Optimization

One of the largest companies in Gerland is Ziebenz. They have huge projects and countless employees. Operating at such a large scale can be challenging. For example, one of their current projects contains m independent jobs and involves n workers. The manager of the project has accurately calculated how much working time each part of the project requires. It turns out that the *execution time* for each job is fixed and doesn't depend on the employee who will perform it (all workers are equally talented MUT graduates). After computing the execution times, the manager has randomly (don't judge him, he's a former MUT-LWB student!) assigned all m jobs to n workers.

Of course, it has turned out that some workers will need much longer to complete their jobs. Now the manager wants to improve the situation: he's going to pick up two different workers, and for each worker choose one of jobs assigned to them. Afterwards, he's going to swap the two jobs between the both workers (i.e. assign the first workers' job to the second worker, and vice versa). We'll cal such operation *optimizing*, if the maximum of the overall working times of both workers has decreased after performing the operation.

Let's look at an example. Imagine that the project consists of 5 jobs with execution times of $3, 6, 4, 8$ and 2, and there're 3 workers involved in this project. The initial random assignment is: the first worker gets jobs 1 and 2 (overall working time: $3 + 6 = 9$), the second worker — job 4 (overall working time 8), and the third — jobs 3 and 5 (overall working time $4 + 2 = 6$). Now, if we assign the first job (initially assigned to the first worker) to the third worker, and the fifth job (initally assigned to the third worker) to the first worker, the overall working time of the first worker becomes $6 + 2 = 8$, and of the third worker — $3 + 4 = 7$. Thus, it is an optimizing operation.

You are given the number of employees and the number of jobs, execution times of all jobs, and the initial assignment of the jobs to the workers. Find the number of possible optimizing operations for the given assignment of jobs.

Input

The first line of the input contains an integer $t \leq 20$. t test cases follow, each of them separated by a blank line. Each testcase has the following structure.

The first line contains two integers n and m ($1 \leq n, m \leq 10^5$) — the number of employees and the number of jobs within the project.

The second line contains m integers where the $i-$th number is the execution time of the $i-$th job (jobs are numbered from 1 to m). All execution times are between 1 and 10^9.

The following n lines describe the assignment of the jobs to the workers. The $i-$th of these lines begins with the number k_i — the total number of jobs assigned to the worker i. k_i integers follow — the numbers of the jobs assigned to the worker.

Output

For each test case output a line containing "Case #i: s", where s is the number of possible optimizing operations.

Sample Input 1

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

Sample Output 1

```
Case #1: 2
```

Sample Input 2

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

Sample Output 2

```
Case #1: 4
```