Problem A. Palindrome Dance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

A group of n dancers rehearses a performance for the closing ceremony. The dancers are arranged in a row, they've studied their dancing moves and can't change positions. For some of them, a white dancing suit is already bought, for some of them — a black one, and for the rest the suit will be bought in the future.

On the day when the suits were to be bought, the director was told that the participants of the olympiad will be happy if the colors of the suits on the scene will form a palindrome. A palindrome is a sequence that is the same when read from left to right and when read from right to left. The director liked the idea, and she wants to buy suits so that the color of the leftmost dancer's suit is the same as the color of the rightmost dancer's suit, the 2nd left is the same as 2nd right, and so on.

The director knows how many burls it costs to buy a white suit, and how many burls to buy a black suit. You need to find out whether it is possible to buy suits to form a palindrome, and if it's possible, what's the minimal cost of doing so. Remember that dancers can not change positions, and due to bureaucratic reasons it is not allowed to buy new suits for the dancers who already have suits, even if it reduces the overall spending.

Input

The first line contains three integers n, a, and b ($1 \le n \le 20$, $1 \le a, b \le 100$) — the number of dancers, the cost of a white suit, and the cost of a black suit.

The next line contains n numbers c_i, i-th of which denotes the color of the suit of the i-th dancer. Number 0 denotes the white color, 1 — the black color, and 2 denotes that a suit for this dancer is still to be bought.

Output

If it is not possible to form a palindrome without swapping dancers and buying new suits for those who have one, then output `-1`. Otherwise, output the minimal price to get the desired visual effect.

Examples

standard input	standard output

`5 100 1`
`0 1 2 1 2`
	`101`

`3 10 12`
`1 2 0`
	`-1`

`3 12 1`
`0 1 0`
	`0`

Note

In the first sample, the cheapest way to obtain palindromic colors is to buy a black suit for the third from left dancer and a white suit for the rightmost dancer.

In the second sample, the leftmost dancer's suit already differs from the rightmost dancer's suit so there is no way to obtain the desired coloring.

In the third sample, all suits are already bought and their colors form a palindrome.

Scoring

There are 103 tests in this problem. Samples are not scored and all other tests cost 1 point each.

It is guaranteed that solutions passing all tests where $n \leq 2$ will get at least 40 points.

Problem B. Subway Pursuit

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

This is an interactive problem. It means that while your submissions will be running there will be a jury program running simultaneously and you should communicate via standard input and standard output. Detailed protocol description is provided below. Also, at the end of this problem statement you will find correct communication examples for different programming languages.

In the Wonderful Metropolis of the Future, there is no need in subway train drivers. Due to the technological progress, they were replaced by the Artificial Intelligence (AI). Unfortunately, one day the predictions of sci-fi writers came true: the AI rebelled and now there is an uncontrollable train in the subway. It can be dangerous! Your task is to find the train and stop the AI.

All other trains were moved to depots, and all the subway lines except for the one with the uncontrollable train were closed, so now the subway of the Metropolis is one line (regular straight line with no self-intersections) with n stations, indexed consecutively from 1 to n, one of which contains the train. You need to determine the index of this station, so that the rails around that station would be barraged and the train would be secured.

To find the station, dispatcher Sarah gave you a gadget that allows you to select arbitrary numbers l and r ($l \leq r$), and then check, whether the train is located on a station with index between l and r. Unfortunalely, recharging of the gadget takes k minutes (and every time you use it as soon as possible), so between two applications of the gadget the train can move to any station that is at most k stations away. Formally, if the train was at the station x when the gadget was applied, then at the next application of the gadget the train can appear at any station y such that $\max(1, x-k) \leq y \leq \min(n, x+k)$. Note that AI is not aware that you are trying to catch the train, so it makes all moves according to its predefined plan.

After an examination of the gadget you found that it is very old and can hold no more than q applications, after which it will break and your mission will be considered a failure.

Can you find the station with the train using no more than q applications of the gadgets?

Interaction Protocol

In the beginning your program receives three integers n ($1 \leq n \leq 10^{18}$), k ($0 \leq k \leq 10$), and q ($q \leq 15\,000$).

In order to apply the gadget you need to enter two space-separated integers l and r ($1 \leq l \leq r \leq n$). You will then recieve either string "`Yes`", if the train is between stations l and r, or string "`No`" otherwise. If $l = r$ and you received "`Yes`", then you found the train successfully, and your program must halt immediately.

Please, note that if your program has already made all q queries it should halt immediately whether it found the answer or not, otherwise the judging system verdict can be arbitrary.

After each query you have to end the line and flush the output buffer — to do that use `flush(output)` in Pascal and Delphi, `fflush(stdout)` or `cout.flush()` in C/C++, `sys.stdout.flush()` in Python, and `System.out.flush()` in Java.

Precisely follow this format of interaction.

Example

standard input	standard output
`10 2 15000`	
	`3 4`
`Yes`	
	`3 3`
`No`	
	`2 2`
`Yes`	

Scoring

Tests for this problem are divided into four groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in all of the **previous** groups.

Group	Points	Extra limitations			Comment	Required groups
		n	k	q		
0	0	$n = 10$	$k = 2$	$q = 15\,000$	Sample test.	—
1	20	$n \le 10^{18}$	$k = 0$	$q = 15\,000$		—
2	20	$n \le 10^3$	$k \le 10$	$q = 15\,000$		0
3	30	$n \le 10^6$	$k \le 10$	$q = 15\,000$		0, 2
4	30	$n \le 10^{18}$	$k \le 10$	$q = 4500$		0, 1, 2, 3

Note

In the first sample, the train was initially at the station 3, after the first application of the gadget it moved to the station 2, and after the second application it did not move.

Below you will find example programs in **C++**, **Java**, **Python 2** and **Python 3**. These programs get 0 points but illustrate the correct interaction with jury program and output buffer flushing.

C++

```
#include <iostream>

using namespace std;

long long n, k, q;

bool ask(long long l, int r) {
    // using endl forces output buffer flush
    cout << l << ' ' << r << endl;
    string ans;
    cin >> ans;
    return ans == "Yes";
}

main() {
    cin >> n >> k >> q;
    while (q--) {
        bool res = ask(1, 1);
        if (res) {
            return 0;
        }
    }
}
```

Java

```
import java.util.Scanner;

public class Main {

    static Scanner in;

    static boolean ask(long l, long r) {
        System.out.print(l);
        System.out.print(" ");
        System.out.print(r);
        System.out.println();
        System.out.flush();
        String ans = in.next();
        return ans.equals("Yes");
    }

    public static void main(String[] args) {
        in = new Scanner(System.in);
        long n = in.nextLong();
        long k = in.nextLong();
        long q = in.nextLong();
        while (q > 0) {
            q--;
            boolean res = ask(1, 1);
            if (res) {
                return;
            }
        }
    }
}
```

Python 2

```
import sys

def ask(l, r):
    print str(l) + ' ' + str(r)
    sys.stdout.flush()
    ans = raw_input()
    return ans == "Yes"

n, k, q = map(int, raw_input().split())
while q > 0:
    q -= 1
    res = ask(1, 1)
    if res:
        exit(0)
```

Python 3

```
import sys

def ask(l, r):
    print(l, r)
    sys.stdout.flush()
    ans = input()
    return ans == "Yes"

n, k, q = map(int, input().split())
while q > 0:
    q -= 1
    res = ask(1, 1)
    if res:
        exit()
```

Problem C. Network Safety

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

The Metropolis computer network consists of n servers, each has an encryption key in the range from 0 to 2^k-1 assigned to it. Let c_i be the encryption key assigned to the i-th server. Additionally, m pairs of servers are directly connected via a data communication channel. Because of the encryption algorithms specifics, a data communication channel can only be considered safe if the two servers it connects have **distinct** encryption keys. The initial assignment of encryption keys is guaranteed to keep all data communication channels safe.

You have been informed that a new virus is actively spreading across the internet, and it is capable to change the encryption key of any server it infects. More specifically, the virus body contains some unknown number x in the same aforementioned range, and when server i is infected, its encryption key changes from c_i to $c_i \oplus x$, where $\oplus$ denotes the bitwise exclusive OR operation (also called XOR).

Sadly, you know neither the number x nor which servers of Metropolis are going to be infected by the dangerous virus, so you have decided to count the number of such situations in which all data communication channels remain safe. Formally speaking, you need to find the number of pairs (A, x), where A is some (possibly empty) subset of the set of servers and x is some number in the range from 0 to $2^k - 1$, such that when all servers from the chosen subset A and none of the others are infected by a virus containing the number x, all data communication channels remain safe. Since this number can be quite big, you are asked to find its remainder modulo $10^9 + 7$.

Input

The first line of input contains three integers n, m and k ($1 \le n \le 500\,000$, $0 \le m \le \min(\frac{n(n-1)}{2}, 500\,000)$, $0 \le k \le 60$) — the number of servers, the number of pairs of servers directly connected by a data communication channel, and the parameter k, which defines the range of possible values for encryption keys.

The next line contains n integers c_i ($0 \le c_i \le 2^k - 1$), the i-th of which is the encryption key used by the i-th server.

The next m lines contain two integers u_i and v_i each ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) denoting that those servers are connected by a data communication channel. It is guaranteed that each pair of servers appears in this list at most once.

Output

The only output line should contain a single integer — the number of safe infections of some subset of servers by a virus with some parameter, modulo $10^9 + 7$.

Examples

standard input	standard output

4 4 2
0 1 0 1
1 2
2 3
3 4
4 1
	50

4 5 3
7 1 7 2
1 2
2 3
3 4
4 1
2 4
	96

Note

Let's define the bitwise exclusive OR (XOR) operation. Given two integers x and y, consider their binary representations (possibly with leading zeroes): $x_k \dots x_2 x_1 x_0$ and $y_k \dots y_2 y_1 y_0$. Here, x_i is the i-th bit of the number x and y_i is the i-th bit of the number y. Let $r = x \oplus y$ be the result of the XOR operation of x and y. Then r is defined as $r_k \dots r_2 r_1 r_0$ where:

$$r_i = \begin{cases} 1, \text{ if } x_i \neq y_i \\ 0, \text{ if } x_i = y_i \end{cases}$$

Scoring

The tests to this problem are divided into four groups. Points for passing a group of tests are awarded only when all tests of that and all the **previous** groups have been passed.

Group	Points	Additional restrictions			Commentary
		n	m	k	
0	0	–	–	–	Samples from statement.
1	10	$n \le 10$	$m \le 45$	$k \le 10$	
2	20	$n \le 5000$	$m \le 5000$	$k \le 14$	
3	30	$n \le 5000$	$m \le 5000$	–	
4	40	–	–	–	

Problem D. You Are Given a Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

A tree is an undirected graph with exactly one simple path between each pair of vertices. We call a set of simple paths k-valid if each vertex of the tree belongs to no more than one of these paths (including endpoints) and each path consists of exactly k vertices.

You are given a tree with n vertices. For each k from 1 to n inclusive find what is the maximum possible size of a k-valid set of simple paths.

Input

The first line of the input contains a single integer n ($2 \le n \le 500\,000$) — the number of vertices in the tree.

Then following $n-1$ lines describe the tree, each of them contains two integers v, u ($1 \le v, u \le n$) — endpoints of the corresponding edge.

It is guaranteed, that the given graph is a tree.

Output

Output n numbers, the i-th of which is the maximum possible number of paths in an i-valid set of paths.

Examples

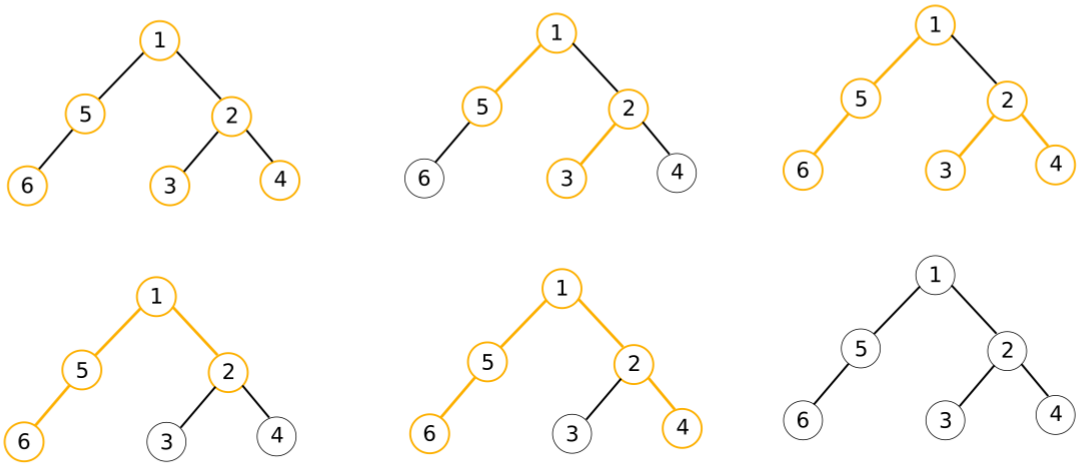
standard input	standard output

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

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

Note

One way to achieve the optimal number of paths for the second sample is illustrated in the following picture:



Scoring

Tests for this problem are divided into 14 groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in all of the **previous** groups.

Group	Points	Additional constraints	Comment
		n	
0	0	—	Sample tests
1	7	$n \le 50$	—
2	7	$n \le 500$	—
3	7	$n \le 3000$	—
4	7	$n \le 10\,000$	—
5	8	$n \le 25\,000$	—
6	6	$n \le 50\,000$	—
7	9	$n \le 75\,000$	—
8	6	$n \le 100\,000$	—
9	7	$n \le 150\,000$	—
10	7	$n \le 200\,000$	—
11	7	$n \le 250\,000$	—
12	8	$n \le 300\,000$	—
13	7	$n \le 400\,000$	—
14	7	$n \le 500\,000$	—