Problem	COLA	PARKING	CATTLE	MEADOW
Executable file	`cola.exe`	`parking.exe`	`cattle.exe`	`meadow.exe`

Source file	`cola.pas`
	`cola.c`
	`cola.cpp`
		`parking.pas`
		`parking.c`
		`parking.cpp`
			`cattle.pas`
			`cattle.c`
			`cattle.cpp`
				`meadow.pas`
				`meadow.c`
				`meadow.cpp`

Input file	`cola.in`	`parking.in`	`cattle.in`	`meadow.in`
Output file	`cola.out`	`parking.out`	`cattle.out`	`meadow.out`
Time limit per test	10 seconds	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10	10
Points per test	6	7	8	9
Total points	**60**	**70**	**80**	**90**
	300			

COLA

There are M bottles filled with extremely popular and tasteful beverage called **cola** in one room. Each bottle is marked by one of the numbers 1, 2, 3 ... M, and there is no pair of bottles marked by the same number.

A queue consisting of N thirsty people is formed at the entrance door to the room. Every person has been assigned one of numbers 1, 2, 3 ... N, and again, no pair of them has the same number. They are all waiting to drink cola from one of the bottles in the room. One by one, ordered by their numbers (person which was assigned number 1 is first to enter, the next one is the person which was assigned number 2 etc.), they enter the room, drink **one decilitre** of cola from **one** of the bottles in the room and then leave. No more than one person is allowed to be in the room at any moment. There are two kinds of people: **wasteful** and **economical**. Wasteful people always drink from (one of) the **most filled** bottle(s). Economical ones always drink from (one of) the **least filled** bottle(s) (empty bottles are ignored, of course).

It is known how much cola remained in each bottle after the last person left the room. You need to write a program that will determine which person drank from which bottle.

Input data

The first line of the input file contains three integers N, M and K, $1 \leq N \leq 1000$, $1 \leq M \leq 100$, $1 \leq K \leq 20$, number of people in the queue, number of bottles in the room and how many decilitres of cola were initially in each bottle, respectively. People are assigned numbers from 1 to N and bottles are marked by numbers 1 ... M as described above.

The second line of the input file contains a sequence of characters 'W' and 'E' of length N. Letter at position **i** describes $\mathbf{i}^{th}$ person: 'W' denotes wasteful person and 'E' denotes economic person.

The third (and the last line) of the input file contains M integers denoting the volumes of remaining cola in the corresponding bottles: the k^{th} number corresponds to the k^{th} bottle. Adjacent numbers are separated by one space character. Each of those M integers are from the set {0, 1, 2 ... K}.

Output data

The first and only line of the output file should contain N integers; **i**th number is the number of the bottle from which **i**th person drank cola. Two adjacent numbers should be separated by one space character.

Remark: There will always be a solution, not necessarily unique, to all test data.

Examples

cola.in

```
4 3 3
EEEE
3 0 2
```

cola.out

```
2 2 2 3
```

cola.in

```
4 3 3
EEWE
3 0 2
```

cola.out

```
2 2 3 2
```

cola.in

```
5 5 2
EEWWE
1 2 0 2 0
```

cola.out

```
3 3 5 1 5
```

PARKING

An underground garage consists of N parking places located in various floors. Each parking place is denoted by a number between 1 and N, and no two parking places are denoted with the same number. Each parking place is an **isolated** small room where just **one car** can be parked. Some of these rooms are directly connected via passages.

A car **cannot** pass through a parking place occupied by another car. Exit is located in the room #1 where no car can be parked (but any car can pass through it).

There is a **unique** path between any two parking places. Each pair of successive rooms on a path leading to the exit are located on successive floors of the garage, where the room nearer to the exit is always located in the **higher** floor.

Ralph Maher had parked his car in the room #P and now he wants to leave the garage. The problem is that there are some other cars parked in some of the rooms he needs to pass through. He has to **free** every occupied room by pushing a car parked in it to a room in a **lower** floor. One push is defined as moving a car **one** floor down from one parking place to another.

Write a program that will determine the **minimal** number of pushes Ralf needs to do to leave the garage, i.e. to reach the room #1. He will not move his car until path to the exit is completely cleared.

Input data

The first line of the input file contains three integers N, P and K ($2 \leq N \leq 5000$, $2 \leq P \leq N$, $0 \leq K \leq N-2$). Number K denotes the total number of cars parked in the garage excluding Ralph's car, which is parked in the parking place #P. Each of the following N lines contains data about parking places. The line (i+1) contains data about parking places directly connected to the parking place #i located one floor **beneath** it:

T A_1 A_2 ... A_T

There are T such parking places and A_1, A_2, ..., A_T are their numbers.

The last line of the input file contains K integers denoting occupied parking places, not counting Ralph's.

Output data

The first and only line of the output file should contain minimal number of pushes Ralph need to do to free the path to the exit. If the solution does not exist, the output file should contain one line with the text '**not solvable**' (without quotes).

Examples

parking.in

```
6 3 2
1 4
1 6
0
1 5
2 2 3
0
4 5
```

parking.out

```
4
```

parking.in

```
6 4 2
2 5 6
0
0
0
2 4 2
1 3
6 5
```

parking.out

```
1
```

parking.in

```
8 5 3
1 3
1 7
2 2 4
2 5 6
0
0
1 8
0
2 3 7
```

parking.out

```
2
```

CATTLE

A group of more or less domesticated bovine animals (hereafter to be referred to as cattle) are to be loaded into railway freight cars of certain train. The train consists of K freight cars (and a locomotive, which is kind of irrelevant to this problem), which can be loaded with at most M animals each. A number from 1 to N is painted on each animal (all numbers are used and no two animals carries the same number so by easy induction on N one concludes that there are exactly N animals).

The animals are patiently waiting to be loaded in a queue sorted by their numbers in ascending order (1 is the first to be loaded etc.). One freight car is loaded at a time by arbitrary number of animals (not exceeding M, of course) from the beginning of the queue and then locked so no subsequent load of animals to that freight car is possible. The process of loading is continued until all freight cars are locked.

It is known that some **clashing pairs** of animals dislikes each other and if loaded in the same freight car then the stronger one will attempt to kill the other. It is also known that there are **friendly pairs** of animals where the stronger animal will always successfully protect the weaker one from attack of any other animal. If no alive friendly protecting animal is present, the attack of stronger animal of a clashing pair is always successfull. All attacks in a freight car starts when it is locked. The transportation lasts long enough for all attacks to finish.

Write a program that will suggest the loading of animals to the freight cars so that the number of alive animals at the end of transportation will be as large as possible.

Input data

The first line of the input file contains three integers N, K, M; $1 \leq N,K \leq 1000$, $1 \leq M \leq 20$.

The second line contains integer D – the number of clashing pairs among the animals.

Each of the following D lines contains three mutually different integers A, B, C. The pair (A,B) represents a clashing pair, and the pair (C,B) represents a friendly pair of animals. The second animal in each pair is always the weaker one.

A stronger animal of a clashing pair cannot be a weaker animal in any other pair (the first number A cannot occur as a second number anywhere). A weaker animal of a clashing pair has exactly one friendly protecting animal (the third number C is uniquely determined by the first two numbers A and B).

Output data

The first and only line of the output file consists of an integer – the maximal number of alive animals at the end of transportation.

Examples

cattle.in

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

cattle.out

```
4
```

cattle.in

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

cattle.out

```
5
```

cattle.in

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

cattle.out

```
5
```

MEADOW

As you have probably guessed, we are talking about an ordinary meadow with lot of flowers scattered all around it. Flying between them, landing on them and taking off from them are numerous industrious bees, collecting raw material known as pollen used to produce honey in their hives. Since lives and prosperity of the bee community depends heavily on it, every flower needs to be visited to collect as much pollen as possible.

Maya is the bee who has to make a scheduling list according to which the bees will visit all the flowers on the meadow. A scheduling list consists of several sets of flowers (their positions, actually). Each bee is given one set and it has to visit all the flowers from it in an arbitrary order. Each flower can be visited arbitrary many times. The **weight of a sequence of flowers** is the **largest** distance between two successive flowers in the sequence. The **weight of a set of** flowers is the **minimal weight of all sequences of flowers from the set.** A bee with given set of flowers will always plan its flight according to the sequence of all flowers from the set with minimal weight. The **weight of a scheduling list** is the **maximal weight of all sets in the scheduling list**.

Write a program that will help Maya to find a scheduling list with the lowest weight.

Input data

The first lline of the input file consists of two natural numbers F and B; $1 \leq F \leq 2000$, $1 \leq B \leq F$, F is the number of flowers in the meadow and B is the number of bees available for collecting pollen.

Each of the next C lines contains two natural numbers X and Y, $1 \leq X,Y \leq 10000$, the coordinates of a flower.

Output data

The first and only line of the output file should contain the minimal possible weight of a scheduling list for the data given in the input file as described above. The result should be rounded to two decimal places. The result may differ from the correct solution by not more than 0.01.

Examples

meadow.in

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

meadow.out

```
1.41
```

meadow.in

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

meadow.out

```
3.00
```

meadow.in

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

meadow.out

```
3.00
```