Freddy

SEASON 2021/2022 – FORTH ROUND

The fifth-grader Sashka is called "Cringe", because she stays all day long on TikTok. She got sick from the **enormous** amount of videos with Freddy Fazbear and Vanessa, so she decided to refresh herself with the following informatics task:

You're given a string $s_1, s_2, s_3, \dots, s_N$, of N symbols. You can choose a substring of consecutive elements $s_l, s_{l+1}, s_{l+2}, \dots, s_r$ $(1 \leq l \leq r \leq N)$ and „flip" the values of the elements in it. More precisely, for each $l \leq i \leq r$, $s_i \coloneqq 1 - s_i$. You should find the minimum amount of operations necessary to sort the string in ascending order. A string s is sorted in ascending order, when $2 \leq i \leq N$, $s_{i-1} \leq s_i$.

After much torment, she managed to solve the problem elegantly (even her brother was impressed!). Now it is your turn to write the program `freddy.cpp`, which solves the mentioned task.

Input

On the first line of `freddy.in` is given the number N. The second line of the file constains N symbol, the i-th from them is s_i.

Output

On one line in `freddy.out` you should print one number – the minimum operations needed for sorting. If sorting isn't possible, print -1.

Constraints

$1 \leq N \leq 10^5$

$0 \leq s_i \leq 1$

Time Limit: 0.2 sec.
Memory Limit: 256 MB.

Sample testcases

Input (freddy.in)	Output (freddy.out)
5 11111	0
5 10011	1
10 0101101011	3
20 01010001100010011010	6

Matrices

SEASON 2021/2022 – FORTH ROUND

Harry has a very large matrix. To be precise matrix $N \times M$. It was Harry's birthday recently, and (to his surprise) he received a present from his friends - a set of non-negative numbers. Now he plays all day arranging the numbers in the matrix. To make it more interesting, he came up with the following rule: a cell in the i-th row of the j-th column (rows and columns are numbered from 1) cannot contain a number greater than $min(i, j)$.

Since Harry values his friends very much, he wants to make them happy by showing them a **nice matrix**. To him, a **nice matrix** is a matrix in which the sum of the numbers in each row and each column is an even number (yes, Harry is weird). The figure below gives an example of a nice matrix of size 3×4:

0	1	0	1
0	1	1	2
0	2	3	1

Harry, however, is a maximalist and one matrix is not enough for him. He wants to know how many **nice matrices** he can make. Unfortunately, he doesn't have time to do these absurd calculations, so he asks you to help him by finding how many matrices satisfy the condition.

Since, this number can be extremely large, you need to print it modulo $1{,}000{,}000{,}007$ $(10^9 + 7)$.

Input

On the first line of the `matrices.in` file, a positive integer T is entered specifying the number of test cases. The next T lines specify the matrix sizes N and M for the corresponding test.

Output

In the `matrices.out` file, you should print the number of **nice matrices** for each test. The answer for each test should be on a separate line.

Constraints

$1 \leq N, M \leq 10^5$

$1 \leq T \leq 10$

Time limit: 1sec.
Memory limit: 256 MB.

Sample test

Input (matrices.in)	Output(matrices.out)
4	9
2 3	27
4 2	1
1 5	716490517
134 231	

Accounting

SEASON 2021/2022 – FORTH ROUND

After a lot of pressure, Sashka decided to become an accountant. Her job is to process queries on the only array of her company $a_1, a_2, a_3, \dots, a_N$, which consists of N elements. The queries are the following:

`- l r x` – For every element a_i $(l \le i \le r)$ of the array, $a_i \coloneqq x - a_i$

`? l r` – Question for the value of $a_l + a_{l+1} + a_{l+2} + \cdots + a_r$.

Sashka hates her job. Because of that, she decided to create a program, which will process the queries instead of her. Unfortunately, the program she created is too slow. You, as her colleague, are concerned about her. That's why you decided to try writing a program `accounting.cpp`, which will help Sashka.

Input

The first line of the file `accounting.in` contains two integers N and Q, respectively the count of elements in the array and the number of queries. On the next line there are N integers, respectively $a_1, a_2, a_3, \dots, a_N$. Every of the last Q lines describes a query in the format, mentioned above.

Output

For every question query, you should print a number on a new line in `accounting.out`, which is the answer of the query.

Constraints

$1 \le N, Q \le 100\,000$

$0 \le a_i, x \le 10^6$

$1 \le l \le r \le N$

Time Limit: 1.5 sec.
Memory Limit: 256 MB.

Sample Testcases

Input (accounting.in)	Output (accounting.out)
5 7 57 18 40 89 7 - 2 3 94 ? 1 5 ? 3 5 - 2 4 41 ? 2 4 ? 1 5 ? 3 4	283 150 -96 -32 -61
10 10 92 7 84 86 14 3 58 56 19 80 - 2 10 63 ? 3 10 ? 5 8 - 1 6 72 ? 3 6 - 2 8 32 ? 2 3 ? 9 10 ? 3 9 ? 4 8	104 121 223 -45 27 1 18

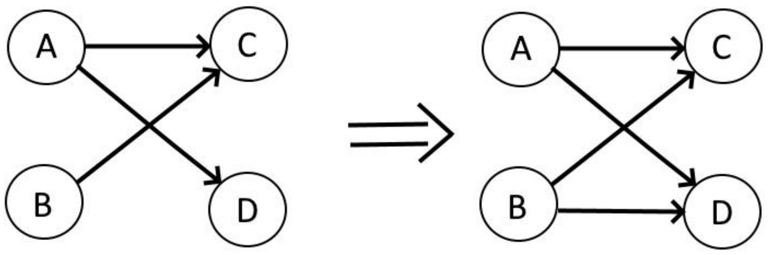
Eksk

SEASON 2021/2022 – FORTH ROUND

Kyusho decided it's time take a break from all the stress of running the company and go on a family trip. After some intense thinking, he chose **N** destinations and **M** direct one-way roads to move along between them. To maximize the satisfaction of the tour, he wants to visit each of the destinations exactly once and after visiting the last one to be able to go directly back home. Thus, he made **T** plans for destinations and road segments between them, and since he is running out of time, he asks you to determine for which of them there is a route of the described type. One last detail Kyusho thinks might be helpful is the strange property he discovered for each plan while reviewing them:

If for two destinations **A** and **B** there is a destination **C** such that there are direct roads from **A** and **B** to it, then any other destination **D** to which there is a road segment from **A** or **B** is like **C** (there is a direct road from both **A** and from **B** to it).

Put differently, if we call destination **Y** a neighbour of **X** when there is a road segment from **X** to **Y**, then if two destinations have at least one neighbour in common, all of their neighbours are common.



Input

From the first line of the file `eksk.in` the number **T** is given. For each plan the first line contains **N** and **M** followed by **M** lines with two numbers **X** and **Y**, specifying that there is a one-way road from destination **X** to destination **Y** (**0** ≤ **X**, **Y** ≤ **N**; **0** indicates the modest mansion of Kyusho (the property described above applies also for it), and the numbers from **1** to **N** - the numbers of the destinations).

Output

For each of the **T** plans, print in the file `eksk.out` on a separate line **YES**, if there is a route visiting each destination once and **NO** otherwise. If the answer is **YES**, on the next line, print **N + 2** integers (the first and last of which are **0**), describing the destinations in the order of their visit. If there is more than one possible routes, print any of them.

Constraints

$1 \leq T \leq 10^2$

$1 \leq$ sum of all **N** $\leq 10^5$

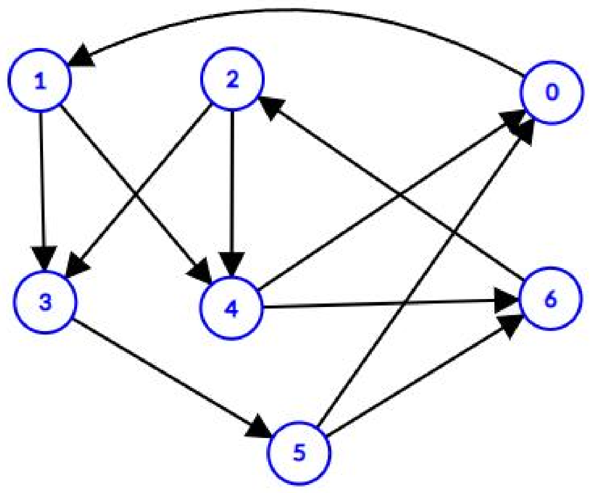
$1 \leq$ sum of all **M** $\leq 5 \times 10^6$

$0 \leq X, Y \leq N$ and $X \neq Y$ for each road segment

Time limit: 1.2 sec.
Memory limit: 256 MB.

Sample testcase

Input (eksk.in)	Output (eksk.out)	Explanation

1
6 11
1 3
2 3
1 4
2 4
3 5
5 6
4 6
4 0
5 0
0 1
6 2
	YES
	0 1 4 6 2 3 5 0
		

XorFun

SEASON 2021/2022 – FORTH ROUND

Sashka loves to regularly visit your class in informatics on Thursdays, where you always cover interesting topics. The last class was with a deputy teacher, who taught the whole class about bitwise operations. They emphasized on the *excluding or* (XOR) operation. Sashka, impressed by the capabilites of the operation, decided to open her older brother's notebook, full of interesting tasks, searching for XOR ones. Initially, she solved the easier problems, but now she stumbled upon the following exotic task:

You are gived a tree of N vertices, with weighted edges. You are allowed to make K changes of the values of the edges in order to minimize the count of *unpleasant* simple paths in the tree. A simple path is a path, which doesn't contain a vertex more than once. A path is *unpleasant*, when the *excluding or* of the values of the edges it contains, has an odd parity of set bits in its bitwise representation. Find the minimal possible count of *unpleasant* paths, after optimal use of operations.

Sashka used all of her knowedge in informatics, and was able to solve the task! At least that's what she thought, until next Thursday, when you said that she always uses $K + 1$ operations. Now she is very unhappy, which gives you the opportunity to cheer her up, by writing a program `xorfun.cpp`, which solves the mentioned task.

Input

The first line of `xorfun.in` constains two natural numbers N and $K.$ The next $N - 1$ lines contain 3 natural numbers – the two endpoints of an edge v_i и u_i, and its weight – $w_{i.}$

Output

On one line in `xorfun.out` you should print one number – the minimal count of *unpleasant* paths.

Constaints

$1 \leq N \leq 5\,000$

$1 \leq K \leq 500$

$1 \leq v_i, u_i \leq N$

$0 \leq w_i \leq 2^{31} - 1$

Time Limit: 3 sec.
Memory Limit: 256 MB.

Sample testcases

Input (xorfun.in)	Output (xorfun.out)

9 3
1 2 6
1 7 4
2 3 7
3 4 12
3 5 14
3 6 13
5 8 2
5 9 11
	8

10 3
5 2 938707311
6 5 312182765
7 5 952433887
1 5 1630822531
3 5 1850473008
9 1 510193547
8 6 1465047925
10 8 1237069467
4 9 1320045850
	9

15 4
1 3 1936267205
3 10 1050608599
15 1 1084713226
15 6 327822164
10 8 1085192654
1 9 479650095
12 9 16193632
12 11 1114880438
8 5 1011320449
8 13 2012210084
14 3 47220444
7 15 1406284003
4 1 16038788
13 2 733939625
	36