.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of input will consist of two integers, the number of processors (n) and the number of processes (p). Both numbers will be positive; $p < 100$ and $n < 21$.

There will be p additional lines representing the processes. Each of these lines will contain an integer representing the number of CPU time slices required for the process and zero or more integers representing the processes that must finish before this process may start. Processes are numbered from 1 to p.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Output consists of n columns of width two, with a single space between columns. The i-th line corresponds to the processes running during the i-th time slice. The process numbers should be right-justified. The scheduling of processes should minimize the amount of time until all of the processes have completed their tasks. Trailing spaces are acceptable, provided that they do not extend beyond the right of the last column.

Sample Input

```
1

3 5
4
3
2 4 2
2 1
1 3
```

Sample Output

```
 1  1  2
```

```
1 1 2
4 4 2
3 3
5
```