UVa Online Judge

13000 VIP Treatment

For any government project, VIPs are always a problem. You can't expect a high profile VIP person to wait for a month for some job to be done like a normal people. This is a big problem for government software projects as you have to create alternate ways to bypass business logic and finish some job early. Now you are working on some government software project and you have to manage some job requests, processing and management as well as have to do VIP treatment. In this problem we will solve a sub-problem of it, assigning the jobs to worker.

For the software there are M kinds of jobs, numbered from 1 to M and N workers numbered from 1 to N. As the jobs are mostly similar, so the time needed for a particular worker to do any kind of job request is same. Different worker may need different amount of time. Now for each j-th ($1 \leq j \leq M$) kind of job, your software have to consider 3 things,

- Number of VIP requests v_j for this kind of job.
- Number of regular requests r_j for this kind of job.
- A non-empty list of workers wr_j who can do this kind of job.

All the VIP job request have to be processed. But the management also wants to process at least K ($0 \leq K \leq \sum_{j=1}^{M}) r_j$) total regular job requests.

The software have to assign this requests to workers and have to do it in a way that total time needed is minimized. Now a worker numbered i ($1 \leq i \leq N$) can complete any kind of job request in W_i time but can't do more than one job at a time. After the requested job is finished, he can do another job again, which will cost him another W_i time. So if any worker numbered i does total T_i jobs, it would take him $T_i * W_i$ time to finish.

The total time needed to finish the project is the maximum time needed by any worker (Maximum $T_i * W_i$ for all i such that $1 \leq i \leq N$). Your job for this problem is to make such an arrangement that the total time needed is minimized.

Input

The first line of input contains a positive integer number TC ($TC \leq 200$), number of test case. Then TC test cases follow.

There will be blank line before each test case.

First line of each test case will contain three space separated integers M ($1 \leq M \leq 50$), N ($1 \leq N \leq 50$) and K. Next line will contain N space separated integer numbers, i-th of those numbers will be W_i, time needed by worker i to do a single job ($1 \leq W_i \leq 100$). Each of the next M lines contains description of a job, where the first three integers of the j-th line are v_j ($0 \leq v_j \leq 1,000,000$) number of VIP job requests of kind j, r_j ($0 \leq r_j \leq 1,000,000$) number of regular job requests of kind j and n_j ($1 \leq n_j \leq N$) number of workers who can do j-th kind of job. Then n_j integers follow. The k-th of the next n_j integers is wr_{jk} ($1 \leq wr_{jk} \leq N$), means worker numbered wr_{jk} can do j-th kind of job.

Output

For each test case print a line in '`Case` I`:` T' format where I is the case number and T is the minimum time required for doing all VIP requests and at least K regular job requests.

Note: Explanation for Sample Case

In the first case, there are $3 + 3 + 4 = 10$ regular job requests and 10 VIP job requests. Also for each type of job, only one worker can do it. So

- The 1st worker get $2 + 3 = 5$ jobs and with 2 unit time per job he needs $5 * 2 = 10$ unit of time.
- The 2nd worker get $2 + 3 = 5$ jobs and with 4 unit time per job he needs $5 * 4 = 20$ unit of time.
- The 3rd worker get $2 + 4 = 6$ jobs and with 8 unit time per job he needs $6 * 8 = 48$ unit of time.

As they all can work independently, total time needed is 48.

In the second case, there in only one worker and he got total $2 + 3 = 5$ VIP job requests and 4 regular job requests. So He need $(5 + 4) * 2 = 18$ unit of time.

In the third case, worker 1 will do 6 jobs with $6 * 1 = 6$ unit of time and worker 2 will do 3 jobs with $3 * 2 = 6$ unit of time. So to do 9 jobs, total time needed is 6.

Sample Input

```
3

3 3 10
2 4 8
2 3 1 1
2 3 1 2
2 4 1 3

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

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

Sample Output

```
Case 1: 48
Case 2: 18
Case 3: 6
```