Gracias Sponsors!

Organizador

UNR Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura FCEIA

Diamond

GTS

Gold

Avature

accenture

Contest 1 - Avanzado

A Range Sorting

You are given an array a, consisting of n distinct integers $a_1, a_2, \ldots, a_n$.

Define the *beauty* of an array $p_1, p_2, \ldots p_k$ as the minimum amount of time needed to sort this array using an arbitrary number of *range-sort* operations. In each range-sort operation, you will do the following:

- Choose two integers l and r ($1 \le l < r \le k$).
- Sort the subarray $p_l, p_{l+1}, \ldots, p_r$ in $r - l$ seconds.

Please calculate the sum of beauty over all subarrays of array a.

A subarray of an array is defined as a sequence of consecutive elements of the array.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^3$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 5 \cdot 10^3$) — the length of the array a.

The second line of each test case consists of n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$). It is guaranteed that all elements of a are pairwise distinct.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^3$.

Output

For each test case, output the sum of beauty over all subarrays of array a.

Example

input

```
5
2
6 4
3
3 10 6
4
4 8 7 2
5
9 8 2 4 6
12
2 6 13 3 15 5 10 8 16 9 11 18
```

output

```
1
2
8
16
232
```

Note

In the first test case:

- The subarray $[6]$ is already sorted, so its beauty is 0.
- The subarray $[4]$ is already sorted, so its beauty is 0.
- You can sort the subarray $[6, 4]$ in one operation by choosing $l = 1$ and $r = 2$. Its beauty is equal to 1.

The sum of beauty over all subarrays of the given array is equal to $0 + 0 + 1 = 1$.
In the second test case:

- The subarray $[3]$ is already sorted, so its beauty is 0.
- The subarray $[10]$ is already sorted, so its beauty is 0.
- The subarray $[6]$ is already sorted, so its beauty is 0.
- The subarray $[3, 10]$ is already sorted, so its beauty is 0.
- You can sort the subarray $[10, 6]$ in one operation by choosing $l = 1$ and $r = 2$. Its beauty is equal to $2 - 1 = 1$.
- You can sort the subarray $[3, 10, 6]$ in one operation by choosing $l = 2$ and $r = 3$. Its beauty is equal to $3 - 2 = 1$.

The sum of beauty over all subarrays of the given array is equal to $0 + 0 + 0 + 0 + 1 + 1 = 2$.

B

In Case of an Invasion, Please...

After Curiosity discovered not just water on Mars, but also an aggressive, bloodthirsty bunch of aliens, the Louvain-la-Neuve municipal government decided to take precautionary measures; they built shelters in order to shelter everyone in the city in the event of an extraterrestial attack.

Several alien-proof shelters have been erected throughout the city, where citizens can weather an alien invasion. However, due to municipal regulations and local building codes the shelters are limited in size. This makes it necessary for the government to assign every citizen a shelter to calmly direct themselves towards in the rare event of a fleet of UFOs blotting out the sun. Conditional on no shelter being assigned more people than it can fit, it is of the utmost importance that the time it takes until everyone has arrived at a shelter is minimized.

We model Louvain-la-Neuve as a network of n locations at which people live, connected by m bidirectional roads. Located at s points throughout the city are the shelters, each with a given maximum capacity. What is the minimum amount of time it takes for everyone to arrive at a shelter, when we assign people to shelters optimally?

The Louvain-la-Neuve municipal government has made sure that there is enough shelter capacity for its citizens and all shelters can be reached from any location, i.e. it is always possible to shelter everyone in some way.

Input

- On the first line are three integers, the number of locations $1 \le n \le 10^5$, roads $0 \le m \le 2 \cdot 10^5$, and shelters $1 \le s \le 10$.
- Then follows a line with n integers $0 \le p_i \le 10^9$, indicating the the number of people living at location $1 \le i \le n$.
- Then follow m lines containing three integers $1 \le u, v \le n$ and $1 \le w \le 10^9$ indicating that there is a bidirectional road connecting u and v that takes w time to traverse. For any two locations there is at most one road connecting them directly, and no road connects a location to itself.
- Finally follow s lines with two integers $1 \le s_i \le n$ and $1 \le c_i \le 10^9$, indicating that there is a shelter with capacity c_i at location s_i.

Output

Print the minimum amount of time it takes to shelter everyone.

Sample Input 1

```
2 1 1
3 2
1 2 4
1 6
```

Sample Output 1

```
4
```

Sample Input 2

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

Sample Output 2

```
5
```

Sample Input 3

```
7 8 3
0 1 1 1 1 0 2
1 2 1
2 3 1
3 1 1
4 6 5
4 3 1
6 7 10
7 5 3
5 6 3
6 5
1 1
2 1
```

Sample Output 3

```
6
```

Sample Input 4

```
2 1 1
0 2
1 2 1000000000
2 2
```

Sample Output 4

```
0
```

C 1D Sokoban

You are playing a game similar to Sokoban on an infinite number line. The game is discrete, so you only consider integer positions on the line.

You start on a position 0. There are n boxes, the i-th box is on a position a_i. All positions of the boxes are distinct. There are also m special positions, the j-th position is b_j. All the special positions are also distinct.

In one move you can go one position to the left or to the right. If there is a box in the direction of your move, then you push the box to the next position in that direction. If the next position is taken by another box, then that box is also pushed to the next position, and so on. **You can't go through the boxes**. **You can't pull the boxes towards you**.

You are allowed to perform any number of moves (possibly, zero). Your goal is to place as many boxes on special positions as possible. Note that some boxes can be initially placed on special positions.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

Then descriptions of t testcases follow.

The first line of each testcase contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of boxes and the number of special positions, respectively.

The second line of each testcase contains n distinct integers in the increasing order $a_1, a_2, \dots, a_n$ ($-10^9 \le a_1 < a_2 < \dots < a_n \le 10^9$; $a_i \ne 0$) — the initial positions of the boxes.

The third line of each testcase contains m distinct integers in the increasing order $b_1, b_2, \dots, b_m$ ($-10^9 \le b_1 < b_2 < \dots < b_m \le 10^9$; $b_i \ne 0$) — the special positions.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$. The sum of m over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase print a single integer — the maximum number of boxes that can be placed on special positions.

Example

input Copy

```
5
5 6
-1 1 5 11 15
-4 -3 -2 6 7 15
2 2
-1 1
-1000000000 1000000000
2 2
-1000000000 1000000000
-1 1
3 5
-1 1 2
-2 -1 1 2 5
2 1
1 2
10
```

output Copy

```
4
2
0
3
1
```

Note

In the first testcase you can go 5 to the right: the box on position 1 gets pushed to position 6 and the box on position 5 gets pushed to position 7. Then you can go 6 to the left to end up on position -1 and push a box to -2. At the end, the boxes are on positions $[-2, 6, 7, 11, 15]$, respectively. Among them positions $[-2, 6, 7, 15]$ are special, thus, the answer is 4.

In the second testcase you can push the box from -1 to -10^9, then the box from 1 to 10^9 and obtain the answer 2.

The third testcase showcases that you are not allowed to pull the boxes, thus, you can't bring them closer to special positions.

In the fourth testcase all the boxes are already on special positions, so you can do nothing and still obtain the answer 3.

In the fifth testcase there are fewer special positions than boxes. You can move either 8 or 9 to the right to have some box on position 10.

D Stressful Training

Berland SU holds yet another training contest for its students today. n students came, each of them brought his laptop. However, it turned out that everyone has forgot their chargers!

Let students be numbered from 1 to n. Laptop of the i-th student has charge a_i at the beginning of the contest and it uses b_i of charge per minute (i.e. if the laptop has c charge at the beginning of some minute, it becomes $c - b_i$ charge at the beginning of the next minute). The whole contest lasts for k minutes.

Polycarp (the coach of Berland SU) decided to buy a **single** charger so that all the students would be able to successfully finish the contest. He buys the charger at the same moment the contest starts.

Polycarp can choose to buy the charger with any non-negative (zero or positive) integer power output. The power output is chosen before the purchase, it can't be changed afterwards. Let the chosen power output be x. **At the beginning of each minute** (from the minute contest starts to the last minute of the contest) he can plug the charger into any of the student's laptops and use it for some **integer** number of minutes. If the laptop is using b_i charge per minute then it will become $b_i - x$ per minute while the charger is plugged in. Negative power usage rate means that the laptop's charge is increasing. The charge of any laptop isn't limited, it can become infinitely large. The charger can be plugged in no more than one laptop at the same time.

The student successfully finishes the contest if the charge of his laptop never is below zero at the beginning of some minute (from the minute contest starts to the last minute of the contest, zero charge is allowed). The charge of the laptop of the minute the contest ends doesn't matter.

Help Polycarp to determine the minimal possible power output the charger should have so that all the students are able to successfully finish the contest. Also report if no such charger exists.

Input

The first line contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 2 \cdot 10^5$) — the number of students (and laptops, correspondigly) and the duration of the contest in minutes.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^{12}$) — the initial charge of each student's laptop.

The third line contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 10^7$) — the power usage of each student's laptop.

Output

Print a single non-negative integer — the minimal possible power output the charger should have so that all the students are able to successfully finish the contest.

If no such charger exists, print `-1`.

Examples

input Copy

```
2 4
3 2
4 2
```

output Copy

```
5
```

input Copy

```
1 5
4
2
```

output

```
1
```

input

```
1 6
4
2
```

output

```
2
```

input

```
2 2
2 10
3 15
```

output

```
-1
```

Note

Let's take a look at the state of laptops in the beginning of each minute on the first example with the charger of power 5:

1. charge: $[3, 2]$, plug the charger into laptop 1;
2. charge: $[3 - 4 + 5, 2 - 2] = [4, 0]$, plug the charger into laptop 2;
3. charge: $[4 - 4, 0 - 2 + 5] = [0, 3]$, plug the charger into laptop 1;
4. charge: $[0 - 4 + 5, 3 - 2] = [1, 1]$.

The contest ends after the fourth minute.

However, let's consider the charger of power 4:

1. charge: $[3, 2]$, plug the charger into laptop 1;
2. charge: $[3 - 4 + 4, 2 - 2] = [3, 0]$, plug the charger into laptop 2;
3. charge: $[3 - 4, 0 - 2 + 4] = [-1, 2]$, the first laptop has negative charge, thus, the first student doesn't finish the contest.

In the fourth example no matter how powerful the charger is, one of the students won't finish the contest.

E Number Deletion Game

Alice and Bob play a game. They have a set that initially consists of n integers. The game is played in k turns. During each turn, the following events happen:

1. firstly, Alice chooses an integer from the set. She can choose any integer **except for the maximum one**. Let the integer chosen by Alice be a;
2. secondly, Bob chooses an integer from the set. He can choose any integer **that is greater than a**. Let the integer chosen by Bob be b;
3. finally, both a and b are erased from the set, and the value of $b - a$ is added to the score of the game.

Initially, the score is 0. Alice wants to maximize the resulting score, Bob wants to minimize it. Assuming that both Alice and Bob play optimally, calculate the resulting score of the game.

Input

The first line contains two integers n and k ($2 \le n \le 400$, $1 \le k \le \lfloor \frac{n}{2} \rfloor$) — the initial size of the set and the number of turns in the game.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$) — the initial contents of the set. These integers are pairwise distinct.

Output

Print one integer — the resulting score of the game (assuming that both Alice and Bob play optimally).

Examples

input Copy

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

output Copy

```
4
```

input Copy

```
7 3
101 108 200 1 201 109 100
```

output Copy

```
283
```

F Loppinha the boy who likes sopinha

Loppinha loves to go to the gym with his friends. He takes his training sessions very seriously, and he always follows his schedule very strictly. He just created a new plan where he has set exactly what he is going to do at every single one of the N minutes of the session.

Loppinha loves having a sopinha (small soup) before training. The soup contains K grams of protein. As you can imagine, he needs that protein to be able to endure his very tough exercises. He is burning protein at every minute he is working out, and the amount of protein he burns in a minute depends on how many minutes he has been working out without a minute of rest. If he has trained for p minutes without resting, in the $(p+1)$-th minute of workout he is going to burn $p+1$ grams of protein.

Of course, if he doesn't have enough protein at any moment he dies. For example, if he has 3 grams of protein at a moment and at that minute his workout requires 4, if he does the workout he dies. Given his schedule and the amount of protein K he has before starting the training session, Loppinha, who is willing to change some minutes of his workout, wants to know what is the minimum amount of minutes he has to change from working out to resting so he does not die.

Input

The first line of the input contains two integers N $(1 \le N \le 450)$ and K $(1 \le K \le 10^7)$, indicating the number of minutes in Loppinha's training session and the amount of proteins in his soup. The next line contains a string with N characters, either 0 or 1, where 0 indicates a minute of rest and a 1 indicates a minute of workout.

Output

Output a single integer - the minimum number of minutes Loppinha has to switch from workout to rest so that he can finish his exercises without dying.

Examples

input Copy
```
4 2
1101
```
output Copy
```
1
```

input Copy
```
10 5
1101100111
```
output Copy
```
3
```

input Copy
```
3 1
111
```
output Copy
```
2
```

Note

In the first test case, if he changes the second minute to rest, he will consume exactly 2 grams of protein. Notice that if he changes the last minute to rest instead, he would need 3 grams of protein to complete his workout session without dying.

G Makoto and a Blackboard

Makoto has a big blackboard with a positive integer n written on it. He will perform the following action exactly k times:

Suppose the number currently written on the blackboard is v. He will randomly pick one of the divisors of v (possibly 1 and v) and replace v with this divisor. As Makoto uses his famous random number generator (RNG) and as he always uses 58 as his generator seed, each divisor is guaranteed to be chosen with equal probability.

He now wonders what is the expected value of the number written on the blackboard after k steps.

It can be shown that this value can be represented as $\frac{P}{Q}$ where P and Q are coprime integers and $Q \not\equiv 0 \pmod{10^9+7}$. Print the value of $P \cdot Q^{-1}$ modulo 10^9+7.

Input

The only line of the input contains two integers n and k ($1 \leq n \leq 10^{15}$, $1 \leq k \leq 10^4$).

Output

Print a single integer — the expected value of the number on the blackboard after k steps as $P \cdot Q^{-1} \pmod{10^9+7}$ for P, Q defined above.

Examples

input

```
6 1
```

output

```
3
```

input

```
6 2
```

output

```
875000008
```

input

```
60 5
```

output

```
237178099
```

Note

In the first example, after one step, the number written on the blackboard is 1, 2, 3 or 6 — each occurring with equal probability. Hence, the answer is $\frac{1+2+3+6}{4} = 3$.

In the second example, the answer is equal to $1 \cdot \frac{9}{16} + 2 \cdot \frac{3}{16} + 3 \cdot \frac{3}{16} + 6 \cdot \frac{1}{16} = \frac{15}{8}$.

H Security System

Fox Ciel safely returned to her castle, but there was something wrong with the security system of the castle: sensors attached in the castle were covering her.

Ciel is at point $(1, 1)$ of the castle now, and wants to move to point (n, n), which is the position of her room. By one step, Ciel can move from point (x, y) to either $(x + 1, y)$ (rightward) or $(x, y + 1)$ (upward).

In her castle, c^2 sensors are set at points $(a + i, b + j)$ (for every integer i and j such that: $0 \le i < c, 0 \le j < c$).

Each sensor has a count value and decreases its count value every time Ciel moves. Initially, the count value of each sensor is t. Every time Ciel moves to point (x, y), the count value of a sensor at point (u, v) decreases by ($|u - x| + |v - y|$). When the count value of some sensor becomes **strictly less than** 0, the sensor will catch Ciel as a suspicious individual!

Determine whether Ciel can move from $(1, 1)$ to (n, n) without being caught by a sensor, and if it is possible, output her steps. Assume that Ciel can move to every point even if there is a censor on the point.

Input

In the first line there are five integers n, t, a, b, c ($2 \le n \le 2\cdot 10^5$, $0 \le t \le 10^{14}$, $1 \le a \le n - c + 1$, $1 \le b \le n - c + 1$, $1 \le c \le n$).

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use the `cin` stream (also you may use the `%I64d` specificator).

Output

If Ciel's objective is possible, output in first line $2n - 2$ characters that represent her feasible steps, where i-th character is R if i-th step is moving rightward, or U if moving upward. If there are several solution, output **lexicographically first** one. Character R is lexicographically earlier than the character U.

If her objective is impossible, output `Impossible`.

Examples

input Copy
```
5 25 2 4 1
```
output Copy
```
RRUURURU
```

input Copy
```
3 6 1 2 2
```
output Copy
```
URUR
```

input Copy
```
3 5 1 2 2
```
output Copy
```
Impossible
```

input Copy

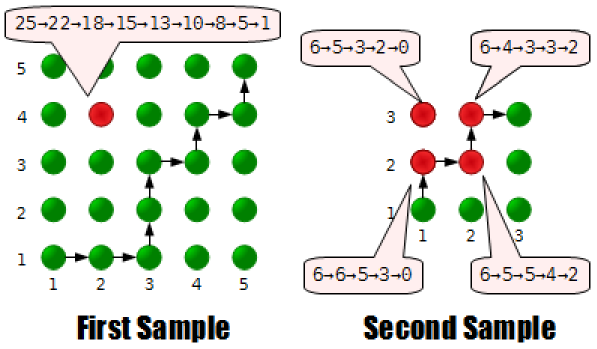
```
20 492 11 4 8
```

output

```
RRRRRRRRRRRRRRRRRUUUUURUUUUURRUUUUUUUUU
```

Note

The answers for the first sample and the second sample are shown on the picture:



Here, a red point represents a point that contains a sensor.

I Jumbo World

Inspired by a game “Small World” by “Days of Wonder” Peter decided to create his own alternative. He called it “Jumbo World”.

The game proceeds on a board consisting of several areas. Some areas are adjacent to each other. Each area has at most 5 adjacent areas. Some areas are adjacent to the edge of the board, these areas are called *border areas*. Each area has some terrain type that can give various bonuses to players.

Several players take part in the game, each controlling one *race*. Different players control different races. Each race has several *units* that can occupy various areas, each area can be occupied by several units of the same race, but units of two different races cannot occupy the same area.

Players move in turns, a turn consists of conquering some areas. Initially a player can conquer any border area. After a player has conquered border area she can continue conquering areas adjacent to other areas she has conquered.

A player has several units of his race. Conquering an area requires $\max(1, 2 + c + b)$ units where c is the number of units of other race occupying the area and b is a bonus value depending on terrain type, conquering race and conqured race. Units conquering an area stay there occupying it until the end of the turn.

If the player doesn’t have enough units to conquer any area adjacent to her areas, her turn is over and she gets points for her areas. Each area by default gives 1 point, this value can be altered by bonuses, also bonuses can alter points depending on conquests.

Races have different abilities that alter conquest bonus value b and points received in the end of the turn. The following abilities are possible.

Ability	**Description**
`+`k `defence`	If this race’s area is being conqured, k more units are needed (b += k).
`+`k `defence on` T	Here T is terrain type. If this race’s area with terrain type T is being conqured, k more units are needed (b += k).
`+`k `defence near` T	Here T is terrain type. If this race’s area which has adjacent area with terrain type T is being conqured, k more units are needed (b += k).
`+`k `defence versus` R	Here R is a race. If this race’s area is being conqured by a race R, k more units are needed (b += k).
`+`k `attack`	If this race conquers some area, k less units are needed (b −= k).
`+`k `attack on` T	Here T is terrain type. If this race conquers area with terrain type T, k less units are needed (b −= k).
`+`k `attack near` T	Here T is terrain type. If this race conquers area which has adjacent area with terrain type T, k less units are needed (b −= k).
`+`k `attack versus` R	Here R is a race. If this race conquers area which is occupied by a race R, k less units are needed (b −= k).
`+`k `points`	This race gets k additional points in the end of the turn for each area it occupies.
`+`k `points on` T	Here T is terrain type. This race gets k additional points in the end of the turn for each area of terrain type T it occupies.
`+`k `points for conquest`	This race gets k additional points in the end of the turn for each area conquered that contained at least one unit of other race ($c > 0$).

For all abilities $1 \leqslant k \leqslant 9$.

You are planning to make a turn in "Jumbo World" and have n units of a given race. None of your units occupies any areas, but some areas may be already occupied by other players' units. Find out what is the maximal number of points you can get in the end of this turn.

Input

The first line of the input file contains four integers: t, r, a and n — number of terrain types, races, areas and units of your race, respectively ($1 \leqslant t \leqslant 10$, $2 \leqslant r \leqslant 10$, $1 \leqslant a \leqslant 30$, $1 \leqslant n \leqslant 10$).

The following t lines contain one word composed of English lowercase letters each and specify terrain types.

Descriptions of r races follow. Each description starts with a line containing the name of the race followed by a line containing v_i — number of its abilities, followed by v_i lines describing abilities as specified above. All race names are single words composed of lowercase letters of English alphabet. Any type of ability for any terrain/race is listed at most once.

Description of areas follows. Areas are numbered from 1 to a, each area is described by four lines. The first line contains its terrain type. The second line contains a word "`border`" if it's border, or "`interior`" if it's not border. The third line contains the name of the race that initially occupies this area followed by number of units that occupy it, or a word "`empty`" if the area is empty. No race is named "`empty`". The fourth line starts with b_i — the number of areas adjacent to it, followed by their numbers. No area is adjacent to itself. If area x is adjacent to area y, then area y is adjacent to area x. It is possible to get from any area to any other area by moving between adjacent areas. There is at least one border area. No area is occupied by more than 10 units.

The following line contains the name of the race that you own.

Output

Output one number: the maximal number of points you can get.

Examples

jumbo.in	jumbo.out
3 3 6 8	5
grass	
hills	
mountains	
dwarves	
1	
+2 points on mountains	
elves	
3	
+1 attack on grass	
+1 defence on grass	
+1 points for conquest	
human	
2	
+1 points on grass	
+1 attack versus dwarves	
mountains	
border	
dwarves 1	
4 2 3 4 5	
grass	
border	
elves 2	
4 1 3 5 6	
hills	
border	
elves 1	
4 1 2 4 6	
hills	
interior	
empty	
4 1 3 5 6	
hills	
interior	
empty	
4 1 2 4 6	
grass	
interior	
empty	
4 2 3 4 5	
human	

You can get 5 points by conquering area 1 with $2 + 1 - 1 = 2$ units and then conquering areas 4, 5 and 6 with 2 units each.

J Tenzing and His Animal Friends

Tell a story about me and my animal friends.

Tenzing has n animal friends. He numbers them from 1 to n.

One day Tenzing wants to play with his animal friends. To do so, Tenzing will host several games.

In one game, he will choose a set S which is a subset of $\{1, 2, 3, \ldots, n\}$ and choose an integer t. Then, he will play the game with the animals in S for t minutes.

But there are some restrictions:

1. Tenzing loves friend 1 very much, so 1 must be an element of S.
2. Tenzing doesn't like friend n, so n must not be an element of S.
3. There are m additional restrictions. The i-th special restriction is described by integers u_i, v_i and y_i, suppose x is the **total** time that **exactly** one of u_i and v_i is playing with Tenzing. Tenzing must ensure that x is less or equal to y_i. Otherwise, there will be unhappiness.

Tenzing wants to know the maximum total time that he can play with his animal friends. Please find out the maximum total time that Tenzing can play with his animal friends and a way to organize the games that achieves this maximum total time, or report that he can play with his animal friends for an infinite amount of time. Also, Tenzing does not want to host so many games, so he will host at most n^2 games.

Input

The first line of input contains two integers n and m ($2 \le n \le 100$, $0 \le m \le \frac{n(n-1)}{2}$) — the number of animal friends and the number of special restrictions.

The i-th of the following m lines of input contains three integers u_i, v_i and y_i ($1 \le u_i < v_i \le n$, $0 \le y_i \le 10^9$) — describing the i-th special restriction. It is guaranteed that for $1 \le i < j \le m$, $(u_i, v_i) \ne (u_j, v_j)$.

Output

If Tenzing can play with his animal friends for an infinite amount of time, output "`inf`". (Output without quotes.)

Otherwise, in the first line, output the total time T ($0 \le t \le 10^{18}$) and the number of games k ($0 \le k \le n^2$).

In the following k lines of output, output a binary string s of length n and an integer t ($0 \le t \le 10^{18}$) — representing the set S and the number of minutes this game will be played. If $s_i = 1$, then $i \in S$, otherwise if $s_i = 0$, then $i \notin S$.

Under the constraints of this problem, it can be proven that if Tenzing can only play with his friends for a finite amount of time, then he can only play with them for at most 10^{18} minutes.

Examples

input

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

output

```
4 4
10000 1
10010 1
```

```
10100 1
11110 1
```

input Copy

```
3 0
```

output Copy

```
inf
```

Note

In the first test case:

1. Tenzing will host a game with friend $\{1\}$ for 1 minute.
2. Tenzing will host a game with friends $\{1, 4\}$ for 1 minute.
3. Tenzing will host a game with friends $\{1, 3\}$ for 1 minute.
4. Tenzing will host a game with friends $\{1, 2, 3, 4\}$ for 1 minute.

If after that, Tenzing host another game with friends $\{1, 2\}$ for 1 minute. Then the time of exactly one of friends 2 or 3 with Tenzing will becomes 2 minutes which will not satisfy the 3-rd special restriction.

In the second test case, there is no special restrictions. So Tenzing can host a game with friend $\{1\}$ for an infinite amount of time.

K

Twenty Four, Again

Yes, we know ... we've used Challenge 24 before for contest problems. In case you've never heard of Challenge 24 (or have a very short memory) the object of the game is to take 4 given numbers (the *base values*) and determine if there is a way to produce the value 24 from them using the four basic arithmetic operations (and parentheses if needed). For example, given the four base values `3 5 5 2`, you can produce 24 in many ways. Two of them are: `5*5-3+2` and `(3+5)*(5-2)`. Recall that multiplication and division have precedence over addition and subtraction, and that equal-precedence operators are evaluated left-to-right.

This is all very familiar to most of you, but what you probably don't know is that you can *grade* the quality of the expressions used to produce 24. In fact, we're sure you don't know this since we've just made it up. Here's how it works: A perfect grade for an expression is 0. Each use of parentheses adds one point to the grade. Furthermore, each inversion (that is, a swap of two adjacent values) of the original ordering of the four base values adds two points. The first expression above has a grade of 4, since two inversions are used to move the 3 to the third position. The second expression has a better grade of 2 since it uses no inversions but two sets of parentheses. As a further example, the initial set of four base values `3 6 2 3` could produce an expression of grade 3 — namely `(3+6+3)*2` — but it also has a perfect grade 0 expression — namely `3*6+2*3`. Needless to say, the lower the grade the "better" the expression.

Two additional rules we'll use: 1) you cannot use unary minus in any expression, so you can't take the base values `3 5 5 2` and produce the expression `-3+5*5+2`, and 2) division can only be used if the result is an integer, so you can't take the base values `2 3 4 9` and produce the expression `2/3*4*9`.

Given a sequence of base values, determine the lowest graded expression resulting in the value 24. And by the way, the initial set of base values `3 5 5 2` has a grade 1 expression — can you find it?

Input

Input consists of a single line containing 4 base values. All base values are between 1 and 100, inclusive.

Output

Display the lowest grade possible using the sequence of base values. If it is not possible to produce 24, display `impossible`.

Sample Input 1	Sample Output 1
`3 5 5 2`	`1`

Sample Input 2	Sample Output 2
`1 1 1 1`	`impossible`

L Graph Without Long Directed Paths

You are given a connected undirected graph consisting of n vertices and m edges. There are no self-loops or multiple edges in the given graph.

You have to direct its edges in such a way that the obtained directed graph does not contain any paths of length two or greater (where the length of path is denoted as the number of traversed edges).

Input

The first line contains two integer numbers n and m ($2 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 2 \cdot 10^5$) — the number of vertices and edges, respectively.

The following m lines contain edges: edge i is given as a pair of vertices u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). There are no multiple edges in the given graph, i. e. for each pair (u_i, v_i) there are no other pairs (u_i, v_i) and (v_i, u_i) in the list of edges. It is also guaranteed that the given graph is connected (there is a path between any pair of vertex in the given graph).

Output

If it is impossible to direct edges of the given graph in such a way that the obtained directed graph does not contain paths of length at least two, print "NO" in the first line.

Otherwise print "YES" in the first line, and then print **any** suitable orientation of edges: a binary string (the string consisting only of '0' and '1') of length m. The i-th element of this string should be '0' if the i-th edge of the graph should be directed from u_i to v_i, and '1' otherwise. Edges are numbered in the order they are given in the input.

Example

input Copy

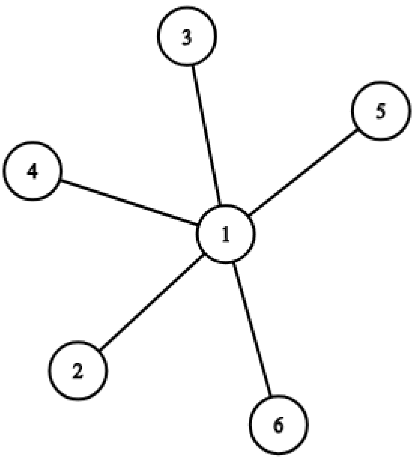
```
6 5
1 5
2 1
1 4
3 1
6 1
```

output Copy

```
YES
10100
```

Note

The picture corresponding to the first example:



And one of possible answers: