

Nuclear Waste

time limit per test: 2 seconds (5 seconds for Python)
memory limit per test: 500 megabytes
input: standard input
output: standard output
points: 100 · problem code: spa2

You are traversing a nuclear wasteland, trying to make it back to your hideout. You are currently at location 1, and your hideout is at location n . Certain locations contain nuclear waste, and you want to stay as far from waste as possible.

Locations are connected by one-way paths. Each path has a length and may also provide food. If you traverse the same path again, you gain that same amount of food again.

Define the **safety** of a route as the minimum distance from any location on the route to the nearest waste site. Distance to waste is measured along path lengths, but for this distance computation every path is treated as **undirected** (you may measure through a path in either direction even though you can only travel it in one direction).

Among all routes from 1 to n , maximize safety. Among all routes that achieve that maximum safety, maximize the total food collected. If you can collect arbitrarily large amounts of food while keeping that maximum safety (by looping), the food answer is infinite.

Input

The first line contains the integers n , m , and w : the number of locations, number of paths, and number of waste locations respectively.

The second line contains w distinct integers, the locations that contain nuclear waste.

The following m lines each contain integers a , b , c , and d : a directed path from a to b with length c and food d .

Locations are numbered 1 through n .

It is guaranteed that there exists a path from location 1 to the hideout.

Output

Print two integers: the maximum safety, and the maximum food achievable at that safety. If the food is infinite, print `-1` instead of the food amount.

Constraints

- $2 \leq n \leq 200$
- $1 \leq m \leq 500$
- $1 \leq w \leq n$
- $1 \leq a, b \leq n$
- $1 \leq c \leq 10^9$
- $0 \leq d \leq 10^9$

Example 1

Input

```
3 3 1
2
1 2 3 10
2 3 4 10
1 3 1 1
```

Output

```
3 1
```

Explanation

Waste is only at location 2, so the waste distances are 3, 0, and 4 for locations 1, 2, and 3. The route $1 \rightarrow 2 \rightarrow 3$ has safety 0. The route $1 \rightarrow 3$ has safety $\min(3, 4) = 3$ and food 1, which is best.

Example 2

Input

```
10 20 2
5 7
1 2 10 0
2 3 10 0
3 4 7 0
4 5 4 0
5 6 10 0
6 7 3 0
7 8 8 0
8 9 10 0
9 10 9 9
10 1 8 0
5 9 2 0
8 7 1 2
7 7 2 0
1 1 1 0
9 10 1 6
8 5 1 0
9 4 3 0
10 6 3 2
2 1 1 0
3 10 3 7
```

Output

```
3 -1
```

Explanation

The maximum safety is 3. On routes that keep that safety there is a positive-food cycle that can still reach the hideout, so the food answer is infinite.

Example 3

Input

```
10 20 1
6
1 2 4 0
2 3 4 6
3 4 8 0
4 5 4 0
5 6 3 0
6 7 5 0
7 8 8 0
8 9 1 0
9 10 10 0
10 1 2 7
9 4 3 0
10 3 3 0
8 7 3 0
5 5 1 0
6 3 2 0
2 2 3 9
5 9 3 10
4 10 3 0
8 2 1 0
2 7 2 9
```

Output

```
5 -1
```

Explanation

There is a route to the hideout with safety 5 on which you can loop and collect unbounded food, so the second number is `-1`.