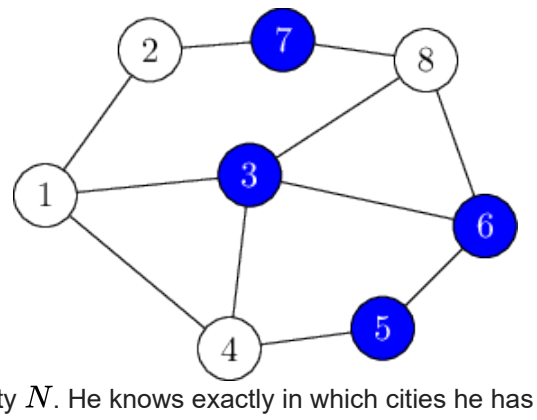
C. Traveling Debtor

time limit per test: 1 second
memory limit per test: 1024 megabytes

The traveling salesman Lex doesn't travel from city to city just because it's profitable to sell his exotic products in various different places. Most likely, due to the high availability of exotic items for sale on the Internet at very low prices, his business outcomes began to decline, and since then, Lex has been accumulating debts with loan sharks he meets in the cities.

With the help of an ancient technique taught by his ancestors, Lex has already solved the problem of visiting all the cities by making a route that goes through each of them only once. Now, Lex's problem is to return home while avoiding, as much as possible, the cities where he owes money.



Lex is currently in city 1 and lives in city N. He knows exactly in which cities he has debts, and now he needs to find out which path he should take to avoid these cities as much as possible, and obviously, he wants to move as little as possible. Lex tried, but couldn't solve this problem, so he asked for your help.

Input

The first line of the input contains three integers: the number of cities N $(2 \leq N \leq 10^5)$, the number of roads connecting the cities M $(1 \leq M \leq 10^5)$, and the number of cities where Lex owes money D $(1 \leq D \leq N)$.

The second line follows with D distinct integers, the cities C_i $(1 \leq C_i \leq N)$ where Lex owes to loan sharks.

Then, there are M distinct lines representing the roads connecting the cities, each one with the integers U and V $(1 \leq U < V \leq N)$, a road that connects the cities U and V in both directions.

It is guaranteed that there is a path between city 1 and city N.

Output

Print on a single line the minimum number of cities where Lex owes money that he will have to pass through on his way home, and the length of the shortest path to pass through this minimum number of cities.

Examples

input Copy

```
8 11 4
3 5 6 7
1 2
1 3
1 4
5 6
4 5
3 6
2 7
7 8
6 8
3 4
3 8
```

output Copy

```
1 2
```

input Copy

```
8 9 5
1 7 2 5 4
1 2
1 5
2 5
3 4
4 8
5 8
6 7
3 5
2 3
```

output Copy

```
2 2
```