Codeforces Round #692 (Div. 1, based on Technocup 2021 Elimination Round 3)

A. Peaceful Rooks

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a $n \times n$ chessboard. Rows and columns of the board are numbered from 1 to n. Cell (x, y) lies on the intersection of column number x and row number y.

Rook is a chess piece, that can in one turn move any number of cells vertically or horizontally. There are m rooks ($m < n$) placed on the chessboard in such a way that no pair of rooks attack each other. I.e. there are no pair of rooks that share a row or a column.

In one turn you can move one of the rooks any number of cells vertically or horizontally. Additionally, it shouldn't be attacked by any other rook after movement. What is the minimum number of moves required to place all the rooks on the main diagonal?

The main diagonal of the chessboard is all the cells (i, i), where $1 \le i \le n$.

Input

The first line contains the number of test cases t ($1 \le t \le 10^3$). Description of the t test cases follows.

The first line of each test case contains two integers n and m — size of the chessboard and the number of rooks ($2 \le n \le 10^5$, $1 \le m < n$). Each of the next m lines contains two integers x_i and y_i — positions of rooks, i-th rook is placed in the cell (x_i, y_i) ($1 \le x_i, y_i \le n$). It's guaranteed that no two rooks attack each other in the initial placement.

The sum of n over all test cases does not exceed 10^5.

Output

For each of t test cases print a single integer — the minimum number of moves required to place all the rooks on the main diagonal.

It can be proved that this is always possible.

Example

input
4 3 1 2 3 3 2 2 1 1 2 5 3 2 3 3 1 1 2 5 4 4 5 5 1 2 2 3 3
output
1 3 4 2

Note

Possible moves for the first three test cases:

1. $(2, 3) \to (2, 2)$
2. $(2, 1) \to (2, 3)$, $(1, 2) \to (1, 1)$, $(2, 3) \to (2, 2)$
3. $(2, 3) \to (2, 4)$, $(2, 4) \to (4, 4)$, $(3, 1) \to (3, 3)$, $(1, 2) \to (1, 1)$

B. Grime Zoo

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Currently, XXOC's rap is a string consisting of zeroes, ones, and question marks. Unfortunately, haters gonna hate. They will write x

angry comments for every occurrence of **subsequence** `01` and y angry comments for every occurrence of **subsequence** `10`. You should replace all the question marks with `0` or `1` in such a way that the number of angry comments would be as small as possible.

String b is a subsequence of string a, if it can be obtained by removing some characters from a. Two occurrences of a subsequence are considered distinct if sets of positions of remaining characters are distinct.

Input
The first line contains string s — XXOC's rap ($1 \le |s| \le 10^5$). The second line contains two integers x and y — the number of angry comments XXOC will recieve for every occurrence of `01` and `10` accordingly ($0 \le x, y \le 10^6$).

Output
Output a single integer — the minimum number of angry comments.

Examples

input
0?1 2 3
output
4

input
????? 13 37
output
0

input
?10? 239 7
output
28

input
01101001 5 7
output
96

Note
In the first example one of the optimum ways to replace is `001`. Then there will be 2 subsequences `01` and 0 subsequences `10`. Total number of angry comments will be equal to $2 \cdot 2 + 0 \cdot 3 = 4$.

In the second example one of the optimum ways to replace is `11111`. Then there will be 0 subsequences `01` and 0 subsequences `10`. Total number of angry comments will be equal to $0 \cdot 13 + 0 \cdot 37 = 0$.

In the third example one of the optimum ways to replace is `1100`. Then there will be 0 subsequences `01` and 4 subsequences `10`. Total number of angry comments will be equal to $0 \cdot 239 + 4 \cdot 7 = 28$.

In the fourth example one of the optimum ways to replace is `01101001`. Then there will be 8 subsequences `01` and 8 subsequences `10`. Total number of angry comments will be equal to $8 \cdot 5 + 8 \cdot 7 = 96$.

C. Poman Numbers

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You've got a string S consisting of n lowercase English letters from your friend. It turned out that this is a number written in *poman* numerals. The poman numeral system is long forgotten. All that's left is the algorithm to transform number from poman numerals to the numeral system familiar to us. Characters of S are numbered from 1 to n from left to right. Let's denote the value of S as $f(S)$, it is defined as follows:

- If $|S| > 1$, an arbitrary integer m ($1 \le m < |S|$) is chosen, and it is defined that $f(S) = -f(S[1, m]) + f(S[m + 1, |S|])$, where $S[l, r]$ denotes the substring of S from the l-th to the r-th position, inclusively.
- Otherwise $S = c$, where c is some English letter. Then $f(S) = 2^{pos(c)}$, where $pos(c)$ is the position of letter c in the alphabet ($pos(\text{a}) = 0$, $pos(\text{z}) = 25$).

Note that m is chosen independently on each step.

Your friend thinks it is possible to get $f(S) = T$ by choosing the right m on every step. Is he right?

Input
The first line contains two integers n and T ($2 \le n \le 10^5$, $-10^{15} \le T \le 10^{15}$).

The second line contains a string S consisting of n lowercase English letters.

Output
Print "Yes" if it is possible to get the desired value. Otherwise, print "No".

You can print each letter in any case (upper or lower).

Examples

input
2 -1 ba
output
Yes

input
3 -7 abc
output
No

input
7 -475391 qohshra
output
Yes

Note
In the second example, you cannot get -7. But you can get 1, for example, as follows:

1. First choose $m = 1$, then $f(\text{abc}) = -f(\text{a}) + f(\text{bc})$
2. $f(\text{a}) = 2^0 = 1$
3. $f(\text{bc}) = -f(\text{b}) + f(\text{c}) = -2^1 + 2^2 = 2$
4. In the end $f(\text{abc}) = -1 + 2 = 1$

D. The Thorny Path

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

According to a legend the Hanoi Temple holds a permutation of integers from 1 to n. There are n stones of distinct colors lying in one line in front of the temple. Monks can perform the following operation on stones: choose a position i ($1 \le i \le n$) and cyclically shift stones at positions i, $p[i]$, $p[p[i]]$, ... That is, a stone from position i will move to position $p[i]$, a stone from position $p[i]$ will move to position $p[p[i]]$, and so on, a stone from position j, such that $p[j] = i$, will move to position i.

Each day the monks must obtain a new arrangement of stones using an arbitrary number of these operations. When all possible arrangements will have been obtained, the world will end. You are wondering, what if some elements of the permutation could be swapped just before the beginning? How many days would the world last?

You want to get a permutation that will allow the world to last as long as possible, using the minimum number of exchanges of two elements of the permutation.

Two arrangements of stones are considered different if there exists a position i such that the colors of the stones on that position are different in these arrangements.

Input
Each test consists of multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). Description of the test cases follows.

The first line of each test case contains n ($3 \le n \le 10^6$). The next line contains n integers $p_1, \ldots, p_n$ ($1 \le p_i \le n$). It is guaranteed that p is a permutation.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output
For each of the t test cases, print two integers on a new line: the largest possible number of days the world can last, modulo $10^9 + 7$, and the minimum number of exchanges required for that.

Examples

input
3 3 2 3 1 3 2 1 3 3 1 2 3
output
3 0 3 1 3 2

input
5 4 2 3 4 1 4 2 3 1 4 4 2 1 4 3 4 2 1 3 4 4 1 2 3 4
output
4 0 4 1 4 0 4 1 4 2

Note

Let's label the colors of the stones with letters. Explanations for the first two test cases of the first example:

1. Using the permutation $[2, 3, 1]$, we can additionally obtain the arrangements CAB and BCA from ABC. This is already the maximum possible result.
2. Using the permutation $[2, 1, 3]$, the only BAC can be obtained from ABC. As we saw in the previous case, two arrangements are not the maximum possible number of distinct arrangements for $n = 3$. To get an optimal permutation, for example, we can swap 1 and 3, so we will get the permutation $[2, 3, 1]$.

E. No Game No Life

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's consider the following game of Alice and Bob on a directed acyclic graph. Each vertex may contain an arbitrary number of chips. Alice and Bob make turns alternating. Alice goes first. In one turn player can move exactly one chip along any edge outgoing from the vertex that contains this chip to the end of this edge. The one who cannot make a turn loses. Both players play optimally.

Consider the following process that takes place every second on a given graph with n vertices:

1. An integer v is chosen equiprobably from $[1, n + 1]$.
2. If $v \leq n$, we add a chip to the v-th vertex and go back to step 1.
3. If $v = n + 1$, Alice and Bob play the game with the current arrangement of chips and the winner is determined. After that, the process is terminated.

Find the probability that Alice will win the game. It can be shown that the answer can be represented as $\frac{P}{Q}$, where P and Q are coprime integers and $Q \not\equiv 0 \pmod{998\,244\,353}$. Print the value of $P \cdot Q^{-1} \bmod 998\,244\,353$.

Input

The first line contains two integers n and m — the number of vertices and edges of the graph ($1 \leq n \leq 10^5$, $0 \leq m \leq 10^5$).

The following m lines contain edges description. The i-th of them contains two integers u_i and v_i — the beginning and the end vertices of the i-th edge ($1 \leq u_i, v_i \leq n$). It's guaranteed that the graph is acyclic.

Output

Output a single integer — the probability of Alice victory modulo $998\,244\,353$.

Examples

input
1 0

output
0

input
2 1 1 2
output
332748118

input
5 5 1 4 5 2 4 3 1 5 5 4
output
931694730

F. My Beautiful Madness

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a tree. We will consider simple paths on it. Let's denote path between vertices a and b as (a, b). Let d-neighborhood of a path be a set of vertices of the tree located at a distance $\leq d$ from at least one vertex of the path (for example, 0-neighborhood of a path is a path itself). Let P be a multiset of the tree paths. Initially, it is empty. You are asked to maintain the following queries:

- $1\ u\ v$ — add path (u, v) into P ($1 \leq u, v \leq n$).
- $2\ u\ v$ — delete path (u, v) from P ($1 \leq u, v \leq n$). Notice that (u, v) equals to (v, u). For example, if $P = \{(1, 2), (1, 2)\}$, than after query $2\ 2\ 1$, $P = \{(1, 2)\}$.
- $3\ d$ — if intersection of all d-neighborhoods of paths from P is not empty output "Yes", otherwise output "No" ($0 \leq d \leq n - 1$).

Input

The first line contains two integers n and q — the number of vertices in the tree and the number of queries, accordingly ($1 \leq n \leq 2 \cdot 10^5$, $2 \leq q \leq 2 \cdot 10^5$).

Each of the following $n - 1$ lines contains two integers x_i and y_i — indices of vertices connected by i-th edge ($1 \leq x_i, y_i \leq n$).

The following q lines contain queries in the format described in the statement.

It's guaranteed that:

- for a query $2\ u\ v$, path (u, v) (or (v, u)) is present in P,
- for a query $3\ d$, $P \neq \varnothing$,
- there is at least one query of the third type.

Output

For each query of the third type output answer on a new line.

Examples

input
1 4 1 1 1 1 1 1 2 1 1 3 0
output
Yes

input
5 3 1 2 1 3 3 4 4 5 1 1 2 1 5 5 3 1
output

No

input
10 6 1 2 2 3 3 4 4 7 7 10 2 5 5 6 6 8 8 9 1 9 9 1 9 8 1 8 5 3 0 3 1 3 2
output
No Yes Yes