Codeforces Round #408 (Div. 2)

A. Buying A House

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

Zane the wizard had never loved anyone before, until he fell in love with a girl, whose name remains unknown to us.

The girl lives in house m of a village. There are n houses in that village, lining in a straight line from left to right: house 1, house 2, ..., house n. The village is also well-structured: house i and house $i + 1$ ($1 \le i < n$) are exactly 10 meters away. In this village, some houses are occupied, and some are not. Indeed, unoccupied houses can be purchased.

You will be given n integers $a_1, a_2, ..., a_n$ that denote the availability and the prices of the houses. If house i is occupied, and therefore cannot be bought, then a_i equals 0. Otherwise, house i can be bought, and a_i represents the money required to buy it, in dollars.

As Zane has only k dollars to spare, it becomes a challenge for him to choose the house to purchase, so that he could live as near as possible to his crush. Help Zane determine the minimum distance from his crush's house to some house he can afford, to help him succeed in his love.

Input

The first line contains three integers n, m, and k ($2 \le n \le 100$, $1 \le m \le n$, $1 \le k \le 100$) — the number of houses in the village, the house where the girl lives, and the amount of money Zane has (in dollars), respectively.

The second line contains n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 100$) — denoting the availability and the prices of the houses.

It is guaranteed that $a_m = 0$ and that it is possible to purchase some house with no more than k dollars.

Output

Print one integer — the minimum distance, in meters, from the house where the girl Zane likes lives to the house Zane can buy.

Examples

input
5 1 20 0 27 32 21 19
output
40

input
7 3 50 62 0 0 0 99 33 22
output
30

input
10 5 100 1 0 1 0 0 0 0 0 1 1
output
20

Note

In the first sample, with $k = 20$ dollars, Zane can buy only house 5. The distance from house $m = 1$ to house 5 is $10 + 10 + 10 + 10 = 40$ meters.

In the second sample, Zane can buy houses 6 and 7. It is better to buy house 6 than house 7, since house $m = 3$ and house 6 are only 30 meters away, while house $m = 3$ and house 7 are 40 meters away.

B. Find The Bone

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

Zane the wizard is going to perform a magic show shuffling the cups.

There are n cups, numbered from 1 to n, placed along the x-axis on a table that has m holes on it. More precisely, cup i is on the table at the position $x = i$.

The problematic bone is initially at the position $x = 1$. Zane will confuse the audience by swapping the cups k times, the i-th time of which involves the cups at the positions $x = u_i$ and $x = v_i$. If the bone happens to be at the position where there is a hole at any time, it will fall into the hole onto the ground and will not be affected by future swapping operations.

Do not forget that Zane is a wizard. When he swaps the cups, he does not move them ordinarily. Instead, he teleports the cups (along with the bone, if it is inside) to the intended positions. Therefore, for example, when he swaps the cup at $x = 4$ and the one at $x = 6$, they will not be at the position $x = 5$ at any moment during the operation.

Zane's puppy, Inzane, is in trouble. Zane is away on his vacation, and Inzane cannot find his beloved bone, as it would be too exhausting to try opening all the cups. Inzane knows that the Codeforces community has successfully helped Zane, so he wants to see if it could help him solve his problem too. Help Inzane determine the final position of the bone.

Input

The first line contains three integers n, m, and k ($2 \le n \le 10^6$, $1 \le m \le n$, $1 \le k \le 3 \cdot 10^5$) — the number of cups, the number of holes on the table, and the number of swapping operations, respectively.

The second line contains m **distinct** integers $h_1, h_2, \ldots, h_m$ ($1 \le h_i \le n$) — the positions along the x-axis where there is a hole on the table.

Each of the next k lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — the positions of the cups to be swapped.

Output

Print one integer — the final position along the x-axis of the bone.

Examples

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

input
5 1 2 2 1 2 2 4
output
2

Note

In the first sample, after the operations, the bone becomes at $x = 2$, $x = 5$, $x = 7$, and $x = 1$, respectively.

In the second sample, after the first operation, the bone becomes at $x = 2$, and falls into the hole onto the ground.

C. Bank Hacking

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

Although Inzane successfully found his beloved bone, Zane, his owner, has yet to return. To search for Zane, he would need a lot of money, of which he sadly has none. To deal with the problem, he has decided to hack the banks.

There are n banks, numbered from 1 to n. There are also $n - 1$ wires connecting the banks. All banks are initially *online*. Each bank also has its initial strength: bank i has initial strength a_i.

Let us define some keywords before we proceed. Bank i and bank j are *neighboring* if and only if there exists a wire directly connecting them. Bank i and bank j are *semi-neighboring* if and only if there exists an ***online*** bank k such that bank i and bank k are *neighboring* and bank k and bank j are *neighboring*.

When a bank is hacked, it becomes *offline* (and no longer *online*), and other banks that are *neighboring* or *semi-neighboring* to it have their strengths increased by 1.

To start his plan, Inzane will choose a bank to hack first. Indeed, the strength of such bank must not exceed the strength of his computer. After this, he will repeatedly choose some bank to hack next until all the banks are hacked, but he can continue to hack bank x if and only if all these conditions are met:

1. Bank x is *online*. That is, bank x is not hacked yet.
2. Bank x is *neighboring* to some *offline* bank.
3. The strength of bank x is less than or equal to the strength of Inzane's computer.

Determine the minimum strength of the computer Inzane needs to hack all the banks.

Input

The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$) — the total number of banks.

The second line contains n integers $a_1, a_2, ..., a_n$ ($-10^9 \le a_i \le 10^9$) — the strengths of the banks.

Each of the next $n - 1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$) — meaning that there is a wire directly connecting banks u_i and v_i.

It is guaranteed that the wires connect the banks in such a way that Inzane can somehow hack all the banks using a computer with appropriate strength.

Output

Print one integer — the minimum strength of the computer Inzane needs to accomplish the goal.

Examples

input

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

output

```
5
```

input

```
7
38 -29 87 93 39 28 -55
1 2
2 5
3 2
2 4
1 7
7 6
```

output

```
93
```

input

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

output

```
8
```

Note

In the first sample, Inzane can hack all banks using a computer with strength 5. Here is how:

- Initially, strengths of the banks are $[1, 2, 3, 4, 5]$.
- He hacks bank 5, then strengths of the banks become $[1, 2, 4, 5, -]$.
- He hacks bank 4, then strengths of the banks become $[1, 3, 5, -, -]$.
- He hacks bank 3, then strengths of the banks become $[2, 4, -, -, -]$.
- He hacks bank 2, then strengths of the banks become $[3, -, -, -, -]$.
- He completes his goal by hacking bank 1.

In the second sample, Inzane can hack banks 4, 2, 3, 1, 5, 7, and 6, in this order. This way, he can hack all banks using a computer with strength 93.

D. Police Stations

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

Inzane finally found Zane with a lot of money to spare, so they together decided to establish a country of their own.

Ruling a country is not an easy job. Thieves and terrorists are always ready to ruin the country's peace. To fight back, Zane and Inzane have enacted a very effective law: from each city it must be possible to reach a police station by traveling at most d kilometers along the roads.

There are n cities in the country, numbered from 1 to n, connected only by exactly $n - 1$ roads. All roads are 1 kilometer long. It is initially possible to travel from a city to any other city using these roads. The country also has k police stations located in some cities. In particular, the city's structure satisfies the requirement enforced by the previously mentioned law. Also note that there can be multiple police stations in one city.

However, Zane feels like having as many as $n - 1$ roads is unnecessary. The country is having financial issues, so it wants to minimize the road maintenance cost by shutting down as many roads as possible.

Help Zane find the maximum number of roads that can be shut down without breaking the law. Also, help him determine such roads.

Input

The first line contains three integers n, k, and d ($2 \le n \le 3 \cdot 10^5$, $1 \le k \le 3 \cdot 10^5$, $0 \le d \le n - 1$) — the number of cities, the number of police stations, and the distance limitation in kilometers, respectively.

The second line contains k integers $p_1, p_2, \ldots, p_k$ ($1 \le p_i \le n$) — each denoting the city each police station is located in.

The i-th of the following $n - 1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$) — the cities directly connected by the road with index i.

It is guaranteed that it is possible to travel from one city to any other city using only the roads. Also, it is possible from any city to reach a police station within d kilometers.

Output

In the first line, print one integer s that denotes the maximum number of roads that can be shut down.

In the second line, print s distinct integers, the indices of such roads, in any order.

If there are multiple answers, print any of them.

Examples

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

output
1 5

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

output
2 4 5

Note

In the first sample, if you shut down road 5, all cities can still reach a police station within $k = 4$ kilometers.

In the second sample, although this is the only largest valid set of roads that can be shut down, you can print either `4 5` or `5 4` in the second line.

E. Exam Cheating

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

Zane and Zane's crush have just decided to date! However, the girl is having a problem with her Physics final exam, and needs your help.

There are n questions, numbered from 1 to n. Question i comes before question $i+1$ $(1 \le i < n)$. Each of the questions cannot be guessed on, due to the huge penalty for wrong answers. The girl luckily sits in the middle of two geniuses, so she is going to cheat.

However, the geniuses have limitations. Each of them may or may not know the answers to some questions. Anyway, it is safe to assume that the answers on their answer sheets are absolutely correct.

To make sure she will not get caught by the proctor, the girl will glance **at most** p times, each time looking at **no more than** k consecutive questions on one of the two geniuses' answer sheet. When the girl looks at some question on an answer sheet, she copies the answer to that question if it is on that answer sheet, or does nothing otherwise.

Help the girl find the maximum number of questions she can get correct.

Input

The first line contains three integers n, p, and k $(1 \le n, p \le 1,000, 1 \le k \le min(n, 50))$ — the number of questions, the maximum number of times the girl can glance, and the maximum number of consecutive questions that can be looked at in one time glancing, respectively.

The second line starts with one integer r $(0 \le r \le n)$, denoting the number of questions the first genius has answered on his answer sheet. Then follow r integers $a_1, a_2, ..., a_r$ $(1 \le a_i \le n)$ — the answered questions, given in a strictly-increasing order (that is, $a_i < a_{i+1}$).

The third line starts with one integer s $(0 \le s \le n)$, denoting the number of questions the second genius has answered on his answer sheet. Then follow s integers $b_1, b_2, ..., b_s$ $(1 \le b_i \le n)$ — the answered questions, given in a strictly-increasing order (that is, $b_i < b_{i+1}$).

Output

Print one integer — the maximum number of questions the girl can answer correctly.

Examples

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

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

Note

Let (x, l, r) denote the action of looking at all questions i such that $l \le i \le r$ on the answer sheet of the x-th genius.

In the first sample, the girl could get 4 questions correct by performing sequence of actions $(1, 1, 3)$ and $(2, 5, 6)$.

In the second sample, the girl could perform sequence of actions $(1, 3, 5)$, $(2, 2, 4)$, and $(2, 6, 8)$ to get 7 questions correct.

F. Sequence Recovery

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

Zane once had a *good* sequence a consisting of n integers $a_1, a_2, ..., a_n$ — but he has lost it.

A sequence is said to be *good* if and only if all of its integers are non-negative and do not exceed 10^9 in value.

However, Zane remembers having played around with his sequence by applying m operations to it.

There are two types of operations:

1. Find the maximum value of integers with indices i such that $l \leq i \leq r$, given l and r.

2. Assign d as the value of the integer with index k, given k and d.

After he finished playing, he restored his sequence to the state it was before any operations were applied. That is, sequence a was no longer affected by the applied type 2 operations. Then, he lost his sequence at some time between now and then.

Fortunately, Zane remembers all the operations and the order he applied them to his sequence, along with the **distinct** results of all type 1 operations. Moreover, among all *good* sequences that would produce the same results when the same operations are applied in the same order, he knows that his sequence a has the greatest *cuteness*.

We define *cuteness* of a sequence as the bitwise OR result of all integers in such sequence. For example, the *cuteness* of Zane's sequence a is a_1 OR a_2 OR ... OR a_n.

Zane understands that it might not be possible to recover exactly the lost sequence given his information, so he would be happy to get any *good* sequence b consisting of n integers $b_1, b_2, ..., b_n$ that:

1. would give the same results when the same operations are applied in the same order, and

2. has the same *cuteness* as that of Zane's original sequence a.

If there is such a sequence, find it. Otherwise, it means that Zane must have remembered something incorrectly, which is possible.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 3 \cdot 10^5$) — the number of integers in Zane's original sequence and the number of operations that have been applied to the sequence, respectively.

The i-th of the following m lines starts with one integer t_i () — the type of the i-th operation.

If the operation is type 1 ($t_i = 1$), then three integers l_i, r_i, and x_i follow ($1 \leq l_i \leq r_i \leq n$, $0 \leq x_i \leq 10^9$) — the leftmost index to be considered, the rightmost index to be considered, and the maximum value of all integers with indices between l_i and r_i, inclusive, respectively.

If the operation is type 2 ($t_i = 2$), then two integers k_i and d_i follow ($1 \leq k_i \leq n$, $0 \leq d_i \leq 10^9$) — meaning that the integer with index k_i should become d_i after this operation.

It is guaranteed that $x_i \neq x_j$ for all pairs (i, j) where $1 \leq i < j \leq m$ and $t_i = t_j = 1$.

The operations are given in the same order they were applied. That is, the operation that is given first was applied first, the operation that is given second was applied second, and so on.

Output

If there does not exist a valid *good* sequence, print "NO" (without quotation marks) in the first line.

Otherwise, print "YES" (without quotation marks) in the first line, and print n space-separated integers $b_1, b_2, ..., b_n$ ($0 \leq b_i \leq 10^9$) in the second line.

If there are multiple answers, print any of them.

Examples

input
5 4 1 1 5 19 1 2 5 1 2 5 100 1 1 5 100
output
YES 19 0 0 0 1

input
5 2

```
1 1 5 0
1 1 5 100
```

output

```
NO
```

Note

In the first sample, it is easy to verify that this *good* sequence is valid. In particular, its *cuteness* is $19 \text{ OR } 0 \text{ OR } 0 \text{ OR } 0 \text{ OR } 1 = 19$.

In the second sample, the two operations clearly contradict, so there is no such *good* sequence.