Codeforces Round #507 (Div. 1, based on Olympiad of Metropolises)

A. Timetable

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

There are two bus stops denoted A and B, and there n buses that go from A to B every day. The shortest path from A to B takes t units of time but some buses might take longer paths. Moreover, buses are allowed to overtake each other during the route.

At each station one can find a sorted list of moments of time when a bus is at this station. We denote this list as $a_1 < a_2 < ... < a_n$ for stop A and as $b_1 < b_2 < ... < b_n$ for stop B. The buses always depart from A and arrive to B according to the timetable, but the order in which the buses arrive may differ. Let's call an order of arrivals valid if each bus arrives at least t units of time later than departs.

It is known that for an order to be valid the latest possible arrival for the bus that departs at a_i is b_{x_i}, i.e. x_i-th in the timetable. In other words, for each i there exists such a valid order of arrivals that the bus departed i-th arrives x_i-th (and all other buses can arrive arbitrary), but there is no valid order of arrivals in which the i-th departed bus arrives $(x_i + 1)$-th.

Formally, let's call a permutation $p_1, p_2, ..., p_n$ valid, if $b_{p_i} \geq a_i + t$ for all i. Then x_i is the maximum value of p_i among all valid permutations.

You are given the sequences $a_1, a_2, ..., a_n$ and $x_1, x_2, ..., x_n$, but not the arrival timetable. Find out any suitable timetable for stop B $b_1, b_2, ..., b_n$ or determine that there is no such timetable.

Input

The first line of the input contains two integers n and t $(1 \leq n \leq 200\,000, 1 \leq t \leq 10^{18})$ — the number of buses in timetable for and the minimum possible travel time from stop A to stop B.

The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \leq a_1 < a_2 < ... < a_n \leq 10^{18})$, defining the moments of time when the buses leave stop A.

The third line contains n integers $x_1, x_2, ..., x_n$ $(1 \leq x_i \leq n)$, the i-th of them stands for the maximum possible timetable position, at which the i-th bus leaving stop A can arrive at stop B.

Output

If a solution exists, print "Yes" (without quotes) in the first line of the output.

In the second line print n integers $b_1, b_2, ..., b_n$ $(1 \leq b_1 < b_2 < ... < b_n \leq 3 \cdot 10^{18})$. We can show that if there exists any solution, there exists a solution that satisfies such constraints on b_i. If there are multiple valid answers you can print any of them.

If there is no valid timetable, print "No" (without quotes) in the only line of the output.

Examples

input
3 10 4 6 8 2 2 3
output
Yes 16 17 21

input
2 1 1 2 2 1
output
No

Note

Consider the first example and the timetable $b_1, b_2, ..., b_n$ from the output.

To get $x_1 = 2$ the buses can arrive in the order $(2, 1, 3)$. To get $x_2 = 2$ and $x_3 = 3$ the buses can arrive in the order $(1, 2, 3)$. x_1 is not 3, because the permutations $(3, 1, 2)$ and $(3, 2, 1)$ (all in which the 1-st bus arrives 3-rd) are not valid (sube buses arrive too early), x_2 is not 3 because of similar reasons.

B. Subway Pursuit

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

This is an interactive problem.

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.

The subway of the Metropolis is one line (regular straight line with no self-intersections) with n stations, indexed consecutively from 1 to n. At each moment the train is at some station. You need to determine the index of this station, so that the train would be secured.

To find the train, 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, inclusive. Unfortunately, recharging of the gadget takes some time (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 4500 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 4500 applications of the gadgets?

Input
The first line contains two integers n and k ($1 \le n \le 10^{18}$, $0 \le k \le 10$) — the number of stations and the maximum number of stations the train can move between two applications of the gadget.

Interaction
You can apply the gadget at most 4500 times. In order to apply the gadget you need to print two space-separated integers l and r ($1 \le l \le r \le n$). You will then receive either string "Yes", if the train is between stations l and r, inclusive, or string "No" otherwise. If $l = r$ and you received "Yes", then you found the train successfully, and your program must halt immediately.

Answer "Bad" instead of "Yes" or "No" means that you made an invalid query or made too many queries. Exit immediately after receiving "Bad" and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

After printing a query do not forget to output end of line and flush the output. Otherwise you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hacks

In order to hack, you should present a test in the following format.

The first line should contain three integers n, k and p ($1 \le n \le 10^{18}$, $0 \le k \le 10$, $1 \le p \le n$) — the number of stations, the maximum number of stations the train can move between two applications of the gadget and the initial position of the train, respectively.

Each of the next 4500 lines should contain a single integer x ($1 \le x \le n$) — the positions of the train after each query. Two consecutive positions (including the initial one) should not differ by more than k.

For example, the following lines are the first lines of the sample test.

```
10 2 5
5
3
5
7
7
...
```

Example

input
10 2 Yes No Yes Yes
output
3 5 3 3 3 4 5 5

Note
In the first sample, the train was initially at the station 5, after the first application of the gadget it did not move, after the second application it moved to the station 3, and after the third application moved again to the station 5.

C. Network Safety

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

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 XOR operation.

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

input
4 4 2 0 1 0 1 1 2 2 3 3 4 4 1
output
50

input
4 5 3 7 1 7 2 1 2 2 3 3 4 4 1 2 4
output
96

Note
Consider the first example.

Possible values for the number x contained by the virus are 0, 1, 2 and 3.

For values 0, 2 and 3 the virus can infect any subset of the set of servers, which gives us 16 pairs for each values. A virus containing the number 1 can infect either all of the servers, or none. This gives us $16 + 2 + 16 + 16 = 50$ pairs in total.

D. You Are Given a Tree

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

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 ($2 \le n \le 100\,000$) — 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.

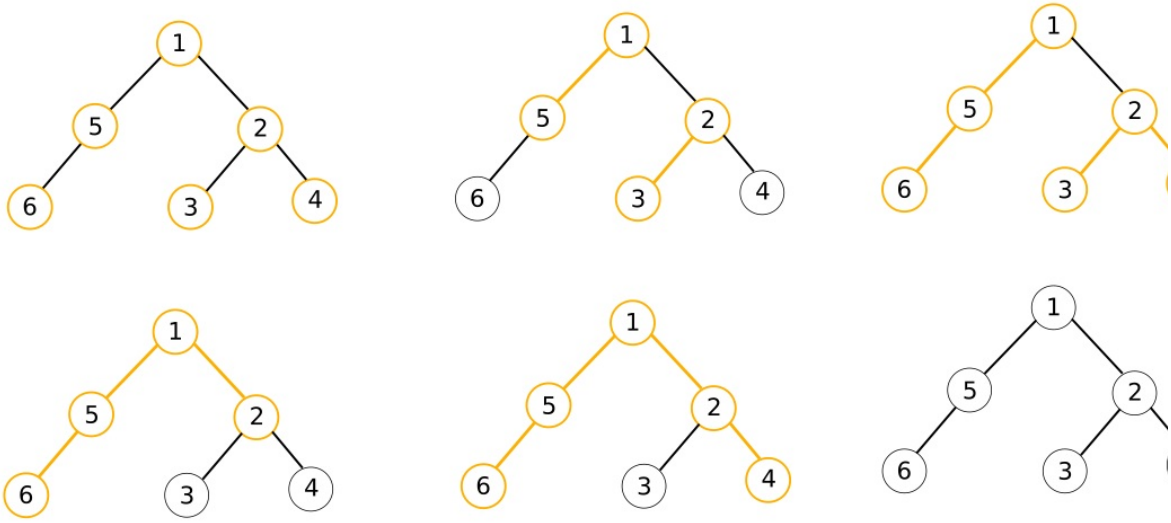
Output
Output n numbers, the i-th of which is the maximum possible number of paths in an i-valid set of paths.

Examples

input
7 1 2 2 3 3 4 4 5 5 6 6 7
output
7 3 2 1 1 1 1

input
6 1 2 2 3 2 4 1 5 5 6
output
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:



E. Summer Oenothera Exhibition

time limit per test: 7 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

While some people enjoy spending their time solving programming contests, Dina prefers taking beautiful pictures. As soon as Byteland Botanical Garden announced Summer Oenothera Exhibition she decided to test her new camera there.

The exhibition consists of $l = 10^{100}$ Oenothera species arranged in a row and consecutively numbered with integers from 0 to $l - 1$. Camera lens allows to take a photo of w species on it, i.e. Dina can take a photo containing flowers with indices from x to $x + w - 1$ for some integer x between 0 and $l - w$. We will denote such photo with $[x, x + w - 1]$.

She has taken n photos, the i-th of which (in chronological order) is $[x_i, x_i + w - 1]$ in our notation. She decided to build a time-lapse video from these photos once she discovered that Oenothera blossoms open in the evening.

Dina takes each photo and truncates it, leaving its segment containing exactly k flowers, then she composes a video of these photos keeping their original order and voilà, a beautiful artwork has been created!

A scene is a contiguous sequence of photos such that the set of flowers on them is the same. The change between two scenes is called a *cut*. For example, consider the first photo contains flowers $[1, 5]$, the second photo contains flowers $[3, 7]$ and the third photo contains flowers $[8, 12]$. If $k = 3$, then Dina can truncate the first and the second photo into $[3, 5]$, and the third photo into $[9, 11]$. First two photos form a scene, third photo also forms a scene and the transition between these two scenes which happens between the second and the third photos is a cut. If $k = 4$, then each of the transitions between photos has to be a cut.

Dina wants the number of cuts to be as small as possible. Please help her! Calculate the minimum possible number of cuts for different values of k.

Input

The first line contains three positive integer n, w, q ($1 \le n, q \le 100\,000$, $1 \le w \le 10^9$) — the number of taken photos, the number of flowers on a single photo and the number of queries.

Next line contains n non-negative integers x_i ($0 \le x_i \le 10^9$) — the indices of the leftmost flowers on each of the photos.

Next line contains q positive integers k_i ($1 \le k_i \le w$) — the values of k for which you have to solve the problem.

It's guaranteed that all k_i are distinct.

Output

Print q integers — for each width of the truncated photo k_i, the minimum number of cuts that is possible.

Examples

input
3 6 5 2 4 0 1 2 3 4 5
output
0 0 1 1 2

input
6 4 3 1 2 3 4 3 2 1 2 3
output
0 1 2