
# UCL Algorithm Contest
## Round 3 - 2016

Do not open before the start of the contest.

This page is not really blank.


# Problem A
# Virus
# 2 pt

A new disease has started spreading and it is urgent to prevent it from being contracted by people. Fortunately a vaccine has just been invented but unfortunately it is very expensive to produce. When someone has received a vaccine that person cannot carry the disease anymore. If an unvaccinated person gets the disease and is in contact with another unvaccinated person, then both will have the disease.

We would like to know, given a group of $n$ persons and knowing which persons interact with each other on a daily basis, what is the minimum number of vaccines that are necessary to give to that group so that no matter who gets sick, that person will not transmit the disease to any other person in the group.

## Input

The first line of the input contains two integers $n$ and $m$ giving the number of people in the group and the number of pairs of people that interact with each other on a daily basis.

Then follow $m$ lines each with two integers $p$ and $q$ representing that persons $p$ and $q$ interact with each other on a daily basis.

**Constraints:**

- $1 \leq n \leq 16$
- $1 \leq m \leq \frac{1}{2}n(n-1)$
- $0 \leq p < q < n$

## Output

A single line with the minimum number of vaccines that are needed to be provided to that group so that the disease cannot be transmitted from a person in the group to another person.

# Example

**Input 1**

```
4 4
0 1
1 2
2 3
3 0
```

**Output 1**

```
2
```

**Input 2**

```
4 3
0 1
0 2
0 3
```

**Output 2**

```
1
```


# Problem B
# Taxi Driver
# 2pt

When you take a taxi you pay some fixed fare for every minute of the ride. Therefore the taxi driver wants the ride to take as long as possible. However if the taxi driver starts taking obviously long roads to reach the destination the client will get suspicious and file a complaint. In order to avoid that, the taxi wants to use his GPS to find the route. As the GPS uses the shortest path in terms of distance, the client will not get suspicious. However there might exist several shortest paths and because of traffic conditions some might take longer to travel than others.

The taxi company contacted you to help them program the GPS so that, given the length of each road and the estimated time to traverse each road, it finds the shortest path in terms of length that takes the longest time.

## Input

The first line of the input contains two integers $n$ and $m$ separated by a single space representing the number of intersections in the city and the number of roads respectively.

Then follow $m$ lines each with four integers $x$, $y$, $l$ and $t$ meaning that there is a bidirectional road from intersection $x$ to intersection $y$ of length $l$ that takes time $t$ to traverse.

Finally follows a line with two integers $o$ ant $d$ giving the ID of the origin intersection and the destination intersection, respectively.

You may assume that there exists at least one path from $o$ to $d$.

**Constraints:**

- $1 \leq n \leq 10^4$
- $1 \leq m \leq \min(\frac{1}{2}n(n-1), 10^5)$
- $x < y$
- $1 \leq l, t \leq 10^5$
- $0 \leq o, d < n$

## Output

A single line with the maximum time it takes to go from $o$ to $d$ by following a shortest path.

## Example

**Input 1**

```
4 4
0 1 1 2
1 3 1 3
3 2 1 2
0 2 1 1
0 3
```

**Output 1**

```
5
```

**Input 2**

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

**Output 2**

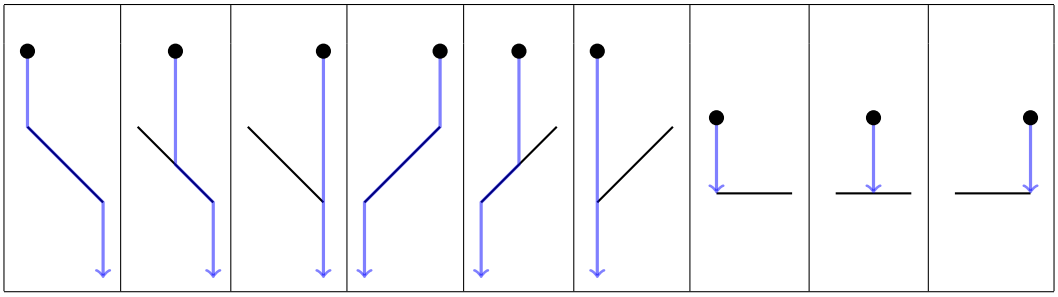
```
1
```

UCL ALGORITHM CONTEST

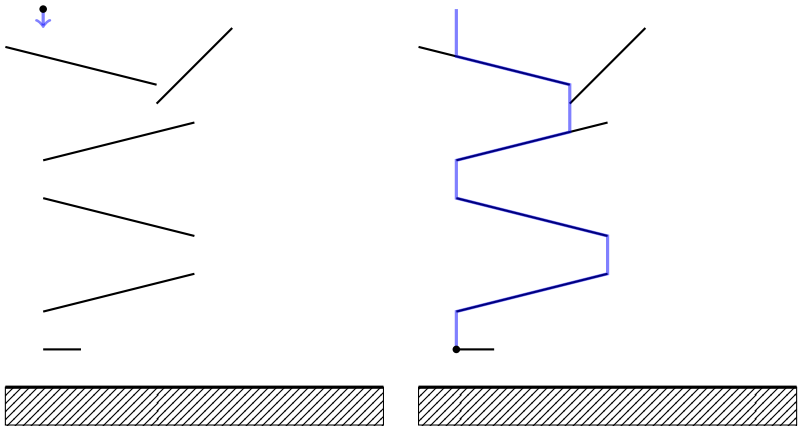
# ● Problem C
# Free Fall
# 1pt + 1pt

In the living room at your parents house, there are several wood shelves attached to a wall. They are there more for artistic reasons than utility since most of them are crooked. You and your best fiend like to play a game that consists of dropping a ball from the top and seeing it fall through the shelves. Since you are not 5 years old anymore, you quickly get bored of that game. You then start to wonder about figuring out the final position of the ball given its initial position.

For simplicty we assume that the shelves are represented by line segments and we assume that the ball does not bounce and falls vertically. Furthermore, we assume that the ball has no size, that no two shelves intersect or share an endpoint and that no shelve is vertical. The following figure shows the 9 possible cases for the behaviour of the ball.



The following picture shows a complete example of the trajectory of the ball when falling:



Since you are a great programmer, you decided to simply write an algorithm that does the computation for you. The floor of the room is represented by the line $y = 0$ which behaves as a horizontal segment. No segments touch or intersect the floor.

# Input

The first line of the input contains a single integer $n$ giving the number of shelves.

Then follow $n$ lines each with four integers $x_1, y_1, x_2, y_2$ separated by singles spaces giving the coordinates of the two endpoints of each of the shelves.

The last line contains two integers $x$ and $y$ giving the position from where you drop the ball.

**Easy constraints (C easy):**

- $1 \leq n \leq 20$
- $0 \leq x_1, x_2 \leq 10^5$
- $1 \leq y_1, y_2 \leq 10^5$
- $x_1 \neq x_2$
- no two shelves intersect nor touch the floor

**Hard constraints (C hard):**

- $1 \leq n \leq 10^5$
- $0 \leq x_1, x_2 \leq 10^5$
- $1 \leq y_1, y_2 \leq 10^5$
- $x_1 \neq x_2$
- no two shelves intersect nor touch the floor

# Output

A single line with two integers $x$ and $y$ separated by a single space giving the final position of the ball.

# Example

**Input 1**

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

**Output 1**

```
1 0
```

**Input 2**

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

**Output 2**

```
2 0
```

This page is not really blank.


# Problem D
# My cat messed up my work
# 2pt

You just finished writing your best paper ever. You feel happy but tired and you fall asleep with your Latex editor opened. If you know anything about cats, you know that they always like to sleep on strange places. One all time favorite is the computer keyboard. When you wake up you see your cat sleeping on top of your keyboard and your Latex document is a mess. Strangely enough, it looks like your cat only deleted all the spaces in your document. It also seems like he pressed some sort of keyboard shortcut that create a text file with a list of all the words in your paper.

Maybe your cat also deleted some other characters. You want to know if what remains of the text is some concatenation of the words in the list. Note that any word can be used multiple times.

## Input

The first line of the input contains a single integer $n$ giving the number of words in the list.

Then follow $n$ lines, each with one of the words $x_1, \ldots, x_n$ in the list.

Finally, the last line contains a string $w$ representing what remains of your paper.

All input strings are defined over the alphabet

$$A = \{a, \ldots, z, A, \ldots, Z, 0 \ldots, 9\}$$

**Constraints:**

- $1 \le n \le 10^6$
- $|w| \le 1000$
- $1 \le |x_i| \le |w|$
- $\sum_{i=1}^{n} |x_i| \le 4 \cdot 10^6$

# Output

A single line with either yes if $w$ can be obtained from a concatenation of words from the list or no otherwise.

# Example

**Input 1**

```
6
a
b
c
d
ab
cd
abcd
```

**Output 1**

```
yes
```

**Input 2**

```
4
aaabbb
aa
aabbb
a
aaabbb
```

**Output 2**

```
yes
```

**Input 3**

```
2
ab
bc
abc
```

**Output 3**

```
no
```


# Problem E
# Even or Odd?
# 2pt

Eve and Oden love playing games with numbers. Their latest favorite works as follows. First they agree on a subset $S$ with $n$ integers. Then Eve has to find a subset of $S$ such that the sums of all of its elements is even. Then Oden has to find a subset of $S$ that has an odd sum. They cannot choose the same subset twice and the first of them that cannot find a new subset loses the game.

Given the initial subset $S$, can you figure out who will win assuming that they never fail to find any subset when playing the game?

## Input

The first line of the input contains a single integer $n$ giving the number of elements in $S$. Then follow $n$ lines each with one integer $x_i$ giving the elements of $S$. All those elements are distinct.

**Constraints**

- $1 \leq n \leq 10000$
- $0 \leq x_i \leq 2^{31} - 1$

## Output

A single line with Eve if Eve will win and Oden otherwise.

## Example

| Input 1 | Output 1 |
| --- | --- |
| 3<br>1<br>2<br>3 | Oden |

**Input 2**

```
1
0
```

**Output 2**

```
Eve
```

This page is not really blank.