Problem 3: Into the Groove

Hackers have recently been trying to break through a multi-stage firewall system that is protecting sensitive information at Acme Corp. Unfortunately the sysadmin is very busy and has a limited number of chances to stop the hackers. In each attack, the hackers have made M attack-attempts. Each stage i, $1 \leq i \leq N$, in the firewall system has an associated probability-pair (p_i, q_i) where p_i is the probability of the sysadmin failing to stop a hacker at stage i if the sysadmin gets involved at that stage and q_i is the probability of the firewall failing to stop a hacker at stage i if the sysadmin decides to ignore that stage. Given these probabilities, it is your job to select those stages that the sysadmin should focus on so as to maximize the probability of stopping the hackers.

Write a program which, given n descriptions of system attack scenarios, computes and outputs for each scenario the stages the sysadmin should select to minimize the probability of a hacker gaining access. The selected stages should be output in ascending order. If multiple selections exist with minimal hacker access probability, output the lexicographically smallest selection (*e.g.*, `[1, 2, 9]` $<$ `[2, 3, 4]`). Your input will be an $(n+1)$-line file in which the first line is the value of n and each of the following lines contains a particular attack scenario specified by the values of N and M followed by N probability-pairs (p_i, q_i). Recall that the probability of event A and event B both occurring is the product of each of their probabilities. You may assume that each stage is independent from the others probability-wise, $M \leq N$, $0 \leq p \leq q \leq 100$, and all input files are formatted correctly.

Sample input #1 (available as file "test3a.dat"):

```
3
1 1 50 60
2 1 50 80 50 90
3 2 30 90 90 100 20 80
```

Sample output #1:

```
[1]
[2]
[1, 3]
```

Sample input #2 (available as file "test3b.dat"):

```
2
3 1 50 60 25 80 60 70
3 2 50 60 25 80 60 70
```

Sample output #2:

```
[2]
[1, 2]
```

Sample input #3 (available as file "test3c.dat"):

```
2
3 2 50 60 10 20 15 20
3 2 15 25 10 20 15 20
```

Sample output #3:

```
[2, 3]
[1, 2]
```