A. Mind the Gap

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

These days Arkady works as an air traffic controller at a large airport. He controls a runway which is usually used for landings only. Thus, he has a schedule of planes that are landing in the nearest future, each landing lasts 1 minute.

He was asked to insert one takeoff in the schedule. The takeoff takes 1 minute itself, but for safety reasons there should be a time space between the takeoff and any landing of at least s minutes from both sides.

Find the earliest time when Arkady can insert the takeoff.

Input

The first line of input contains two integers n and s ($1 \le n \le 100$, $1 \le s \le 60$) — the number of landings on the schedule and the minimum allowed time (in minutes) between a landing and a takeoff.

Each of next n lines contains two integers h and m ($0 \le h \le 23$, $0 \le m \le 59$) — the time, in hours and minutes, when a plane will land, starting from current moment (i. e. the current time is 0 0). These times are given in increasing order.

Output

Print two integers h and m — the hour and the minute from the current moment of the earliest time Arkady can insert the takeoff.

Examples

input

```
6 60
0 0
1 20
3 21
5 0
19 30
23 40
```

output

```
6 1
```

input

```
16 50
0 30
1 20
3 0
4 30
6 10
7 50
9 30
11 10
12 50
14 30
16 10
17 50
19 30
21 10
22 50
23 59
```

output

```
24 50
```

input

```
3 17
0 30
1 0
12 0
```

output

```
0 0
```

Note

In the first example note that there is not enough time between `1:20` and `3:21`, because each landing and the takeoff take one minute.

In the second example there is no gaps in the schedule, so Arkady can only add takeoff after all landings. Note that it is possible that one should wait more than 24 hours to insert the takeoff.

In the third example Arkady can insert the takeoff even between the first landing.

B. Watering System

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

Arkady wants to water his only flower. Unfortunately, he has a very poor watering system that was designed for n flowers and so it looks like a pipe with n holes. Arkady can only use the water that flows from the first hole.

Arkady can block some of the holes, and then pour A liters of water into the pipe. After that, the water will flow out from the non-blocked holes proportionally to their sizes $s_1, s_2, \ldots, s_n$. In other words, if the sum of sizes of non-blocked holes is S, and the i-th hole is not blocked, $\frac{s_i \cdot A}{S}$ liters of water will flow out of it.

What is the minimum number of holes Arkady should block to make at least B liters of water flow out of the first hole?

Input

The first line contains three integers n, A, B ($1 \le n \le 100\,000$, $1 \le B \le A \le 10^4$) — the number of holes, the volume of water Arkady will pour into the system, and the volume he wants to get out of the first hole.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 10^4$) — the sizes of the holes.

Output

Print a single integer — the number of holes Arkady should block.

Examples

input

```
4 10 3
2 2 2 2
```

output

```
1
```

input

```
4 80 20
3 2 1 4
```

output

```
0
```

input

```
5 10 10
1000 1 1 1 1
```

output

```
4
```

Note

In the first example Arkady should block at least one hole. After that, $\frac{10 \cdot 2}{6} \approx 3.333$ liters of water will flow out of the first hole, and that suits Arkady.

In the second example even without blocking any hole, $\frac{80 \cdot 3}{10} = 24$ liters will flow out of the first hole, that is not less than 20.

In the third example Arkady has to block all holes except the first to make all water flow out of the first hole.

C. Stairs and Elevators

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

In the year of $30XX$ participants of some world programming championship live in a single large hotel. The hotel has n floors. Each floor has m sections with a single corridor connecting all of them. The sections are enumerated from 1 to m along the corridor, and all sections with equal numbers on different floors are located exactly one above the other. Thus, the hotel can be represented as a rectangle of height n and width m. We can denote sections with pairs of integers (i, j), where i is the floor, and j is the section number on the floor.

The guests can walk along the corridor on each floor, use stairs and elevators. Each stairs or elevator occupies all sections $(1, x)$, $(2,

$x)$, $\ldots$, (n, x) for some x between 1 and m. All sections not occupied with stairs or elevators contain guest rooms. It takes one time unit to move between neighboring sections on the same floor or to move one floor up or down using stairs. It takes one time unit to move up to v floors in any direction using an elevator. You can assume you don't have to wait for an elevator, and the time needed to enter or exit an elevator is negligible.

You are to process q queries. Each query is a question "what is the minimum time needed to go from a room in section (x_1, y_1) to a room in section (x_2, y_2)?"

Input

The first line contains five integers n, m, c_l, c_e, v ($2 \leq n, m \leq 10^8$, $0 \leq c_l, c_e \leq 10^5$, $1 \leq c_l + c_e \leq m - 1$, $1 \leq v \leq n - 1$) — the number of floors and section on each floor, the number of stairs, the number of elevators and the maximum speed of an elevator, respectively.

The second line contains c_l integers $l_1, \ldots, l_{c_l}$ in increasing order ($1 \leq l_i \leq m$), denoting the positions of the stairs. If $c_l = 0$, the second line is empty.

The third line contains c_e integers $e_1, \ldots, e_{c_e}$ in increasing order, denoting the elevators positions in the same format. It is guaranteed that all integers l_i and e_i are distinct.

The fourth line contains a single integer q ($1 \leq q \leq 10^5$) — the number of queries.

The next q lines describe queries. Each of these lines contains four integers x_1, y_1, x_2, y_2 ($1 \leq x_1, x_2 \leq n$, $1 \leq y_1, y_2 \leq m$) — the coordinates of starting and finishing sections for the query. It is guaranteed that the starting and finishing sections are distinct. It is also guaranteed that these sections contain guest rooms, i. e. y_1 and y_2 are not among l_i and e_i.

Output

Print q integers, one per line — the answers for the queries.

Example

input

```
5 6 1 1 3
2
5
3
1 1 5 6
1 3 5 4
3 3 5 3
```

output

```
7
5
4
```

Note

In the first query the optimal way is to go to the elevator in the 5-th section in four time units, use it to go to the fifth floor in two time units and go to the destination in one more time unit.

In the second query it is still optimal to use the elevator, but in the third query it is better to use the stairs in the section 2.

D. Resource Distribution

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

One department of some software company has n servers of different specifications. Servers are indexed with consecutive integers from 1 to n. Suppose that the specifications of the j-th server may be expressed with a single integer number c_j of artificial resource units.

In order for production to work, it is needed to deploy two services S_1 and S_2 to process incoming requests using the servers of the department. Processing of incoming requests of service S_i takes x_i resource units.

The described situation happens in an advanced company, that is why each service may be deployed using not only one server, but several servers simultaneously. If service S_i is deployed using k_i servers, then the load is divided equally between these servers and each server requires only x_i / k_i (that may be a fractional number) resource units.

Each server may be left unused at all, or be used for deploying exactly one of the services (but not for two of them simultaneously). The service should not use more resources than the server provides.

Determine if it is possible to deploy both services using the given servers, and if yes, determine which servers should be used for deploying each of the services.

Input

The first line contains three integers n, x_1, x_2 ($2 \leq n \leq 300\,000$, $1 \leq x_1, x_2 \leq 10^9$) — the number of servers that the department may use, and resource units requirements for each of the services.

The second line contains n space-separated integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq 10^9$) — the number of resource units provided by each of the servers.

Output

If it is impossible to deploy both services using the given servers, print the only word "No" (without the quotes).

Otherwise print the word "Yes" (without the quotes).

In the second line print two integers k_1 and k_2 ($1 \leq k_1, k_2 \leq n$) — the number of servers used for each of the services.

In the third line print k_1 integers, the indices of the servers that will be used for the first service.

In the fourth line print k_2 integers, the indices of the servers that will be used for the second service.

No index may appear twice among the indices you print in the last two lines. If there are several possible answers, it is allowed to print any of them.

Examples

input

```
6 8 16
3 5 2 9 8 7
```

output

```
Yes
3 2
1 2 6
5 4
```

input

```
4 20 32
21 11 11 12
```

output

```
Yes
1 3
1
2 3 4
```

input

```
4 11 32
5 5 16 16
```

output

```
No
```

input

```
5 12 20
7 8 4 11 9
```

output

```
No
```

Note

In the first sample test each of the servers 1, 2 and 6 will will provide $8 / 3 = 2.(6)$ resource units and each of the servers 5, 4 will provide $16 / 2 = 8$ resource units.

In the second sample test the first server will provide 20 resource units and each of the remaining servers will provide $32 / 3 = 10.(6)$ resource units.

E. Big Secret

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

Vitya has learned that the answer for The Ultimate Question of Life, the Universe, and Everything is not the integer ~~54~~ 42, but an increasing integer sequence $a_1, \ldots, a_n$. In order to not reveal the secret earlier than needed, Vitya encrypted the answer and obtained the sequence $b_1, \ldots, b_n$ using the following rules:

- $b_1 = a_1$;
- $b_i = a_i \oplus a_{i - 1}$ for all i from 2 to n, where $x \oplus y$ is the bitwise XOR of x and y.

It is easy to see that the original sequence can be obtained using the rule $a_i = b_1 \oplus \ldots \oplus b_i$.

However, some time later Vitya discovered that the integers b_i in the cypher got shuffled, and it can happen that when decrypted using the rule mentioned above, it can produce a sequence that is not increasing. In order to save his reputation in the scientific community, Vasya decided to find some permutation of integers b_i so that the sequence $a_i = b_1 \oplus \ldots \oplus b_i$ is strictly increasing. Help him find such a permutation or determine that it is impossible.

Input

The first line contains a single integer $$$n$$$ ($$$1 \leq n \leq 10^5$$$).

The second line contains $$$n$$$ integers $$$b_1, \ldots, b_n$$$ ($$$1 \leq b_i < 2^{60}$$$).

Output

If there are no valid permutations, print a single line containing "`No`".

Otherwise in the first line print the word "`Yes`", and in the second line print integers $$$b'_1, \ldots, b'_n$$$ — a valid permutation of integers $$$b_i$$$. The unordered multisets $$$\{b_1, \ldots, b_n\}$$$ and $$$\{b'_1, \ldots, b'_n\}$$$ should be equal, i. e. for each integer $$$x$$$ the number of occurrences of $$$x$$$ in the first multiset should be equal to the number of occurrences of $$$x$$$ in the second multiset. Apart from this, the sequence $$$a_i = b'_1 \oplus \ldots \oplus b'_i$$$ should be strictly increasing.

If there are multiple answers, print any of them.

Examples

input
```
3
1 2 3
```
output
```
No
```

input
```
6
4 7 7 12 31 61
```
output
```
Yes
4 12 7 31 7 61
```

Note

In the first example no permutation is valid.

In the second example the given answer lead to the sequence $$$a_1 = 4$$$, $$$a_2 = 8$$$, $$$a_3 = 15$$$, $$$a_4 = 16$$$, $$$a_5 = 23$$$, $$$a_6 = 42$$$.

F. Aztec Catacombs

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

Indiana Jones found ancient Aztec catacombs containing a golden idol. The catacombs consists of $$$n$$$ caves. Each pair of caves is connected with a two-way corridor that can be opened or closed. The entrance to the catacombs is in the cave $$$1$$$, the idol and the exit are in the cave $$$n$$$.

When Indiana goes from a cave $$$x$$$ to a cave $$$y$$$ using an open corridor, all corridors connected to the cave $$$x$$$ change their state: all open corridors become closed, all closed corridors become open. Indiana wants to go from cave $$$1$$$ to cave $$$n$$$ going through as small number of corridors as possible. Help him find the optimal path, or determine that it is impossible to get out of catacombs.

Input

The first line contains two integers $$$n$$$ and $$$m$$$ ($$$2 \leq n \leq 3\cdot 10^5$$$, $$$0 \leq m \leq 3 \cdot 10^5$$$) — the number of caves and the number of open corridors at the initial moment.

The next $$$m$$$ lines describe the open corridors. The $$$i$$$-th of these lines contains two integers $$$u_i$$$ and $$$v_i$$$ ($$$1 \leq u_i, v_i \leq n$$$, $$$u_i \neq v_i$$$) — the caves connected by the $$$i$$$-th open corridor. It is guaranteed that each unordered pair of caves is presented at most once.

Output

If there is a path to exit, in the first line print a single integer $$$k$$$ — the minimum number of corridors Indians should pass through ($$$1 \leq k \leq 10^6$$$). In the second line print $$$k+1$$$ integers $$$x_0, \ldots, x_k$$$ — the number of caves in the order Indiana should visit them. The sequence $$$x_0, \ldots, x_k$$$ should satisfy the following:

- $$$x_0 = 1$$$, $$$x_k = n$$$;
- for each $$$i$$$ from $$$1$$$ to $$$k$$$ the corridor from $$$x_{i - 1}$$$ to $$$x_i$$$ should be open at the moment Indiana walks along this corridor.

If there is no path, print a single integer $$$-1$$$.

We can show that if there is a path, there is a path consisting of no more than $$$10^6$$$ corridors.

Examples

input

```
4 4
1 2
2 3
1 3
3 4
```

output

```
2
1 3 4
```

input

```
4 2
1 2
2 3
```

output

```
4
1 2 3 1 4
```