Problem 1. Friend group

(Time Limit: 1 seconds)

Problem Description

There are several people participating a party and some of them are friends. A group of people is a friend group if the people in the group are mutually friends. A maximum friend group is a friend group with maximum size. Your task is to find the size of maximum friend group in the party.

Input Format

There are several test cases. For each test case, the first line contains two integers n and m , in which n is the number of people and m is the number of friend pairs. In the following m lines, the people are labeled from 0 to $n-1$, and friend pairs are given by the labels of two people, line by line. It is assumed that $1 < n < 31$.

Output Format

For each test case, output the size of a maximum friend group in one line.

Example

Sample Input:	Sample Output:
5 6 0 1 1 2 1 3 1 4 2 3 3 4	3

Problem 2. Social Network

(Time Limit: 3 seconds)

Problem Description

On-line social networking site has become a popular tool for making friends. An actor in a social network can have followees and followers at the same time. Let us consider a social network that is a rooted tree. The root of such a social network is the only actor who does not have any followee. Each actor except the root has exactly one followee and several (perhaps 0) followers. An actor without any followers is called a leaf. The depth of the network is the maximum distance from the root to a leaf. The degree of an actor is the number of actors who has a (either following or followed) relationship with the actor. Due to the capacity of social interaction, each actor can have only limited number of relationships with other actors. Hence, we assume that the degree of each actor cannot be greater than V. In addition, an actor can influence his follower in a social network. However, the power of influence will decrease with distance. Thus, we assume that the depth of the social network is D. Then, the problem is to compute how large a social network can be (i.e. the maximum number of actors) with the given degree and depth constraints.

Input Format

First line of the input contains a positive integer $T(T \leq 150)$. Each of the following T lines contains two integers $D(0 \leq D \leq 2 \times 10^9)$ and $V(1 \leq V \leq 2 \times 10^9)$, respectively.

Output Format

For each case, print a line of the form "Case < x >: < y >", where x is the case number and y is the maximum possible number of actors in the social network. As the value of y can be quite large, print the value modulo $1000000007(10^9 + 7)$. If it is not possible to construct the network with depth D, print "Case < x >: -1".

Sample Input:	Sample Output:
3 0 1 1 2 1 5	Case 1: 1 Case 2: 3 Case 3: 6

Problem 3. Even or Odd

(Time Limit: 1 second)

Problem Description

A mathematician loves challenge of solving difficult puzzle. One day, he is given a set of n integral variables $(x_1, x_2, \ldots, x_n)$. His student challenged him by randomly pairing two variables and place one of the following four operators between them: add (+), subtract (-), multiplication (*), or power (^), creating m such pairs of evaluations. He is asked if there exist positive integral solutions of those n variables such that the multiplication of all pairs of evaluations is odd. For instance, given two pairs of evaluations: $(x_1 + x_2) * (x_1 - x_3)$. The mathematician should answer "Yes" since he can assign $2,1,3$ to x_1, x_2, x_3 and lead to multiplication of -3, which is an odd number. On the other hand, if given $(x_1 + x_2) * (x_1 * x_2)$, he should answer "No", since there is no integral solutions such that the multiplication of entire expression can yield an odd number. As a final instance, $(x_1 + x_2) * (x_1 \hat{} x_2)$ can produce an odd number (i.e., $(1 + 2) \times 1^2 = 3$) if setting $x_1 = 1$ and $x_2 = 2$. Given expressions of such integral evaluations, write a program to help the mathematician to answer if the entire expression can yield an odd number.

Technical Specification

- The number of paired evaluations *m* ranges from 1 to 10000.
- The number of variables *n* ranges from 1 to 10000.
- The operator within each paired evaluation is add (+), subtract (-), multiplication (*), or power (^) of the two integral variables.
- The number of test cases is below 60.

Input Format

The first line contains the number of test cases. Each of the following two lines represents a test case. For each test case, the starting line stores the numbers of variables (n) and of evaluations (m) separated by a white space. The next line stores the paired evaluations separated by a white space. For each evaluation, there are two integers representing the i-th and j-th variable. Between the two integers, there is an operator representing add (+), subtract (-), multiplication (*), and power (^). Note that the two integers and operator are also separated by a white space.

Output Format

The output should consist of one line for each test case. Each line contains "Yes" if there exists a positive integral solution such that the entire expression can yield an odd number. Otherwise, output "No" if there are no such solutions existed. The first test case shown below (i.e., "1 + 2 1 - 3") represents $(x_1 + x_2) * (x_1 - x_3)$. The second test case (i.e., "1 + 2 1 * 2") represents $(x_1 + x_2) * (x_1 * x_2)$. The last case (i.e., "1 + 2 1 ^ 2") is $(x_1 + x_2) * (x_1 \hat{} x_2)$.

Example

Sample Input:	**Sample Output:**
3 3 2 1 + 2 1 - 3 2 2 1 + 2 1 * 2 2 2 1 + 2 1 ^ 2	Yes No Yes

Problem 4. Recycling Boxes

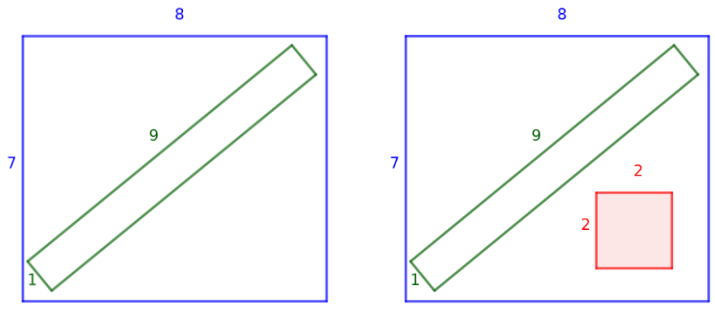
(Time Limit: 3 seconds)

Problem Description

Howard had ordered a lot of goods from Nomaza two days ago. Nomaza put the items into n paper boxes (numbered from 1 to n) and delivered to his apartment yesterday. Since he really bought many things, he spent whole day to open the boxes of all the goods. And now, he has a trouble to dispose the boxes. Assume that the length, the width, and the height of box i are l_i, w_i and h_i, respectively. According to the law of his country, the government recycles paper boxes only if they are stacked properly. Howard has to divide n paper boxes into several stacks without violating the following rules.

1. For each box, at most one box can be put on its top directly.
2. Only the bottom faces of boxes may be stacked on the top faces of boxes.
3. The top face of the box below must fully cover the bottom face of the box atop.
4. Rotating a box without changing the top face is allowed. That is, Howard may ignore the heights of the boxes.

For example, Howard needs to dispose 3 boxes, and $l_1 = 1$, $w_1 = 9$, $l_2 = 7$, $w_2 = 8$, $l_3 = 2$, $w_3 = 2$. He may stack box 1 onto box 2, since he can rotate box 1 like Figure 1 (a). However, he cannot stack both box 1 and box 3 onto box 2 like Figure 1 (b) which violates rule 1.



(a) (b)

Figure 1

There is a simple way to dispose all n boxes: just divide them into n stacks of a single box. But Howard wants to produce as less as possible stacks. Please write a

program to help Howard.

Technical Specification

- T, the number of test cases, is at most 25.
- n, the number of paper boxes, is at most 10^3.
- $l_1, l_2, \ldots, l_n, w_1, w_2, \ldots w_n$ are positive integers no more than 10^3.

Input Format

The first line of the input file contains an integer T indicating the number of test cases. The first line of each test case contains a positive integer n indicating the number of paper boxes. The $(i+1)$-th line of each test case containing two positive integers l_i and w_i indicating the length and the width of box i, respectively.

Output Format

For each test case, please output the minimum number of proper stacks of boxes.

Example

Sample Input:	Sample Output:
2 3 1 9 7 8 2 2 3 2 5 3 6 7 4	2 1

Problem 5. Divisions

(Time Limit: 3 seconds)

Problem Description

Paul is the king of a country called Holy-wood. There are *n* cities in Holy-wood, which are connected by a network of $n-1$ roads. Each road is bidirectional, connects two different cities, and has a positive integer length. Furthermore, there is exactly one possible path connecting any pair of cities. That is, there is exactly one way to travel from one city to another city by a sequence of roads without visiting any city twice. In addition, these $n-1$ roads form a simple line. That is, each city is connected to at most two roads.

There is a considerable population in each city of Holy-wood. Therefore, it is very difficult to manage every city by king Paul alone. To improve the efficiency of management, Paul is planning to divide the cities of Holy-wood into several regions, each of which will be managed by a local administrator, according to the following conditions:

(1) The cities of each region R are *connected*. That is, the cities on the path connecting any two cities in R must also be contained in R.
(2) Each city belongs to exactly one region.
(3) The total population of any region is at least L.

To make a local administrator manage a region simpler, king Paul hopes to have as many regions as possible. Two regions R_1 and R_2 are *adjacent* if and only if there is a road connecting a city in R_1 and a city in R_2. King Paul also hopes that the distance between any two adjacent regions should not be too large. More specifically, king Paul wishes to maximize the number of regions; and in case there are several ways to maximize the number of regions, he wants the way that minimizes the total distance between adjacent regions. Your task is to help king Paul to compute the maximum number of regions along with the minimum total distance between adjacent regions.

Figure 1(a) shows an example, in which $L = 9$ and there are seven cities and six roads. Each node represents a city with a number inside indicating its population; and each edge represents a road. Figure 1(b) shows a feasible way to divide the cities into regions. The number of regions is three, where the populations are 14, 12, and 12

respectively. The total distance between adjacent regions is $2 + 4 = 6$. Figure 1(c) shows another feasible way, which is the best way to divide the cities. The number of regions is still three, but the total distance between adjacent regions is only $2 + 1 = 3$.

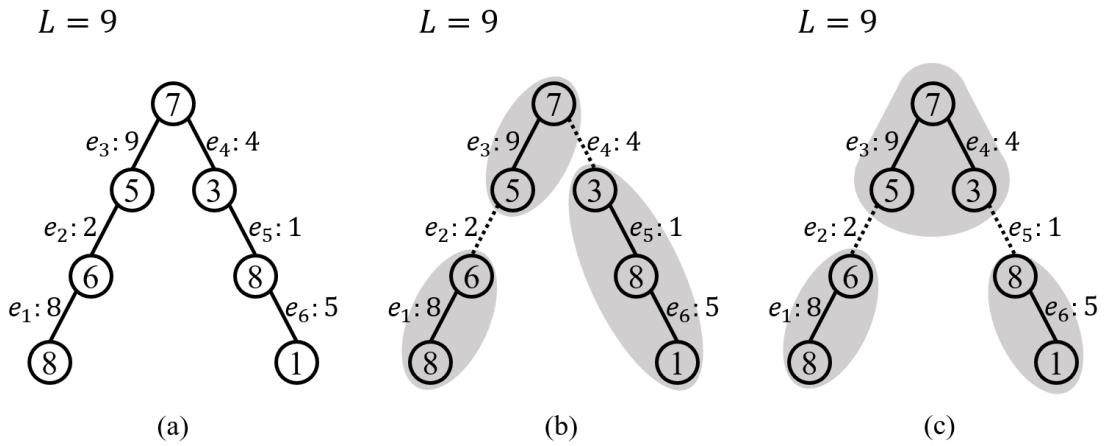


Figure 1. An example.

Technical Specification

- There are at most 10 test cases.
- The number, n, of cities is an integer between 1 and 10^4.
- The population of each city is an integer between 1 and 1000.
- The length of each road is an integer between 1 and 1000.
- The minimum population, L, of a region is an integer between 1 and 100.

Input Format

The first line of the input file contains an integer indicating the number of test cases. Each test case begins with a line containing two integers n and L, where $1 \leq n \leq 10^4$ and $1 \leq L \leq 100$. The integer n indicates the number of cities; and the integer L is the minimum population of a region. Then, n lines follow, each containing an integer p_i, $1 \leq p_i \leq 1000$, indicating the population in city i, $1 \leq i \leq n$. Then, $n-1$ lines follow, each containing three integers c_1, c_2, d, $1 \leq c_1, c_2 \leq n$, $1 \leq d \leq 1000$, indicating there is a road of length d connecting cities c_1 and c_2.

Output Format

For each test case, print a line containing two integers k and S. The integer k indicates the maximum number of regions; and S indicates, when the cities are divided

to k regions, the minimum total distance between adjacent regions. If there is no feasible way to divide the cities, print '-1' instead of two integers k and S.

Example

Sample Input:	Sample Output:
3 7 9 7 5 3 6 8 8 1 1 2 9 2 4 2 4 6 8 7 5 5 5 3 1 3 1 4 1 2 1 1 1 1	3 3 -1 1 0