Customs

Making money is hard, not just for people but also for countries. It is so hard that an especially broke country, Poorland, has come up with a completely new idea: They will split their country into two countries. This way, they can collect customs duties each time someone passes from one country to another.

As they want to maximize the amount of money they collect, the number of roads that lead from one of the new countries to the other should be maximal. Finding an optimal solution is very costly, thus they asked Lea to find a solution that differs from the optimal solution by at most a factor of $\frac{1}{2}$. As streets start and end at cities, the task will be to partition the cities into two sets.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a number n, the number of cities in Poorland, n lines follow. The i-th line describes city i and contains an integer k_i followed by k_i integers $c_1, ..., c_{k_i}$. k_i is the number of streets exiting city i, $c_1, ..., c_{k_i}$ are the cities directly connected to city i by streets. Cities are indexed from 1 to n.

You can assume that streets are undirected: If city i is directly connected to city j, then city j will be directly connected to city i.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1. Output one more line containing integers $a_1...a_l$ in ascending order such that when the country is partitioned into the cities $a_1, ..., a_l$ and the remaining cities, the number of roads between the parts is at least half as big as in the optimal solution. If there are multiple solutions, any of them will be accepted. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 300$
- $0 \leq k_i \leq n$ for all $1 \leq i \leq n$
- $1 \leq c_i \leq n$ for all $1 \leq i \leq k_i$

Sample Input 1	Sample Output 1
2 2 1 2 1 1 4 1 3 1 4 2 1 4 2 2 3	Case #1: 1 Case #2: 1 2 4

Sample Input 2

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

4
0
2 3 4
1 2
1 2

5
1 4
0
0
1 1
0

3
1 2
1 1
0

4
2 2 3
2 1 4
1 1
1 2

6
1 4
2 4 5
2 4 6
5 1 2 3 5 6
2 2 4
2 3 4

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

6
1 4
2 3 5
2 2 4
3 1 3 5
2 2 4
0
```

Sample Output 2

```
Case #1:
1 3 5 6
Case #2:
1 2
Case #3:
1 2 3 5
Case #4:
1 3
Case #5:
1 4
Case #6:
1 2 3 5 6
Case #7:
1 3
Case #8:
1 3 5 6
```