A. Party Time!

2 seconds, 512 megabytes

Your favourite ADA TA is back with yet another challenge. His birthday is quite near, so he is planning to throw a birthday party for all the ADA students. For the party, he has stored some special juice (yes, only juice) in N containers of different capacities y_i in his room. Each container already contains some x_i juice. As your TA has a small room (just like you), he wants to minimize the number of containers used to store the special juice.

In order to minimize the number of containers, you need to pour the special juice from one container to another optimally to minimize the number of containers used (Only containers with non-zero amounts would be considered as used). Let's say if you pour z amount of special juice from container A to container B, then your effort score will increase by z amount.

Hence, the challenge for you is to minimize the number of containers as well as minimize the effort score e to transfer the special juice from N containers to only c container(s) (c is the minimum number of container required to store the special juice) to help your favourite TA organize his birthday party nicely.

Note: x_i is always less than or equal to y_i, meaning the amount of juice stored will always be less than or equal to the capacity of the container.

Input

The first line contains positive integer $N(1 \leq N \leq 100)$ — the number of containers. Next line has N integers $x_1, x_2, \ldots x_n$ such that $(1 \leq x_i \leq 100)$ Next line has N integers $y_1, y_2, \ldots y_n$ such that $(1 \leq y_i \leq 100)$

It is provided that $(1 \leq x_i \leq y_i \leq 100)$.

Output

In a single line, output c (minimum number of containers used) and e (the minimum effort required to transfer special juice to the minimum number of containers).

input

```
5
3 4 10 6 7
10 4 11 8 9
```

output

```
3 10
```

In the given test case, a minimum of 3 containers is required to store the juice. This is evident as containers with capacities of 10, 11, and 9 are sufficient to hold all the juice present in the room. The minimum effort score would be 10, as this represents the least amount of effort needed to transfer the juice from the discarded containers into the 3 selected containers that still have some available space. To elaborate, one possible solution involves transferring all the juice from container 2 into container 1, while distributing the juice from container 4 among selected containers 1, 3, and 5 based on their capacities.

B. The Teleportation Machine

1 second, 512 megabytes

Tony Stark has created another breakthrough. He has created a teleportation machine which you can use to teleport anywhere. The machine has 2 modes (1 and 2). Mode 1 means that the machine is not ready for teleportation and is doing some computation and mode 2 means it is ready. The machine automatically alternates between modes 1 and 2 without any human intervention. You can use the following steps to use the machine:-

- If the machine is in mode 1 it will output a number x . You need to store this number with you (maybe append it to an array).
- If the machine is in mode 2, it will again show a number x. Now you have to enter the details of where you want to go and along with it a special key. This key is calculated as the total count of subsequences in your array having bitwise OR equal to x .

At any time, the array with you is nothing but the list of numbers generated by the machine in mode 1 till that time. In this problem, you basically need to give the special key every time the machine is in mode 2.

A subsequence can be obtained from a sequence of numbers by removing some or all elements of the sequence.

Input

The first line contains the number of queries Q $(1 \leq Q \leq 10^5)$. This is the total number of mode changes done by the machine.

The following Q lines contain 2 integers, M and x, the type of mode and corresponding integer $x(0 \leq x \leq 1023)$.

If $M = 1$, then the machine outputs the number x.

If $M = 2$, the machine asks you to find the special key by couting the number of valid subsequences where bitwise OR of all the numbers is x.

Output

For every query of second type, output the count of all valid subsequences possible, modulo $10^9 + 7$.

input

```
6
2 0
2 1
1 2
1 2
2 2
2 1000
```

output

```
1
0
3
0
```

input

```
7
1 7
1 8
1 9
1 6
1 4
1 11
2 15
```

output

```
44
```

In the first testcase, initially the array is empty. Since OR of 0 is 0, we have exactly 1 way to achieve 0, and 0 ways to achieve 1.

The machine then proceeds to output 2 twice. Now there are 3 ways to reach 2 (selected numbers are indicated in red) - $[2, 2]$, $[2, 2]$, $[2, 2]$

C. Artist's Dilemma

3 seconds, 256 megabytes

~~You are the sole survivor of the dungeon of strings. Now, since ...~~

Ok. Let us snap back to reality for once. Simon is your best friend and hence you are preparing a gift for him. You have decided to give him a painting as he loves art, but you only know data structures and algorithms. So, you decided to take upon your favourite data structure(graph) and draw some mysterious animal using it.

The animal will just have a face and many hands (Not creepy at all). Given this, the beauty of a graph is described as the product of the number of **vertices** used for making the face and the number of hands (one hand is an edge). Now, the condition for an edge to be a part of the face is as follows:

- The number marked on the vertices in the face must increase strictly as you move forward in one direction.
- The face can't have multiple connected components. This means that the vertices that are a part of the face will have to be connected.

For the hands, you decide that a hand will be a single edge such that one of the vertices in the edge is an **endpoint** for the face.

Now, what is the maximum possible beauty of the art?

Note: It is given that two distinct edges don't connect the same pair of points. Also, there is no edge from a vertex to itself. Assume you can take an edge in both face and hand if it makes the art more beautiful.

Hint Check the test case below and try making your own test cases as well.

Input

In the first line, you are given $n(1 \leq n \leq 131072)$ and $m(1 \leq m \leq 300000)$ where n is the number of vertices and m is the number of edges that are present in the graph. The following m lines contain two integers x and y such that the vertices marked x and y have an edge between them. $(1 \leq x, y \leq n)$.

Output

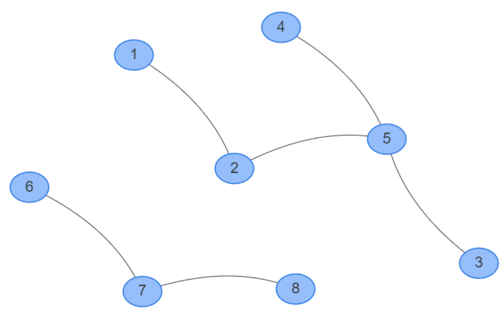
You must output the maximum amount of beauty that can be achieved using the given graph.

input

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

output

```
9
```



Graph for above testcase

In this test case, we can have the following vertices in the face: $1, 2, 5$

This will make the number of vertices in the face = 3.

Now, we can have the following edges as the hands: $(2, 5), (3, 5), (4, 5)$

This gives us the total beauty equal to the product of 3 and 3.

D. The Cooper-Hofstadter Realization

1 second, 256 megabytes

Sheldon Cooper and Leonard Hofstadter are two brilliant Physicists. They are trying to solve string theory and Sheldon finally has a way to solve a part of the problem. So the entire world has been condensed in the form of a directed graph with N vertices and M edges. In the graph, edge i connects two different vertices U_i and V_i with a length of W_i which represents that it takes W_i seconds to move from U_i to V_i. Sheldon is currently standing on vertex 1 and Leonard can currently be standing on any vertex from 2 to N (call his position k) . Now. each node of the graph has some knowledge which the person standing there can acquire.

Also, it is known that Sheldon and Leonard have traveled to various nodes and accumulated a lot of extra knowledge in addition to their original aptitude, and their positions described above are their current positions(i.e. after getting all the knowledge from different nodes).

Now Sheldon believes that in order to solve the problem the two of them need to meet, and in the shortest time possible. Also, he believes (like always) that he is the more important and intelligent member of the two and hence it doesn't matter where Leonard stands currently (which is why his position is variable and Sheldon's is fixed).

So for all $2 \leq k \leq N$ (all possible positions of Leonard), Sheldon wants to know the minimum time in seconds required for them to meet so that he can ask Leonard to modify his journey accordingly (such that Leonard is currently standing from where he can reach Sheldon in minimum time and the problem is solved !!)

Input

The first line contains two integers N and M $(2 \leq N \leq 10^5, 0 \leq M \leq 2 \cdot 10^5)$ — the number of vertices and edges in the graph.

The i^{th} of the next M lines contains three integers U_i, V_i, and W_i $(1 \leq U_i, V \leq N, 1 \leq W_i \leq 10^9)$ — a directed edge that connects two different vertices U_i and V_i with a length of W_i.

Output

Output a line containing $N - 1$ integers. The j^th integer represents the minimum time in seconds needed by the two of them to meet if initially Sheldon is standing on vertex 1 and Leonard on vertex $j + 1$. Print -1 if it is impossible for them to meet according to the current configuration.

input

```
5 7
2 3 2
3 5 1
5 2 4
3 1 3
1 5 1
1 3 4
3 2 1
```

output

```
4 2 -1 1
```

If initially Sheldon is on vertex 1 and Leonard is on vertex 2, they can do the following moves:

- Sheldon moves to vertex 5 in 1 second.
- Leonard moves to vertex 3 in 2 seconds.
- Leonard then moves to vertex 5 in 1 second.

In total, they need $1 + 2 + 1 = 4$ seconds. It can be proven that there is no other way that is faster.

E. RIISE ONCE AGAIN

4 seconds, 512 megabytes

Just a few days ago, we completed RIISE'24, the annual flagship event of IIIT Delhi, where many high-profile individuals from industry and academia came together. Suppose you are the Events head for next year's RIISE'25. All the professors have given you a long list of tasks that need to be completed in N different rooms of our large campus connected by M one-way connections (yes, a directed graph, you are smart). You have called and assembled the entire team in Room 1. Before dispersing the team to different rooms, it is important to estimate the number of paths from Room 1 to all possible N rooms, as the distribution of work depends on the number of paths to each room from Room 1.

A path from any Room X to any Room Y is a sequence of connections where Room X should be at the start of the first connection of the path, and Room Y is at the end of the last connection of the path. The following connection starts at the Room where the previous connection ends within the path. It is essential to keep in mind that it is possible that there would not be any path from Room 1 to any Room X, or there can be exactly 1 path from Room 1 to any Room X, or there can be more than 1 path from Room 1 to Room X, or there can be an infinite number of paths from 1 to Room X.

If there is no path from Room 1 to any Room X, then return 0 for that room. If there is exactly 1 path from Room 1 to any Room X, then return 1 for that room. If there are more than 1 but a finite number of paths from Room 1 to any Room X, then return 2 for that room. If there are more than 1 but an infinite number of paths from Room 1 to any Room X, then return -1 for that room. Your task is to output the appropriate integer varies from **-1 to 2** for all N rooms.

Note: Self-loops are possible, meaning connections from Room X to Room X are allowed. Multiple connections between rooms, let's say X and Y, are not possible; hence, at most 1 connection is allowed between any Room X and Room Y.

Input

The first line contains T $(1 \le T \le 10^4)$ test cases.

For each test case t:

The first line contain two integers N and M $(1 \le N \le 4 * 10^5, 0 \le M \le 4 * 10^5)$

The next M lines contain two integers X_i and Y_i, where i varies from 1 to M. $(1 \le X_i, Y_i \le N)$

The sum of N over all the test cases does not exceed $4 * 10^5$, and similarly for M.

Output

Output in T lines.

Each line t contains N integers from -1 to 2.

input

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

output

```
1 0 1 2 -1 -1
```

For Room 1, there is one route only of length (from Room 1 to 1) => 1

For Room 2, there is no way to reach from 1. => 0

For Room 3, there is one route of length 1 (from Room 1 to 3) => 1

For Room 4, there are 2 routes, one is directly from room 1 and other one via room 3 => 2

For Room 5, there are infinite routes due to self loop at room 5. => -1

For Room 6, there are again infinite routes due to self loop at 5. => -1