Codeforces Round #121 (Div. 1)

A. Dynasty Puzzles

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

The ancient Berlanders believed that the longer the name, the more important its bearer is. Thus, Berland kings were famous for their long names. But long names are somewhat inconvenient, so the Berlanders started to abbreviate the names of their kings. They called every king by the first letters of its name. Thus, the king, whose name was Victorious Vasily Pupkin, was always called by the berlanders VVP.

In Berland over its long history many dynasties of kings replaced each other, but they were all united by common traditions. Thus, according to one Berland traditions, to maintain stability in the country, the first name of the heir should be the same as the last name his predecessor (hence, the first letter of the abbreviated name of the heir coincides with the last letter of the abbreviated name of the predecessor). Berlanders appreciate stability, so this tradition has never been broken. Also Berlanders like perfection, so another tradition requires that the first name of the first king in the dynasty coincides with the last name of the last king in this dynasty (hence, the first letter of the abbreviated name of the first king coincides with the last letter of the abbreviated name of the last king). This tradition, of course, has also been always observed.

The name of a dynasty is formed by very simple rules: we take all the short names of the kings in the order in which they ruled, and write them in one line. Thus, a dynasty of kings "ab" and "ba" is called "abba", and the dynasty, which had only the king "abca", is called "abca".

Vasya, a historian, has recently found a list of abbreviated names of all Berland kings and their relatives. Help Vasya to find the maximally long name of the dynasty that could have existed in Berland.

Note that in his list all the names are ordered by the time, that is, if name A is earlier in the list than B, then if A and B were kings, then king A ruled before king B.

Input

The first line contains integer n ($1 \le n \le 5 \cdot 10^5$) — the number of names in Vasya's list. Next n lines contain n abbreviated names, one per line. An abbreviated name is a non-empty sequence of lowercase Latin letters. Its length does not exceed 10 characters.

Output

Print a single number — length of the sought dynasty's name in letters.

If Vasya's list is wrong and no dynasty can be found there, print a single number 0.

Sample test(s)

input
3 abc ca cba
output
6

input
4 vvp vvp dam vvp
output
0

input
3 ab c def
output
1

Note

In the first sample two dynasties can exist: the one called "abcca" (with the first and second kings) and the one called "abccba" (with the first and third kings).

In the second sample there aren't acceptable dynasties.

The only dynasty in the third sample consists of one king, his name is "c".

B. Demonstration

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

In the capital city of Berland, Bertown, demonstrations are against the recent election of the King of Berland. Berland opposition, led by Mr. Ovalny, believes that the elections were not fair enough and wants to organize a demonstration at one of the squares.

Bertown has n squares, numbered from 1 to n, they are numbered in the order of increasing distance between them and the city center. That is, square number 1 is central, and square number n is the farthest from the center. Naturally, the opposition wants to hold a meeting as close to the city center as possible (that is, they want an square with the minimum number).

There are exactly k $(k < n)$ days left before the demonstration. Now all squares are free. But the Bertown city administration never sleeps, and the approval of an application for the demonstration threatens to become a very complex process. The process of approval lasts several days, but every day the following procedure takes place:

- The opposition shall apply to hold a demonstration at a free square (the one which isn't used by the administration).
- The administration tries to move the demonstration to the worst free square left. To do this, the administration organizes some long-term activities on the square, which is specified in the application of opposition. In other words, the administration starts using the square and it is no longer free. Then the administration proposes to move the opposition demonstration to the worst free square. If the opposition has applied for the worst free square **then request is accepted and administration doesn't spend money**. If the administration does not have enough money to organize an event on the square in question, the opposition's application is accepted. If administration doesn't have enough money to organize activity, then rest of administration's money spends and application is accepted
- If the application is not accepted, then the opposition can agree to the administration's proposal (that is, take the worst free square), or withdraw the current application and submit another one the next day. If there are no more days left before the meeting, the opposition has no choice but to agree to the proposal of City Hall. If application is accepted opposition can reject it. It means than opposition still can submit more applications later, **but square remains free**.

In order to organize an event on the square i, the administration needs to spend a_i bourles. Because of the crisis the administration has only b bourles to confront the opposition. What is the best square that the opposition can take, if the administration will keep trying to occupy the square in question each time? Note that the administration's actions always depend only on the actions of the opposition.

Input

The first line contains two integers n and k — the number of squares and days left before the meeting, correspondingly ($1 \le k < n \le 10^5$).

The second line contains a single integer b — the number of bourles the administration has ($1 \le b \le 10^{18}$).

The third line contains n space-separated integers a_i — the sum of money, needed to organise an event on square i ($1 \le a_i \le 10^9$).

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print a single number — the minimum number of the square where the opposition can organize the demonstration.

Sample test(s)

input
5 2 8 2 4 5 3 1
output
2

input
5 2 8 3 2 4 1 5
output
5

input
5 4 1000000000000000 5 4 3 2 1
output
5

Note

In the first sample the opposition can act like this. On day one it applies for square 3. The administration has to organize an event there and end up with 3 bourles. If on the second day the opposition applies for square 2, the administration won't have the money to intervene.

In the second sample the opposition has only the chance for the last square. If its first move occupies one of the first four squares, the administration is left with at least 4 bourles, which means that next day it can use its next move to move the opposition from any square to the last one.

In the third sample administration has a lot of money, so opposition can occupy only last square.

C. Fools and Roads

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

They say that Berland has exactly two problems, fools and roads. Besides, Berland has n cities, populated by the fools and connected by the roads. All Berland roads are bidirectional. As there are many fools in Berland, between each pair of cities there is a path (or else the fools would get upset). Also, between each pair of cities there is no more than one simple path (or else the fools would get lost).

But that is not the end of Berland's special features. In this country fools sometimes visit each other and thus spoil the roads. The fools aren't very smart, so they always use only the simple paths.

A simple path is the path which goes through every Berland city not more than once.

The Berland government knows the paths which the fools use. Help the government count for each road, how many distinct fools can go on it.

Note how the fools' paths are given in the input.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of cities.

Each of the next n - 1 lines contains two space-separated integers u_i, v_i ($1 \le u_i$, $v_i \le n$, $u_i \ne v_i$), that means that there is a road connecting cities u_i and v_i.

The next line contains integer k ($0 \le k \le 10^5$) — the number of pairs of fools who visit each other.

Next k lines contain two space-separated numbers. The i-th line ($i > 0$) contains numbers a_i, b_i ($1 \le a_i$, $b_i \le n$). That means that the fool number $2i$ - 1 lives in city a_i and visits the fool number $2i$, who lives in city b_i. The given pairs describe simple paths, because between every pair of cities there is only one simple path.

Output

Print n - 1 integer. The integers should be separated by spaces. The i-th number should equal the number of fools who can go on the i-th road. The roads are numbered starting from one in the order, in which they occur in the input.

Sample test(s)

input

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

output

```
2 1 1 1
```

input

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

output

```
3 1 1 1
```

Note

In the first sample the fool number one goes on the first and third road and the fool number 3 goes on the second, first and fourth ones.

In the second sample, the fools number 1, 3 and 5 go on the first road, the fool number 5 will go on the second road, on the third road goes the fool number 3, and on the fourth one goes fool number 1.

D. Metro Scheme

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

Berland is very concerned with privacy, so almost all plans and blueprints are secret. However, a spy of the neighboring state managed to steal the Bertown subway scheme.

The Bertown Subway has n stations, numbered from 1 to n, and m bidirectional tunnels connecting them. All Bertown Subway consists of lines. To be more precise, there are two types of lines: circular and radial.

A *radial line* is a sequence of stations $v_1, ..., v_k$ $(k > 1)$, where stations v_i and v_{i+1} $(i < k)$ are connected by a tunnel and no station occurs in the line more than once ($v_i \neq v_j$ for $i \neq j$).

A *loop line* is a series of stations, $v_1, ..., v_k$ $(k > 2)$, where stations v_i и v_{i+1} are connected by a tunnel. In addition, stations v_1 and v_k are also connected by a tunnel. No station is occurs in the loop line more than once.

Note that a single station can be passed by any number of lines.

According to Berland standards, there can't be more than one tunnel between two stations and each tunnel belongs to exactly one line. Naturally, each line has at least one tunnel. Between any two stations there is the way along the subway tunnels. In addition, in terms of graph theory, a subway is a vertex cactus: if we consider the subway as a graph in which the stations are the vertexes and the edges are tunnels, then each vertex lies on no more than one simple cycle.

Unfortunately, scheme, stolen by the spy, had only the stations and the tunnels. It was impossible to determine to which line every tunnel corresponds. But to sabotage successfully, the spy needs to know what minimum and maximum number of lines may be in the Bertown subway.

Help him!

Input

The first line contains two integers n and m ($1 \le n \le 10^5$, $0 \le m \le 3 \cdot 10^5$) — the number of stations and the number of tunnels, correspondingly.

Each of the next m lines contain two integers — the numbers of stations connected by the corresponding tunnel. The stations are numbered with integers from 1 to n.

It is guaranteed that the graph that corresponds to the subway has no multiple edges or loops, it is connected and it is a vertex cactus.

Output

Print two numbers — the minimum and maximum number of lines correspondingly.

Sample test(s)

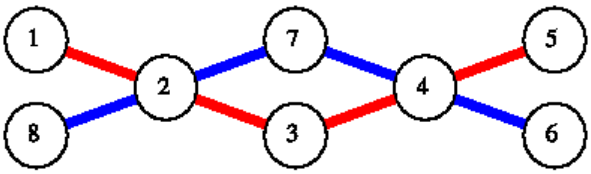
input
```
3 3
1 2
2 3
3 1
```
output
```
1 3
```

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

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

Note

The subway scheme with minimum possible number of lines for the second sample is:



E. Thwarting Demonstrations

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

It is dark times in Berland. Berlyand opposition, funded from a neighboring state, has organized a demonstration in Berland capital Bertown. Through the work of intelligence we know that the demonstrations are planned to last for k days.

Fortunately, Berland has a special police unit, which can save the country. It has exactly n soldiers numbered from 1 to n. Berland general, the commander of the detachment, must schedule the detachment's work in these difficult k days. In each of these days, the general must send a certain number of police officers to disperse riots. Since the detachment is large and the general is not very smart, he can only select a set of all soldiers numbered from l to r, inclusive, where l and r are selected arbitrarily.

Now the general has exactly two problems. First, he cannot send the same group twice — then soldiers get bored and they rebel. Second, not all soldiers are equally reliable. Every soldier has a reliability of a_i. The reliability of the detachment is counted as the sum of reliabilities of soldiers in it. The reliability of a single soldier can be negative, then when you include him in the detachment, he will only spoil things. The general is distinguished by his great greed and shortsightedness, so each day he sends to the dissolution the most reliable group of soldiers possible (that is, of all the groups that have not been sent yet).

The Berland Government has decided to know what would be the minimum reliability of the detachment, sent to disperse the demonstrations during these k days. The general himself can not cope with such a difficult task. Help him to not embarrass himself in front of his superiors!

Input

The first line contains two integers n and k $(1 \leqslant n \leqslant 10^5, 1 \leqslant k \leqslant \frac{n \cdot (n+1)}{2})$ — the number of soldiers in the detachment and the number of times somebody goes on duty.

The second line contains n space-separated integers a_i, their absolute value doesn't exceed 10^9 — the soldiers' reliabilities.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++, it is preferred to use `cin`, `cout` streams of the `%I64d` specifier.

Output

Print a single number — the sought minimum reliability of the groups that go on duty during these k days.

Sample test(s)

input
3 4 1 4 2
output
4

input
4 6 2 -1 2 -1
output
1

input
8 10 1 -2 3 -4 5 -6 7 -8
output
2