

Codeforces Round #556 (Div. 1)

A. Prefix Sum Primes

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

We're giving away nice huge bags containing number tiles! A bag we want to present to you contains n tiles. Each of them has a single number written on it — either 1 or 2.

However, there is one condition you must fulfill in order to receive the prize. You will need to put all the tiles from the bag in a sequence, in any order you wish. We will then compute the sums of all prefixes in the sequence, and then count how many of these sums are prime numbers. If you want to keep the prize, you will need to maximize the number of primes you get.

Can you win the prize? Hurry up, the bags are waiting!

Input

The first line of the input contains a single integer n ($1 \leq n \leq 200\,000$) — the number of number tiles in the bag. The following line contains n space-separated integers a_1, a_2, \dots, a_n ($a_i \in \{1, 2\}$) — the values written on the tiles.

Output

Output a permutation b_1, b_2, \dots, b_n of the input sequence (a_1, a_2, \dots, a_n) maximizing the number of the prefix sums being prime numbers. If there are multiple optimal permutations, output any.

Examples

input
5 1 2 1 2 1
output
1 1 1 2 2

input
9 1 1 2 1 1 1 2 1 1
output
1 1 1 2 1 1 1 2 1

Note

The first solution produces the prefix sums 1, **2, 3, 5, 7** (four primes constructed), while the prefix sums in the second solution are 1, **2, 3, 5, 6, 7, 8, 10, 11** (five primes). Primes are marked bold and blue. In each of these cases, the number of produced primes is maximum possible.

B. Three Religions

time limit per test: 3 seconds
 memory limit per test: 256 megabytes
 input: standard input
 output: standard output

During the archaeological research in the Middle East you found the traces of three ancient religions: First religion, Second religion and Third religion. You compiled the information on the evolution of each of these beliefs, and you now wonder if the followers of each religion could coexist in peace.

The Word of Universe is a long word containing the lowercase English characters only. At each moment of time, each of the religion beliefs could be described by a word consisting of lowercase English characters.

The three religions can coexist in peace if their descriptions form disjoint subsequences of the Word of Universe. More formally, one can paint some of the characters of the Word of Universe in three colors: 1, 2, 3, so that each character is painted in at most one color, and the description of the i -th religion can be constructed from the Word of Universe by removing all characters that aren't painted in color i .

The religions however evolve. In the beginning, each religion description is empty. Every once in a while, either a character is appended to the end of the description of a single religion, or the last character is dropped from the description. After each change, determine if the religions could coexist in peace.

Input

The first line of the input contains two integers n, q ($1 \leq n \leq 100\,000, 1 \leq q \leq 1000$) — the length of the Word of Universe and the number of religion evolutions, respectively. The following line contains the Word of Universe — a string of length n consisting of lowercase English characters.

Each of the following line describes a single evolution and is in one of the following formats:

- $+ i\ c$ ($i \in \{1, 2, 3\}, c \in \{a, b, \dots, z\}$): append the character c to the end of i -th religion description.
- $- i$ ($i \in \{1, 2, 3\}$) - remove the last character from the i -th religion description. You can assume that the pattern is non-empty.

You can assume that no religion will have description longer than 250 characters.

Output

Write q lines. The i -th of them should be YES if the religions could coexist in peace after the i -th evolution, or NO otherwise.

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

Examples

input
6 8 abdabc + 1 a + 1 d + 2 b + 2 c + 3 a + 3 b + 1 c - 2
output
YES YES YES YES YES YES NO YES

input
6 8 abbaab + 1 a + 2 a + 3 a + 1 b + 2 b + 3 b - 1 + 2 z
output
YES YES YES YES YES NO YES NO

Note

In the first example, after the 6th evolution the religion descriptions are: ad, bc, and ab. The following figure shows how these descriptions form three disjoint subsequences of the Word of Universe:

Word	a	b	d	a	b	c
ad	a		d			
bc		b				c
ab				a	b	

C. Tree Generator™

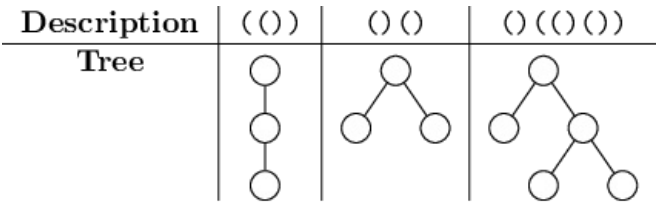
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Owl Pacino has always been into trees — unweighted rooted trees in particular. He loves determining the *diameter* of every tree he

sees — that is, the maximum length of any simple path in the tree.

Owl Pacino's owl friends decided to present him the Tree Generator™ — a powerful machine creating rooted trees from their *descriptions*. An n -vertex rooted tree can be *described* by a bracket sequence of length $2(n - 1)$ in the following way: find any walk starting and finishing in the root that traverses each edge exactly twice — once down the tree, and later up the tree. Then follow the path and write down "(" (an opening parenthesis) if an edge is followed down the tree, and ")" (a closing parenthesis) otherwise.

The following figure shows sample rooted trees and their descriptions:



Owl wrote down the description of an n -vertex rooted tree. Then, he rewrote the description q times. However, each time he wrote a new description, he picked two different characters in the description he wrote the last time, swapped them and wrote down the resulting string. He always made sure that each written string was the description of a rooted tree.

Pacino then used Tree Generator™ for each description he wrote down. What is the diameter of each constructed tree?

Input

The first line of the input contains two integers n, q ($3 \leq n \leq 100\,000, 1 \leq q \leq 100\,000$) — the number of vertices in the tree and the number of changes to the tree description. The following line contains a description of the initial tree — a string of length $2(n - 1)$ consisting of opening and closing parentheses.

Each of the following q lines describes a single change to the description and contains two space-separated integers a_i, b_i ($2 \leq a_i, b_i \leq 2n - 3$) which identify the indices of two brackets to be swapped. You can assume that the description will change after each query, and that after each change a tree can be constructed from the description.

Output

Output $q + 1$ integers — the diameter of each constructed tree, in the order their descriptions have been written down.


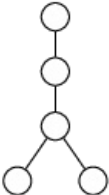
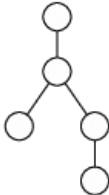
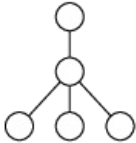

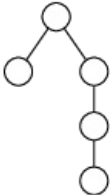
Examples

input
5 5 (((())) 4 5 3 4 5 6 3 6 2 5
output
4 3 3 2 4 4

input
6 4 ((())) 6 7 5 4 6 4 7 4
output
4 4 4 5 3

Note

The following figure shows each constructed tree and its description in the first example test:

Query		4 5	3 4	5 6	3 6	2 5
Brackets	((((()))	(((()))	(((()))	(((()))	(((()))	(((()))
Tree						

D. Abandoning Roads

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Codefortia is a small island country located somewhere in the West Pacific. It consists of n settlements connected by m bidirectional gravel roads. Curiously enough, the beliefs of the inhabitants require the time needed to pass each road to be equal either to a or b seconds. It's guaranteed that one can go between any pair of settlements by following a sequence of roads.

Codefortia was recently struck by the financial crisis. Therefore, the king decided to abandon some of the roads so that:

- it will be possible to travel between each pair of cities using the remaining roads only,
- the sum of times required to pass each remaining road will be minimum possible (in other words, remaining roads must form minimum spanning tree, using the time to pass the road as its weight),
- among all the plans minimizing the sum of times above, the time required to travel between the king's residence (in settlement 1) and the parliament house (in settlement p) using the remaining roads only will be minimum possible.

The king, however, forgot where the parliament house was. For each settlement $p = 1, 2, \dots, n$, can you tell what is the minimum time required to travel between the king's residence and the parliament house (located in settlement p) after some roads are abandoned?

Input

The first line of the input contains four integers n, m, a and b ($2 \leq n \leq 70, n - 1 \leq m \leq 200, 1 \leq a < b \leq 10^7$) — the number of settlements and gravel roads in Codefortia, and two possible travel times. Each of the following lines contains three integers u, v, c ($1 \leq u, v \leq n, u \neq v, c \in \{a, b\}$) denoting a single gravel road between the settlements u and v , which requires c minutes to travel.

You can assume that the road network is connected and has no loops or multiedges.

Output

Output a single line containing n integers. The p -th of them should denote the minimum possible time required to travel from 1 to p after the selected roads are abandoned. Note that for each p you can abandon a different set of roads.

Examples

input
5 5 20 25 1 2 25 2 3 25 3 4 20 4 5 20 5 1 20
output
0 25 60 40 20

input
6 7 13 22 1 2 13 2 3 13 1 4 22 3 4 13 4 5 13 5 6 13 6 1 13
output
0 13 26 39 26 13

Note

The minimum possible sum of times required to pass each road in the first example is 85 — exactly one of the roads with passing time 25 must be abandoned. Note that after one of these roads is abandoned, it's now impossible to travel between settlements 1 and 3 in time 50.

E. Election Promises

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

In Byteland, there are two political parties fighting for seats in the Parliament in the upcoming elections: *Wrong Answer Party* and *Time Limit Exceeded Party*. As they want to convince as many citizens as possible to cast their votes on them, they keep promising lower and lower taxes.

There are n cities in Byteland, connected by m one-way roads. Interestingly enough, the road network has no cycles — it's impossible to start in any city, follow a number of roads, and return to that city. Last year, citizens of the i -th city had to pay h_i bourles of tax.

Parties will now alternately hold the election conventions in various cities. If a party holds a convention in city v , the party needs to *decrease* the taxes in this city to a non-negative integer amount of bourles. However, at the same time they can arbitrarily modify the taxes in each of the cities that can be reached from v using a single road. The only condition that must be fulfilled that the tax in each city has to remain a non-negative integer amount of bourles.

The first party to hold the convention is *Wrong Answer Party*. It's predicted that the party to hold the last convention will win the election. Can *Wrong Answer Party* win regardless of *Time Limit Exceeded Party*'s moves?

Input

The first line of the input contains two integers n, m ($1 \leq n \leq 200\,000$, $0 \leq m \leq 200\,000$) — the number of cities and roads in Byteland.

The next line contains n space-separated integers h_1, h_2, \dots, h_n ($0 \leq h_i \leq 10^9$); h_i denotes the amount of taxes paid in the i -th city.

Each of the following m lines contains two integers ($1 \leq u, v \leq n$, $u \neq v$), and describes a one-way road leading from the city u to the city v . There will be no cycles in the road network. No two roads will connect the same pair of cities.

We can show that the conventions cannot be held indefinitely for any correct test case.

Output

If *Wrong Answer Party* can win the election, output WIN in the first line of your output. In this case, you're additionally asked to produce any convention allowing the party to win regardless of the opponent's actions. The second line should contain n non-negative integers h'_1, h'_2, \dots, h'_n ($0 \leq h'_i \leq 2 \cdot 10^{18}$) describing the amount of taxes paid in consecutive cities after the convention. If there are multiple answers, output any. We guarantee that if the party has any winning move, there exists a move after which no city has to pay more than $2 \cdot 10^{18}$ bourles.

If the party cannot assure their victory, output LOSE in the first and only line of the output.

Examples

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

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

input
3 3 314 159 265 1 2 1 3 3 2
output
WIN 0 0 0

input
6 4 2 2 5 5 6 6

1 3 2 4 3 5 4 6
output
LOSE

Note

In the first example, *Wrong Answer Party* should hold the convention in the city 1. The party will decrease the taxes in this city to 1 bourle. As the city 2 is directly reachable from 1, we can arbitrarily modify the taxes in this city. The party should change the tax to 5 bourles. It can be easily proved that *Time Limit Exceeded* cannot win the election after this move if *Wrong Answer Party* plays optimally.

The second example test presents the situation we created after a single move in the previous test; as it's *Wrong Answer Party's* move now, the party cannot win.

In the third test, we should hold the convention in the first city. This allows us to change the taxes in any city to any desired value; we can for instance decide to set all the taxes to zero, which allows the *Wrong Answer Party* to win the election immediately.