10319 Manhattan

You are the mayor of a city with severe traffic problems. To deal with the situation, you have decided to make a new plan for the street grid. As it is impossible to make the streets wider, your approach is to make them one-way (only traffic in one direction is allowed on a street), thus creating a more efficient flow of traffic.

The streets in the city form an orthogonal grid — like on Manhattan avenues run in north-south-direction, while streets run in east-west-direction. Your mission is to make all the streets and avenues one-way, i.e. fix the direction in which traffic is allowed, while maintaining a short driving distance between some ordered pairs of locations. More specifically, a route in the city is defined by two street-avenue crossings, the start and goal location. On a one-way street grid, a route has a legal path if it is possible to drive from the start location to the goal location along the path passing streets and avenues in their prescribed direction only. A route does not define a specific path between the two locations - there may be many possible paths for each route. A legal path in a one-way street grid is considered simple if it requires at most one turn, i.e. a maximum of one street and one avenue need to be used for the path.

When traveling by car from one location to another, a simple path will be preferred over a non-simple one, since it is faster. However, as each street in the grid is one-way, there may always be routes for which no simple path exists. On your desk lies a list of important routes which you want to have simple paths after the re-design of the street grid.

Your task is to write a program that determines if it is possible to fix the directions of the one-way streets and avenues in such a way that each route in the list has at least one simple path.

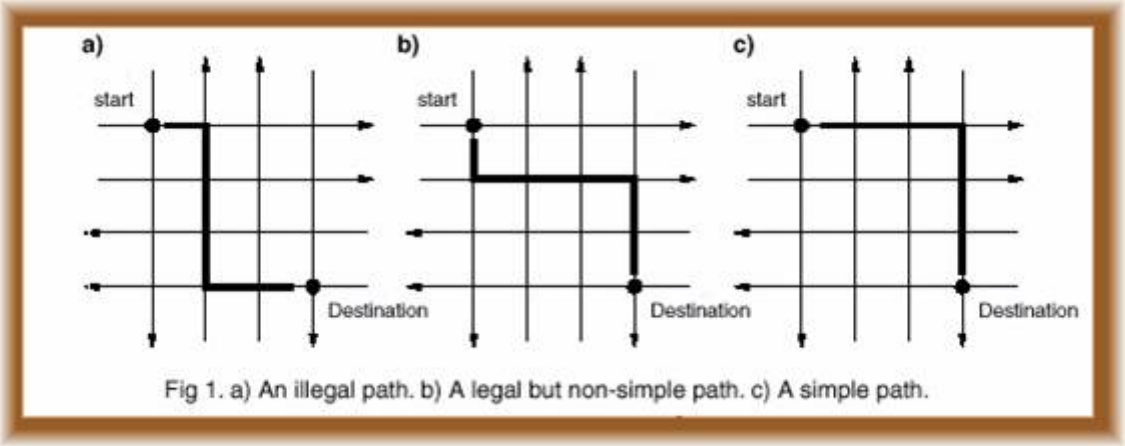

Fig 1. a) An illegal path. b) A legal but non-simple path. c) A simple path.

Input

On the first line of the input, there is a single integer n, telling how many city descriptions that follows. Each city description begins with a line containing three integers: the number of streets $0 < S \leq 30$ and avenues $0 < A \leq 30$ in the street grid, and the number of routes $0 < m \leq 200$ that should have at least one simple path. The next m lines define these routes, one on each line. Each route definition consists of four integers, $s1$, $a1$, $s2$, $a2$, where the start location of the route is at the crossing of street $s1$ and avenue $a1$, and the goal location is at the crossing of street $s2$ and avenue $a2$. Obviously, $0 < s1, s2 \leq S$ and $0 < a1, a2 \leq A$.

Output

For each city, your program should output '`Yes`' on a single line if it is possible to make the streets and avenues one-way, so that each route has at least one simple path. Otherwise the text '`No`' should be printed on a line of its own.

Sample Input

```
3
6 6 2
1 1 6 6
6 6 1 1
7 7 4
1 1 1 6
6 1 6 6
6 6 1 1
4 3 5 1
9 8 6
2 2 4 4
4 5 3 2
3 4 2 2
3 2 4 4
4 5 2 2
2 1 3 4
```

Sample Output

```
Yes
No
No
```

C. Checkposts

Your city has n junctions. There are m **one-way** roads between the junctions. As a mayor of the city, you have to ensure the security of all the junctions.

To ensure the security, you have to build some police checkposts. Checkposts can only be built in a junction. A checkpost at junction i can protect junction j if either $i = j$ or the police patrol car can go to j from i and then come back to i.

Building checkposts costs some money. As some areas of the city are more expensive than others, building checkpost at some junctions might cost more money than other junctions.

You have to determine the minimum possible money needed to ensure the security of all the junctions. Also you have to find the number of ways to ensure the security in minimum price and **in addition in minimum number of checkposts**. Two ways are different if any of the junctions contains a checkpost in one of them and do not contain in the other.

Input

In the first line, you will be given an integer n, number of junctions $(1 \le n \le 10^5)$. In the next line, n space-separated integers will be given. The i^{th} integer is the cost of building checkpost at the i^{th} junction (costs will be non-negative and will not exceed 10^9).

The next line will contain an integer m $(0 \le m \le 3 \cdot 10^5)$. And each of the next m lines contains two integers u_i and v_i $(1 \le u_i, v_i \le n;\ u \ne v)$. A pair u_i, v_i means, that there is a one-way road which goes from u_i to v_i. There will not be more than one road between two nodes in the same direction.

Output

Print two integers separated by spaces. The first one is the minimum possible money needed to ensure the security of all the junctions. And the second one is the number of ways you can ensure the security modulo 1000000007 $(10^9 + 7)$.

Examples

input

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

output

```
3 1
```

input

```
5
2 8 0 6 0
6
1 4
1 3
2 4
3 4
4 5
5 1
```

output

```
8 2
```

input

```
10
1 3 2 2 1 3 1 4 10 10
12
1 2
2 3
3 1
3 4
4 5
5 6
5 7
6 4
7 3
8 9
9 10
10 9
```

output

```
15 6
```

input

```
2
7 91
2
1 2
2 1
```

output

```
7 1
```

UVa Online Judge

11294 Wedding

Up to thirty couples will attend a wedding feast, at which they will be seated on either side of a long table.

The bride and groom sit at one end, opposite each other, and the bride wears an elaborate headdress that keeps her from seeing people on the same side as her. It is considered bad luck to have a husband and wife seated on the same side of the table. Additionally, there are several pairs of people conducting adulterous relationships (both different-sex and same-sex relationships are possible), and it is bad luck for the bride to see both members of such a pair. Your job is to arrange people at the table so as to avoid any bad luck.

Input

The input consists of a number of test cases, followed by a line containing '`0 0`'. Each test case gives n, the number of couples, followed by the number of adulterous pairs, followed by the pairs, in the form '`4h 2w`' (husband from couple 4, wife from couple 2), or '`10w 4w`', or '`3h 1h`'. Couples are numbered from 0 to $n-1$ with the bride and groom being '`0w`' and '`0h`'.

Output

For each case, output a single line containing a list of the people that should be seated on the same side as the bride. If there are several solutions, any one will do. If there is no solution, output a line containing '`bad luck`'.

Sample Input

```
10 6
3h 7h
5w 3w
7h 6w
8w 3w
7h 3w
2w 5h
0 0
```

Sample Output

```
1h 2h 3w 4h 5h 6h 7h 8h 9h
```