Day 7. Final Contest

A. Tree height

1 second, 512 megabytes

You are given a rooted tree. Calculate its height, maximum number of vertices on path from the root to a leaf.

Input
The first line contains number of vertices n ($1 \le n \le 10^5$). The second line contains n integer numbers from -1 to $n-1$ – parents of vertices. If i-th one is -1, vertex i is the root, else it's 0-based index of parent of i-th vertex. There is exactly one root.

Output
The only integer number – height of the tree.

input
5 4 -1 4 1 1
output
3

$O(n^2)$ solution will get TLE.

B. Ladder

2 seconds, 256 megabytes

A ladder has n steps numbered by $1, 2, ..., n$ from bottom to top. Each step has a number written on it. Starting from the bottom of the ladder (the step with the identifier 0) you have to reach the top of the ladder (the step with the identifier n). At each turn, you can go up by one or two steps. When we get to the top, we sum the numbers written on the visited steps. You have to maximize this sum.

Input
The first line of the input contains n ($1 \le n \le 100$) — the number of steps in the ladder. The second line contains n integers $a_1, a_2, ..., a_n$ ($-10\,000 \le a_i \le 10\,000$) — the numbers written on the steps.

Output
The sole line of the output should contain one integer — the maximal sum that can be obtained when climbing the ladder.

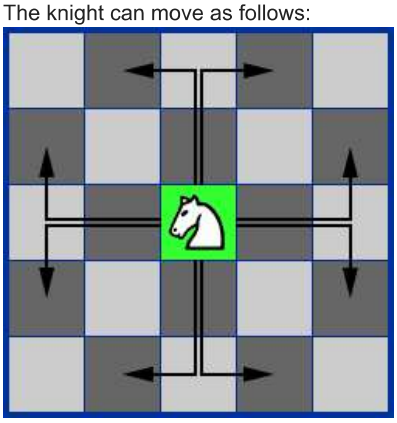
input
2 1 2
output
3

C. Knight 3

1 second, 256 megabytes

You are given two positions (x_s, y_s) and (x_f, y_f) on a 8×8 chessboard. There is a knight at position (x_s, y_s). Find the shortest knight path to position (x_f, y_f).

The knight can move as follows:



Input
The first line of input contains position (x_s, y_s).

The second line of input contains position (x_f, y_f).

Each position is described as a small letter 'a'–'h' and a digit 1–8.

Output
Print the shortest knight path from (x_s, y_s) to (x_f, y_f) in the same format as in the input.

input
a1 b1
output
a1 c2 a3 b1

D. Edges on Some Shortest Path

1 second, 256 megabytes

You are given a simple undirected connected graph with n vertices and m edges.

You are also given two vertices a and b.

Your task is to find all edges that belong to **at least one** shortest path between vertices a and b.

Input
The first line of the input contains four integers n, m, a and b ($2 \le n \le 10^5; n-1 \le m \le min(\frac{n(n-1)}{2}, 10^5; 1 \le a, b \le n; a \ne b$) — the number of vertices in the graph, the number of edges in the graph and vertices a and b, respectively.

The next m lines describe edges. The edge i is given as two integers x_i and y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$), where x_i and y_i are vertices that the i-th edge connects.

It is guaranteed that the given graph is undirected, simple and connected.

Output
In the first line, print one integer k ($1 \le k \le m$) — the number of edges that belong to at least one shortest path between a and b.

In the next k lines, print edges themselves. Each edge should be printed **once** (for example, if you printed the edge $(3, 5)$ then you shouldn't print the edge $(5, 3)$).

You can print edges in **any** order.

input
7 10 3 4 7 5 2 4 1 3 5 3 2 1 2 3 6 2 5 1 6 4 1 6

output
2 2 3 2 4

input
7 9 1 7 1 2 2 3 3 4 4 5 5 6 6 7 5 2 3 6 1 7

output
1 1 7

E. Colored Graph

1 second, 256 megabytes

You are given a simple undirected graph, where each edge of the graph have one of the three colors.

You have to find the **shortest** path from the vertex 1 to the vertex n in the graph such that there are no two consecutive edges **of the same color** in the path.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^5; n - 1 \le m \le min(\frac{n(n-1)}{2}, 10^5)$) — the number of vertices and the number of edges in the graph, respectively.

The next m lines describe edges. The edge i is given as three integers x_i, y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$) and c_i ($1 \le c_i \le 3$), where x_i and y_i are vertices that the i-th edge connects and c_i is the color of the i-th edge.

It is guaranteed that the given graph is undirected, simple and connected.

Output

If there is no required path in the given graph, print `NO` in the only line.

Otherwise, print `YES` in the first line. Then print one integer k in the second line — the number of vertices in the path. Then print k integers $v_1, v_2, \ldots, v_k$ in the third line — vertices of the path.

The first vertex v_1 should be 1, the last vertex v_k should be n and there should be no two consecutive edges of the same color in the path.

If there are multiple answers, you can print any.

input
4 4 1 2 1 2 3 2 3 4 3 2 4 1

output
YES 4 1 2 3 4

input
3 2 1 2 1 2 3 1

output
NO

F. Colored Graph But Cooler

1 second, 256 megabytes

You are given a simple undirected weighted graph, where each edge of the graph have one of the three colors (in addition to the weight).

You have to find the **shortest** path from the vertex 1 to the vertex n in the graph such that there are no two consecutive edges **of the same color** in the path.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^5; n - 1 \le m \le min(\frac{n(n-1)}{2}, 10^5)$) — the number of vertices and the number of edges in the graph, respectively.

The next m lines describe edges. The edge i is given as four integers x_i, y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$), c_i ($1 \le c_i \le 3$) and w_i ($1 \le w_i \le 10^4$), where x_i and y_i are vertices that the i-th edge connects, c_i is the color of the i-th edge and w_i is the weight of the i-th edge.

It is guaranteed that the given graph is undirected, simple and connected.

Output

If there is no required path in the given graph, print `NO` in the only line.

Otherwise, print `YES` in the first line. Then print one integer k in the second line — the number of vertices in the path. Then print k integers $v_1, v_2, \ldots, v_k$ in the third line — vertices of the path.

The first vertex v_1 should be 1, the last vertex v_k should be n and there should be no two consecutive edges of the same color in the path.

If there are multiple answers, you can print any.

input
4 4 1 2 1 1 2 3 2 2 3 4 3 1 2 4 2 4

output
YES 4 1 2 3 4

input
3 2 1 2 1 12 2 3 1 2
output
NO

G. Socks

2 seconds, 256 megabytes

Arseniy is already grown-up and independent. His mother decided to leave him alone for m days and left on a vacation. She have prepared a lot of food, left some money and washed all Arseniy's clothes.

Ten minutes before her leave she realized that it would be also useful to prepare instruction of which particular clothes to wear on each of the days she will be absent. Arseniy's family is a bit weird so all the clothes is enumerated. For example, each of Arseniy's n socks is assigned a unique integer from 1 to n. Thus, the only thing his mother had to do was to write down two integers l_i and r_i for each of the days — the indices of socks to wear on the day i (obviously, l_i stands for the left foot and r_i for the right). Each sock is painted in one of k colors.

When mother already left Arseniy noticed that according to instruction he would wear the socks of different colors on some days. Of course, that is a terrible mistake cause by a rush. Arseniy is a smart boy, and, by some magical coincidence, he posses k jars with the paint — one for each of k colors.

Arseniy wants to repaint some of the socks in such a way, that for each of m days he can follow the mother's instructions and wear the socks of the same color. As he is going to be very busy these days he will have no time to change the colors of any socks so he has to finalize the colors now.

The new computer game Bota-3 was just realised and Arseniy can't wait to play it. What is the minimum number of socks that need their color to be changed in order to make it possible to follow mother's instructions and wear the socks of the same color during each of m days.

Input

The first line of input contains three integers n, m and k ($2 \le n \le 200\,000$, $0 \le m \le 200\,000$, $1 \le k \le 200\,000$) — the number of socks, the number of days and the number of available colors respectively.

The second line contain n integers c_1, c_2, ..., c_n ($1 \le c_i \le k$) — current colors of Arseniy's socks.

Each of the following m lines contains two integers l_i and r_i ($1 \le l_i, r_i \le n$, $l_i \ne r_i$) — indices of socks which Arseniy should wear during the i-th day.

Output

Print one integer — the minimum number of socks that should have their colors changed in order to be able to obey the instructions and not make people laugh from watching the socks of different colors.

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

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

In the first sample, Arseniy can repaint the first and the third socks to the second color.

In the second sample, there is no need to change any colors.

H. Magnetic Triangles

2 seconds, 64 megabytes

In the Future City all building are skyscrapers. To move between them some triples of skyscrapers are connected with special magnetic triangles. Each triangle connects exactly three skyscrapers. It is possible to move from one skyscraper to another if they are directly or indirectly connected with triangles.

The system of magnetic triangles is organized in such a way that it is possible to get from any skyscraper to any other skyscraper.

However, operating magnetic triangles is very expensive so city authorities want to pick one of them and turn it off. They ask you to determine for each of the triangles, whether it will be possible to get from any skyscraper to any other skyscraper if this particular triangle will be turned off.

Input

The first line of the input file contains two integers n and m — the number of skyscrapers and the number of magnetic triangles respectively ($3 \le n \le 100\,000$, $1 \le m \le 100\,000$). Each of the next m lines contains three distinct integers — indices of skyscrapers connected by this triangle. Skyscrapers are numbered from 1 to n. It's guaranteed that it's possible to get from any skyscraper to any other skyscraper using only existing triangles.

Output

First print the number of magnetic triangles that can't be turned off without lose of connectivity. Then print their numbers. Triangles are numbered starting from 1 in the order they appear in the input file.

input
3 1 1 2 3
output
1 1

input
3 2 1 2 3 3 2 1
output
0

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

output

```
1
4
```

I. Lazy Coordinator

1 second, 512 megabytes

Boris is a contest coordinator on WWWforces, an online platform that conducts regular "What? Where? When?" competitions. There are two types of events in his life:

1. Someone submits a contest proposal (a set of questions) to Boris. He puts this new proposal in the waiting pool.
2. The time comes to organize a new contest. As Boris doesn't remember the initial order of proposals in his pool, he just picks a random proposal, removes it from the pool and uses it for the upcoming contest. Each proposal in the pool is picked with the same probability, regardless on how long it already waits.

There will be exactly n events of the first type and exactly n events of the second type. Moreover, for the k-th event of the second type, there will be at least k events of the first type preceding it. In other words, there will be at least one proposal in the pool any time the coordinator needs it. Finally, it is guaranteed that no two events happen simultaneously.

For each proposal, you should determine the expected time it will stay in the waiting pool.

Input

The first line of the input contains a single integer n ($1 \le n \le 100\,000$) — the number of events of each type.

Then $2n$ lines follow that describe the events. Each line is either "+ t_i" or "– t_i" meaning that an event of the first or the second type (respectively) occurs at the moment t_i.

It is guaranteed that all t_i are positive integers not exceeding 10^9, and the events will be given in the order of strictly increasing t_i. Moreover, for the k-th event of the second type, there will be at least k events of the first type preceding it. Finally, it is guaranteed that there will be exactly n events of the first type and exactly n events of the second type.

Output

For each proposal (event of the first type), print the expected time it will stay in the waiting pool. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

input

```
2
+ 1
+ 3
- 8
- 12
```

output

```
9.0000000000
7.0000000000
```

input

```
4
+ 1
- 2
+ 3
- 4
+ 5
- 6
+ 7
- 8
```

output

```
1.0000000000
1.0000000000
1.0000000000
1.0000000000
```

input

```
3
+ 4
+ 10
- 11
+ 16
- 20
- 100
```

output

```
31.5000000000
25.5000000000
44.0000000000
```

In the first sample, Boris receives all two proposals first and then uses them in two contests. Each proposal has the probability $\frac{1}{2}$ to be used in each of the contests. Thus, the expected waiting times are $(8-1)\cdot 0.5 + (12-1)\cdot 0.5 = 3.5 + 5.5 = 9$ for the first proposal and $(8-3)\cdot 0.5 + (12-3)\cdot 0.5 = 2.5 + 4.5 = 7$ for the second one.

In the second sample, each proposal is almost immediately used to conduct a contest.

J. Vanya and Jackets

2 seconds, 512 megabytes

Vanya attempted to change his life once again and decided to create a schedule of jackets he is going to wear during the next n days.

He read several manuals on jackets operation and found out that different jackets are designed for different temperature ranges. For each of his m jackets he determined values of l_i and r_i, the minimum and maximum temperature value that admits wearing the i-th jacket.

Vanya knows the weather forecast for the next n days, namely there will be the temperature of a_j during the j-th day. Since Vanya is sane enough, he will choose the appropriate jacket for each temperature, that is, on the j-th day he will wear any jacket i such that $l_i \le a_j \le r_i$. Also, Vanya tries to be really fashionable, so he never wears the same jacket for two consecutive days.

Given the fact that Vanya's mother does not allow him leaving home without the jacket or wearing several jackets, create a schedule of which jackets he should wear during the next n days satisfying all the requirements of Vanya.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$), the number of days and the number of jackets in Vanya's wardrobe respectively.

The second line of the input contains n integers a_i ($0 \le a_i \le 10^9$), the temperature at the i-th day.

Then m lines follow, i-th of them contains two integers l_i and r_i ($0 \le l_i \le r_i \le 10^9$), defining the temperature range of the i-th jacket.

Output

If there exists a way of choosing a jacket for each of the n days, output the word "Yes" (without the quotes) in the first line, and n numbers b_i in the second line, where b_i is the index of a jacket Vanya should wear in the i-th day. Otherwise output the only line containing the word "No" (without the quotes). Jackets are indexed starting from one in the order they appear in the input.

If there are several satisfying schedules, you are allowed to output any of them.

input
4 4 25 25 30 50 10 40 20 30 70 100 50 50
output
Yes 2 1 2 4

input
4 2 30 40 50 60 30 40 50 60
output
No

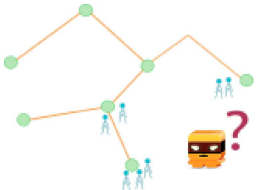
In the first sample the answer "2 1 2 4" is not the only possible, another correct answer is "1 2 1 4".

In the second sample there is no schedule satisfying the requirements of Vanya, since he is obligated to wear the first jackets for both the first and the second days.

K. Super M

2 seconds, 256 megabytes

Ari the monster is not an ordinary monster. She is the hidden identity of Super M, the Byteforces' superhero. Byteforces is a country that consists of n cities, connected by $n - 1$ bidirectional roads. Every road connects exactly two distinct cities, and the whole road system is designed in a way that one is able to go from any city to any other city using only the given roads. There are m cities being attacked by humans. So Ari... we meant Super M have to immediately go to each of the cities being attacked to scare those bad humans. Super M can pass from one city to another only using the given roads. Moreover, passing through one road takes her exactly one kron - the time unit used in Byteforces.



However, Super M is not on Byteforces now - she is attending a training camp located in a nearby country Codeforces. Fortunately, there is a special device in Codeforces that allows her to instantly teleport from Codeforces to any city of Byteforces. The way back is too long, so for the purpose of this problem teleportation is used exactly once.

You are to help Super M, by calculating the city in which she should teleport at the beginning in order to end her job in the minimum time (measured in krons). Also, provide her with this time so she can plan her way back to Codeforces.

Input

The first line of the input contains two integers n and m ($1 \le m \le n \le 123456$) - the number of cities in Byteforces, and the number of cities being attacked respectively.

Then follow $n - 1$ lines, describing the road system. Each line contains two city numbers u_i and v_i ($1 \le u_i, v_i \le n$) - the ends of the road i.

The last line contains m distinct integers - numbers of cities being attacked. These numbers are given in no particular order.

Output

First print the number of the city Super M should teleport to. If there are many possible optimal answers, print the one with the lowest city number.

Then print the minimum possible time needed to scare all humans in cities being attacked, measured in Krons.

Note that the correct answer is always unique.

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

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

In the first sample, there are two possibilities to finish the Super M's job in 3 krons. They are:

$2 \to 1 \to 3 \to 7$ and $7 \to 3 \to 1 \to 2$.

However, you should choose the first one as it starts in the city with the lower number.

L. Alpine Problem

5 seconds (13 seconds for Java), 512 megabytes

Arkady is tired of programming and algorithms so he decided to take a pause and completely change his everyday life. He always was fond of mountains so he used all the prize money he got from programming competitions to buy a small ski resort and is now preparing it for the upcoming winter season.

The resort consists of a single track which itself consists of n checkpoints, numbered from 1 to n. Checkpoint i is located at altitude h_i and no two checkpoints share the same altitude. The resort has only one elevator so skiers always start at checkpoint s and finish at checkpoint t. There are m pairs of checkpoints connected by a track segment that always leads from the higher of these two checkpoints down to the lower one.

The *attraction* of the track segment going directly from checkpoint u to checkpoint v is equal to the difference in their altitudes, i.e. $h_u - h_v$. The attraction of the route consecutively passing through checkpoints $v_1, v_2, ..., v_k$ is the **minimum** attraction of track segment forming this route, i.e. minimum value among $h_{v_1} - h_{v_2}, h_{v_2} - h_{v_3}, ..., h_{v_k} - h_{v_{k-1}}$.

Tourists visiting Arkady resort are interested in the attractiveness of the route and in its length that is defined as the number of track segments forming the route. Because Arkady isn't able to solve such problems, you are the one who has to find the solution. For each possible length of the route x from 1 to $n - 1$ compute the maximum possible attraction of the route from s to t that has length **at least** x.

Input

The first line of the input contains four integers n, m, s and t ($2 \le n \le 100\,000$, $1 \le m \le 200\,000$, $1 \le s, t \le n$, $s \ne t$), the number of checkpoints along the track, the number of track segments, the index of the start checkpoint and the index of the finish checkpoint respectively.

The second line contains n distinct integers $h_1, h_2, ..., h_n$ ($0 \le h_i \le 100\,000$) providing the altitudes of the corresponding checkpoints.

Then follow m lines describing track segments. The i-th of these lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), providing that the i-th track segment goes from checkpoint u_i to checkpoint v_i. It's guaranteed that no two track segments directly connect the same pair of checkpoints, that the altitude of checkpoint u_i is greater than altitude of v_i and that there exists at least one route from checkpoint s to checkpoint t.

Output

Print $n - 1$ integers. The i-th of them should be equal to the maximum possible attraction of the route from s to t that has length of at least i. If for some i there is no path of length greater than or equal to i print -1 in the corresponding line of the output.

input

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

output

```
3
1
1
```

input

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

output

```
2
-1
```

input

```
5 10 1 5
8 6 4 3 1
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
4 5
```

output

```
7
3
2
1
```

In the first sample, there exists a direct track from start checkpoint to finish checkpoint. The attraction of this route is 3, while the length is 1. Also there exists a route of length 3 that passes all the checkpoints in order 2, 1, 3 and 4 and has attraction 1. For $x = 2$ the answer is also this path of length 3 as this is the most attractive path of length greater than or equal to 2.

M. The Sting

2 seconds, 512 megabytes

Recently Ostap Bender (famous Russian fictional character) came up with a 401-st relatively legitimate way of acquiring money from people, and, what is most important: it is absolutely safe! He noticed that people of Stargorod are huge fans of a local team "Star Dogs" in a Professional Football League of Fictional Cities, and they love making bets on football game outcome.

Ostap has already received n proposals of bets on outcome of the next game. Each of the proposers chooses one of the three possible game outcomes: either favourite Stargorod team wins, loses or there will be a draw. In case game outcome matches the chosen, Ostap has to pay the bet proposer a_i burles. In case of any other game outcome the bet proposer pays Ostap b_i burles.

Ostap's plan is to accept some of the proposals he received and decline the remaining proposals. Ostap wants to choose such subset of proposals that maximizes his profit **in the worst case**.

Input

The first line of the input contains one integer n ($1 \le n \le 500$), the number of bet proposals Ostap received.

Each of the following n lines contains one bet description, consisting of a character t_i and two integers a_i and b_i ($1 \le a_i, b_i \le 500$), where t_i is "W", "D" or "L" depending on if the proposer believes in win, lose or draw of a Stargorod favorite tea, a_i is amount of money Ostap has to pay in case chosen outcome happens and b_i is amount of money the proposer has to pay Ostap in remaining cases.

Output

Output the maximum possible worst-case profit Ostap can have if he chooses the optimal subset.

input

```
3
W 2 2
D 2 2
L 2 2
```

output

```
2
```

input

```
3
W 3 1
D 3 1
L 3 1
```

output

```
0
```

input

```
5
W 3 3
W 2 2
D 10 100
L 3 1
D 3 3
```

output

```
1
```

N. FoodSberry

1 second, 512 megabytes

With the speed of light are food technologies conquering cities and towns of Berland, and, especially, its gorgeous capital City N.

FoodBerry is the largest food delivery that operates in City N. Consider the city as a two-dimensional square with sides parallel to coordinate axis. The bottom left corner of this square has coordinates $(-10^4, -10^4)$ and the top right corner has coordinates $(10^4, 10^4)$. All coordinates are given in meters.

FoodBerry has one Super Large Warehouse located at $(0, 0)$ and n dark stores (smaller warehouses), the i-th of them is located at point (x_i, y_i). Each dark store has two types of delivery: by foot and by car. Delivery by foot can be used for all points that are located no more than a meters from this dark store, while delivery by car can be used for up to b meters distances ($a \le b$, obviously). During one day each dark store is able to execute no more than c deliveries. Moreover, due to the limitation of vehicles, no more than d of these deliveries can be done by car ($d \le c$).

Super Large Warehouse is located at $(0, 0)$ and is capable of performing any number of deliveries to any locations in the city. However, using this option is expensive and FoodBerry tries to avoid it to for as long as possible.

In this problem we are going to analyze one day of FoodBerry in City N. There were m orders during that day, they are given in the order they were made and processed. The i-th order asks for the delivery to point (x'_i, y'_i). For the purpose of this problem we introduce some bold assumptions.

1. First all orders were received. Then all deliveries were executed.
2. No two orders appear in the same moment of time.
3. After each order appears, there is enough time to re-process everything. Scheduling center assumes there will be no more orders today and makes a distribution of orders between dark stores (and the warehouse). Moreover, for each delivery it is determined whether it should be done by foot or by car. This distribution is completely re-done after any new order appears, it doesn't need to depend on the previous distribution in any way.

Of course, each distribution is done with respect to limitations on distance and quantity of orders per dark store given by parameters a, b, c and d. If there is a distribution of orders that satisfies all the requirements and executes all the deliveries without using Super Large Warehouse, the scheduling program goes for it.

Assuming all distributions are made optimal, what is the minimum order index i such that there will be no way to make a valid distribution without using Super Large Warehouse?

Input

The first line of the input contains six integers n, m, a, b, c and d ($1 \le n, m \le 500$, $1 \le a \le b \le 30\,000$, $1 \le d \le c \le 1000$). Here n is the number of dark stores in City N, m is the number of orders to process, a is the maximum possible distance for delivery from dark store by foot, b is the maximum possible distance for delivery from dark store by car, c is the maximum possible total number of deliveries a dark store can execute a day, and d is the maximum possible number of deliveries by car a dark store can execute a day.

Then follow n lines containing two integers x_i and y_i ($-10\,000 \le x_i, y_i \le 10\,000$) each — coordinates of dark stores.

Finally go m lines with coordinates of orders. Each of them contains two integers x'_i and y'_i ($-10\,000 \le x'_i, y'_i \le 10\,000$).

Any points of the input can coincide, i.e. there can be two or more dark stores in the same point, two or more orders in the same point, a dark store and an order in the same point, a dark store or an order at point $(0, 0)$ and so on.

It is possible that some order is located out of reach of any dark store. Note that it is still possible to execute all the orders as FoodSberry has Super Large Warehous that is capable of doing any number of deliveries to any location in City N.

Output

If it is possible to obey all the requirements and satisfy all m orders without using Super Large Warehous, print `-1` in the only line of the output.

Otherwise, print minimum possible index i such that Super Large Warehouse is required to satisfy first i orders.

input

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

output

```
3
```

input

```
3 6 1 1 2 2
0 1
-2 1
2 1
-1 1
1 1
0 2
0 0
-2 1
2 1
```

output

```
-1
```

O. XOR and Favorite Number

4 seconds, 256 megabytes

Bob has a favorite number k and a_i of length n. Now he asks you to answer m queries. Each query is given by a pair l_i and r_i and asks you to count the number of pairs of integers i and j, such that $l \le i \le j \le r$ and the xor of the numbers $a_i, a_{i+1}, ..., a_j$ is equal to k.

Input

The first line of the input contains integers n, m and k ($1 \le n, m \le 100\,000$, $0 \le k \le 1\,000\,000$) — the length of the array, the number of queries and Bob's favorite number respectively.

The second line contains n integers a_i ($0 \le a_i \le 1\,000\,000$) — Bob's array.

Then m lines follow. The i-th line contains integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the parameters of the i-th query.

Output

Print m lines, answer the queries in the order they appear in the input.

input
6 2 3 1 2 1 1 0 3 1 6 3 5

output
7 0

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

output
9 4 4

In the first sample the suitable pairs of i and j for the first query are: $(1, 2)$, $(1, 4)$, $(1, 5)$, $(2, 3)$, $(3, 6)$, $(5, 6)$, $(6, 6)$. Not a single of these pairs is suitable for the second query.

In the second sample xor equals 1 for all subarrays of an odd length.

P. Second Shortest Path

1 second, 256 megabytes

You are given a simple **directed** weighted graph. All edges weights are positive integers.

Your task is to find the **second by weight** shortest path from the vertex 1 to the vertex n. If all paths from the vertex 1 to the vertex n have the same weights, print -1.

It is guaranteed that there is at least one path from the vertex 1 to the vertex n.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^5; n - 1 \le m \le min(\frac{n(n-1)}{2}, 10^5)$) — the number of vertices and the number of edges in the graph, respectively.

The next m lines describe edges. The edge i is given as three integers x_i, y_i ($1 \le x_i, y_i \le n; x_i \ne y_i$) and w_i ($1 \le w_i \le 10^4$), where x_i and y_i are vertices that the i-th edge connects and w_i is the weight of the i-th edge.

It is guaranteed that the given graph is directed and simple.

It is guaranteed that there is at least one path from the vertex 1 to the vertex n.

Output

If all paths from the vertex 1 to the vertex n have the same weights, print -1. Otherwise, print the weight of the second by weight shortest path from the vertex 1 to the vertex n.

input
6 8 1 3 6 1 5 1 1 6 2 2 5 3 2 6 7 3 4 100 3 5 1 5 6 4

output
5

input
3 2 1 2 10000 2 3 10000

output
-1

The picture corresponding to the first example: