8197 Jumping Frog

Pog the Frog wants to compete in the World Frog Jump competition, which will take place in Nlogonia. In the competition, each frog must perform a sequence of acrobatic jumps in a specially built arena. The arena is composed of N equally spaced positions around a circumference (the arc between two adjacent positions is always the same length) where each position can be either a rock or a pond. The positions are numbered sequentially from 0 to $N-1$ in the clockwise direction, so that judges can easily make notes about which jumps were performed in each position. Thus, position 0 is adjacent to positions 1 and $N-1$ in the arena.

The competition rules stipulate that the sequence of jumps of each frog must start at a rock, always go from a rock to another rock, and finish at the same position it started. The rules do not require frogs to use every rock in the arena for their sequence of jumps.

Pog the Frog is currently practicing for the competition. He must develop two skills. First, he needs to get better at jumping from one rock to another, since landing on either a pond or outside of the marked positions can mean disqualification. Besides that, he must learn impressing acrobatic moves. With that in mind, he has decided on a practicing strategy. In the beginning of each practice session, Pog the Frog will pick a starting rock and an integer *jump length* K between 1 and $N-1$. After that, whenever he is standing on a rock numbered i, he will aim his next acrobatic jump at the rock whose number is obtained by getting the remainder of the division of $i+K$ by N. He will stop when he lands on the starting rock. For example, if the arena has 3 positions, all of them rocks, and Pog the Frog starts at position 0 and picks $K = 2$, he will first jump from rock 0 to rock 2, then to rock 1, and finally jump back to rock 0. At this point, his practice session ends.

Given the description of the N positions in the arena, help Pog the Frog by answering this question: how many distinct values of K can he choose for his practice sessions, if he can use any rock as a starting position for his sequence of jumps?

Input

The input file contains several test cases, each of them consists of a single line that contains a string S of N characters ($3 \leq N \leq 10^5$), representing the positions in the arena. The i-th character of S ($i = 0, 1, \ldots, N-1$) indicates that the position i in the arena is either a rock (uppercase letter '`R`') or a pond (uppercase letter '`P`').

Output

For each test case, output a single line with an integer representing the number of distinct jump lengths that Pog the Frog can choose for his practice sessions, given that he can use any rock as a starting position for his sequence of jumps.

Sample Input

```
RRR
RRPR
PRP
```

Sample Output

```
2
1
0
```

8189 Buggy ICPC

Alan Curing is a famous sports programmer. He is the creator of the theoretical model of computation known as the Alan Curing Machine (ACM). He's most famous for creating his own computer for programming competitions: the Integrated Computer for Programming Contests (ICPC). This computer has a specialized operating system with commands for submitting code and testing executables on sample inputs, an input generator, a wide display for debugging, and a very soft keyboard. However, as it happens even to the best, Alan's creation has a nasty bug. Every time Alan types a vowel on the ICPC, the content of the current line is reversed.

The bug has been extremely hard to track down, so Alan has decided to accept the challenge and use the computer as it is. He is currently training touch typing on the ICPC. For now, he is only typing strings using lowercase letters, and no spaces. When Alan types a consonant, it is appended to the end of the current line, as one would expect. When he types a vowel, however, the typed character is first added to the end of the line, but right after that the whole line is reversed. For example, if the current line has "`imc`" and Alan types "`a`" (a vowel), for a brief moment the line will become "`imca`", but then the bug kicks in and turns the line into "`acmi`". If after that he types the consonants "`c`", "`p`" and "`c`", in that order, the line becomes "`acmicpc`".

When practicing, Alan first thinks of the text he wants to type, and then tries to come up with a sequence of characters he can type in order to obtain that text. He is having trouble, however, since he realized that he cannot obtain some texts at all (such as "`ca`"), and there are multiple ways of obtaining other texts (as "`ac`", which is obtained whether he types "`ac`" or "`ca`"). Help Alan in his training by telling him in how many ways he can type each text he wishes to type. A way of typing a text T can be encoded by a string W with $|T|$ characters such that if the characters are typed on the ICPC in the order they appear in W (i.e. $W_1, W_2, \ldots, W_{|T|}$) the final result is equal to T, considering ICPC's known bug. Two ways are considered different if they are encoded by different strings. The letters that trigger the bug in the ICPC when typed are "`a`", "`e`", "`i`", "`o`" and "`u`".

Input

The input file contains several test cases, each of them consists of a single line that contains a non-empty string T of at most 10^5 lowercase letters, representing the text Alan wants to type on the ICPC.

Output

For each test case, output a single line with an integer representing the number of distinct ways Alan can type the desired text T considering ICPC's known bug.

Sample Input

```
ac
ca
acmicpc
```

Sample Output

```
2
0
3
```

6187 Never Wait for Weights

In a laboratory, an assistant, Nathan Wada, is measuring weight differences between sample pieces pair by pair. He is using a balance because it can more precisely measure the weight difference between two samples than a spring scale when the samples have nearly the same weight.

He is occasionally asked the weight differences between pairs of samples. He can or cannot answer based on measurement results already obtained.

Since he is accumulating a massive amount of measurement data, it is now not easy for him to promptly tell the weight differences. Nathan asks you to develop a program that records measurement results and automatically tells the weight differences.

Input

The input consists of multiple datasets. The first line of a dataset contains two integers N and M. N denotes the number of sample pieces ($2 \leq N \leq 100,000$). Each sample is assigned a unique number from 1 to N as an identifier. The rest of the dataset consists of M lines ($1 \leq M \leq 100,000$), each of which corresponds to either a measurement result or an inquiry. They are given in chronological order.

A measurement result has the format,

`!` a b w

which represents the sample piece numbered b is heavier than one numbered a by w micrograms ($a \neq b$). That is, $w = w_b - w_a$, where w_a and w_b are the weights of a and b, respectively. Here, w is a non-negative integer not exceeding 1,000,000.

You may assume that all measurements are exact and consistent.

An inquiry has the format,

`?` a b

which asks the weight difference between the sample pieces numbered a and b ($a \neq b$).

The last dataset is followed by a line consisting of two zeros separated by a space.

Output

For each inquiry, `?` a b, print the weight difference in micrograms between the sample pieces numbered a and b, $w_b - w_a$, followed by a newline if the weight difference can be computed based on the measurement results prior to the inquiry. The difference can be zero, or negative as well as positive. You can assume that its absolute value is at most 1,000,000. If the difference cannot be computed based on the measurement results prior to the inquiry, print '`UNKNOWN`' followed by a newline.

Sample Input

```
2 2
! 1 2 1
? 1 2
2 2
! 1 2 1
? 2 1
4 7
! 1 2 100
```

```
? 2 3
! 2 3 100
? 2 3
? 1 3
! 4 3 150
? 4 1
0 0
```

Sample Output

```
1
-1
UNKNOWN
100
200
-50
```

6189 Company Organization

You started a company a few years ago and fortunately it has been highly successful. As the growth of the company, you noticed that you need to manage employees in a more organized way, and decided to form several groups and assign employees to them.

Now, you are planning to form n groups, each of which corresponds to a project in the company. Sometimes you have constraints on members in groups. For example, a group must be a subset of another group because the former group will consist of senior members of the latter group, the members in two groups must be the same because current activities of the two projects are closely related, the members in two groups must not be exactly the same to avoid corruption, two groups cannot have a common employee because of a security reason, and two groups must have a common employee to facilitate collaboration.

In summary, by letting X_i $(i = 1, \ldots, n)$ be the set of employees assigned to the i-th group, we have five types of constraints as follows.

1. $X_i \subseteq X_j$
2. $X_i = X_j$
3. $X_i \neq X_j$
4. $X_i \cap X_j = \emptyset$
5. $X_i \cap X_j \neq \emptyset$

Since you have listed up constraints without considering consistency, it might be the case that you cannot satisfy all the constraints. Constraints are thus ordered according to their priorities, and you now want to know how many constraints of the highest priority can be satisfied.

You do not have to take ability of employees into consideration. That is, you can assign anyone to any group. Also, you can form groups with no employee. Furthermore, you can hire or fire as many employees as you want if you can satisfy more constraints by doing so.

For example, suppose that we have the following five constraints on three groups in the order of their priorities, corresponding to the first dataset in the sample input.

- $X_2 \subseteq X_1$
- $X_3 \subseteq X_2$
- $X_1 \subseteq X_3$
- $X_1 \neq X_3$
- $X_3 \subseteq X_1$

By assigning the same set of employees to X_1, X_2, and X_3, we can satisfy the first three constraints. However, no matter how we assign employees to X_1, X_2, and X_3, we cannot satisfy the first four highest priority constraints at the same time. Though we can satisfy the first three constraints and the fifth constraint at the same time, the answer should be three.

Input

The input consists of several datasets. The first line of a dataset consists of two integers n ($2 \leq n \leq 100$) and m ($1 \leq m \leq 10000$), which indicate the number of groups and the number of constraints, respectively. Then, description of m constraints follows. The description of each constraint consists of three integers s ($1 \leq s \leq 5$), i ($1 \leq i \leq n$), and j ($1 \leq j \leq n, j \neq i$), meaning a constraint of the s-th type imposed on the i-th group and the j-th group. The type number of a constraint is as listed above. The constraints are given in the descending order of priority.

The input ends with a line containing two zeros.

Output

For each dataset, output the number of constraints of the highest priority satisfiable at the same time.

Sample Input

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

Sample Output

```
3
4
4
2
```

CHAIN - Strange Food Chain

no tags

There are 3 kinds of animals A,B and C. A can eat B,B can eat C,C can eat A. It's interesting,isn't it?

Now we have n animals,numbered from 1 to n. Each of them is one of the 3 kinds of animals:A,B,C.

Today Mary tells us k pieces of information about these n animals. Each piece has one of the two forms below:

- 1 x y: It tells us the kind of x and y are the same.
- 2 x y: It tells us x can eat y.

Some of these k pieces are true,some are false. The piece is false if it satisfies one of the 3 conditions below, otherwise it's true.

- X or Y in this piece is larger than n.
- This piece tells us X can eat X.
- This piece conflicts to some true piece before.

Input

The first line contains a single integer t.t blocks follow.

To every block,the first line contains two integers n(1<=n<=50000) and k (1<=k<=100000). k lines follow,each contains 3 positive integers D(1<=D<=2),X,Y,separated by single spaces.

Output

t lines,each contains a single integer - the number of false pieces in the corresponding block.

Example

Sample input:

```
1
100 7
1 101 1
2 1 2
2 2 3
2 3 3
1 1 3
2 3 1
1 5 5
```

Sample output:

```
3
```

Hint:

```
The false pieces are the 1st,the 4th and the 5th ones.
```

Warning: large Input/Output data, be careful with certain languages

8165 Justified Jungle

As you probably know, a *tree* is a graph consisting of n nodes and $n-1$ undirected edges in which any two nodes are connected by exactly one path. A *forest* is a graph consisting of one or more trees. In other words, a graph is a forest if every connected component is a tree. A forest is *justified* if all connected components have the same number of nodes.

Given a tree G consisting of n nodes, find all positive integers k such that a justified forest can be obtained by erasing exactly k edges from G. Note that erasing an edge never erases any nodes. In particular when we erase all $n-1$ edges from G, we obtain a justified forest consisting of n one-node components.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer n ($2 \leq n \leq 1000000$) — the number of nodes in G. The k-th of the following $n-1$ lines contains two different integers a_k and b_k ($1 \leq a_k, b_k \leq n$) — the endpoints of the k-th edge.

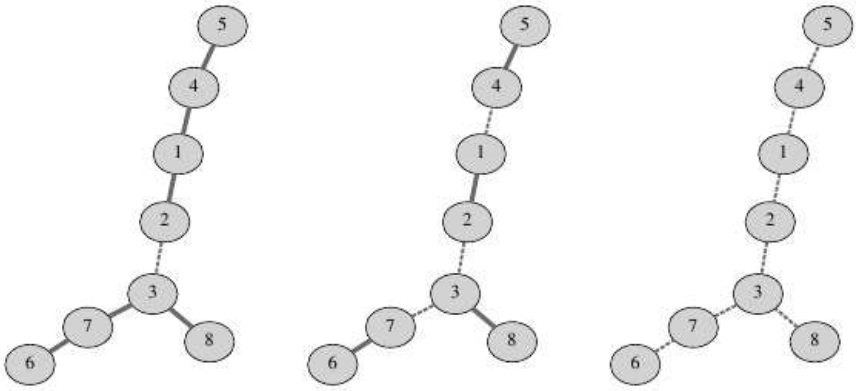
Output

For each test case, on a line by itself, should contain all wanted integers k, in increasing order.

Note: Figures depict justified forests obtained by erasing 1, 3 and 7 edges from the tree in the example input.

Sample Input

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



Sample Output

```
1 3 7
```

7209 Galactic taxes

The year is 2115. The Interplanetary Commercial Planning Center (ICPC) is supported by the Autonomous Communication Ministry (ACM).

A commercial operation is performed executing transactions between connected ACM offices throughout the galaxy. The execution of a transaction between two connected ACM offices involves a non-negative tax whose value increases, or decreases, continuously as a linear function $A \times t + B$ of time t, where t is a real number measured in minutes during the day ($0 \leq t \leq 24 \times 60$).

The total tax of a commercial operation performed between a source ACM office and a destination ACM office at some time t, is calculated as the minimum possible sum of the taxes of the executed transactions between the ACM offices visited along some path from the source ACM office to the destination ACM office. The tax of each transaction is calculated at the same time t.

Since the tax of the transactions between connected ACM offices is continually changing during the day, it would be better to perform the commercial operation at some specific time in the day, in order to maximize the collected tax. At that time, ACM decides to perform the commercial operation, and not before or after.

Your task is to write a program that receives as input the description of the ACM office network and returns as output the maximum total tax of the commercial operation that can be achieved during the day, that is, the maximum total tax that ACM can collect.

Input

The input contains several test cases; each test case is formatted as follows. The first line contains two integers N and M, representing respectively the number of ACM offices in the network, and the number of connections ($2 \leq N \leq 1000$ and $1 \leq M \leq 10^4$). The ACM offices are identified with distinct integers from 1 to N, being 1 the source ACM office and N the destination ACM office. Each of the next M lines describes a connection with four integers I, J, A and B, indicating that there is a bidirectional connection between office I and office J ($1 \leq I < J \leq N$), such that the tax of a transaction executed between office I and office J at time t is defined by the formula $A \times t + B$ ($-100 \leq A \leq 100$ and $0 \leq B \leq 10^6$). Taxes are non-negative, so $A \times t + B \geq 0$ for $0 \leq t \leq 24 \times 60$. There is at most one connection between each pair of ACM offices, and there is at least one path between the source ACM office and the destination ACM office.

Output

For each test case in the input, output a line with a rational number representing the maximum total tax that ACM can collect. The result must be output as a rational number with exactly five digits after the decimal point, rounded if necessary.

Sample Input

```
2 1
1 2 1 0
5 8
1 2 27 610658
2 3 -48 529553
3 4 -6 174696
4 5 47 158238
```

```
3 5 84 460166
1 3 -21 74502
2 4 -13 858673
1 5 -90 473410
3 3
1 2 1 0
2 3 1 0
1 3 -1 1440
4 5
1 2 1 0
2 4 2 0
1 4 0 500
1 3 -1 1440
3 4 -2 2880
2 1
1 2 0 0
```

Sample Output

```
1440.00000
419431.27273
960.00000
500.00000
0.00000
```

C. Weakness and Poorness

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

You are given a sequence of n integers $a_1, a_2, \ldots, a_n$.

Determine a real number x such that the *weakness* of the sequence $a_1 - x, a_2 - x, \ldots, a_n - x$ is as small as possible.

The *weakness* of a sequence is defined as the maximum value of the *poorness* over all segments (contiguous subsequences) of a sequence.

The *poorness* of a segment is defined as the absolute value of sum of the elements of segment.

Input

The first line contains one integer n ($1 \le n \le 200\,000$), the length of a sequence.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($|a_i| \le 10\,000$).

Output

Output a real number denoting the minimum possible *weakness* of $a_1 - x, a_2 - x, \ldots, a_n - x$. Your answer will be considered correct if its relative or absolute error doesn't exceed 10^{-6}.

Examples

input Copy

```
3
1 2 3
```

output Copy

```
1.000000000000000
```

input Copy

```
4
1 2 3 4
```

output Copy

```
2.000000000000000
```

input Copy

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

output Copy

```
4.500000000000000
```

Note

For the first case, the optimal value of x is 2 so the sequence becomes $-1, 0, 1$ and the max poorness occurs at the segment "-1" or segment "1". The poorness value (answer) equals to 1 in this case.

For the second sample the optimal value of x is 2.5 so the sequence becomes $-1.5, -0.5, 0.5, 1.5$ and the max poorness occurs on segment "-1.5 -0.5" or "0.5 1.5". The poorness value (answer) equals to 2 in this case.

D. Devu and his Brother

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Devu and his brother love each other a lot. As they are super geeks, they only like to play with arrays. They are given two arrays a and b by their father. The array a is given to Devu and b to his brother.

As Devu is really a naughty kid, he wants the minimum value of his array a should be at least as much as the maximum value of his brother's array b.

Now you have to help Devu in achieving this condition. You can perform multiple operations on the arrays. In a single operation, you are allowed to decrease or increase any element of any of the arrays by 1. Note that you are allowed to apply the operation on any index of the array multiple times.

You need to find minimum number of operations required to satisfy Devu's condition so that the brothers can play peacefully without fighting.

Input

The first line contains two space-separated integers n, m $(1 \le n, m \le 10^5)$. The second line will contain n space-separated integers representing content of the array a $(1 \le a_i \le 10^9)$. The third line will contain m space-separated integers representing content of the array b $(1 \le b_i \le 10^9)$.

Output

You need to output a single integer representing the minimum number of operations needed to satisfy Devu's condition.

Examples

input Copy

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

output Copy

```
3
```

input Copy

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

output Copy

```
4
```

input Copy

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

output Copy

```
0
```

Note

In example 1, you can increase a_1 by 1 and decrease b_2 by 1 and then again decrease b_2 by 1. Now array a will be $[3; 3]$ and array b will also be $[3; 3]$. Here minimum element of a is at least as large as maximum element of b. So minimum number of operations needed to satisfy Devu's condition are 3.

In example 3, you don't need to do any operation, Devu's condition is already satisfied.

D. Deduction Queries

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

There is an array a of 2^{30} integers, indexed from 0 to $2^{30}-1$. Initially, you know that $0 \le a_i < 2^{30}$ ($0 \le i < 2^{30}$), but you do not know any of the values. Your task is to process queries of two types:

- `1 l r x`: You are informed that the **bitwise xor** of the subarray $[l, r]$ (ends inclusive) is equal to x. That is, $a_l \oplus a_{l+1} \oplus \ldots \oplus a_{r-1} \oplus a_r = x$, where $\oplus$ is the bitwise xor operator. In some cases, the received update contradicts past updates. In this case, you should **ignore** the contradicting update (the current update).
- `2 l r`: You are asked to output the bitwise xor of the subarray $[l, r]$ (ends inclusive). If it is still impossible to know this value, considering all past updates, then output -1.

Note that the queries are **encoded**. That is, you need to write an **online** solution.

Input

The first line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Each of the next q lines describes a query. It contains one integer t ($1 \le t \le 2$) — the type of query.

The given queries will be **encoded** in the following way: let $last$ be the answer to the last query of the second type that you have answered (initially, $last = 0$). If the last answer was -1, set $last = 1$.

- If $t = 1$, three integers follow, l', r', and x' ($0 \le l', r', x' < 2^{30}$), meaning that you got an update. **First, do the following**:
$$l = l' \oplus last, r = r' \oplus last, x = x' \oplus last$$
and, if $l > r$, swap l and r.

  This means you got an update that the bitwise xor of the subarray $[l, r]$ is equal to x (notice that you need to ignore updates that contradict previous updates).

- If $t = 2$, two integers follow, l' and r' ($0 \le l', r' < 2^{30}$), meaning that you got a query. **First, do the following**:
$$l = l' \oplus last, r = r' \oplus last$$
and, if $l > r$, swap l and r.

  For the given query, you need to print the bitwise xor of the subarray $[l, r]$. If it is impossible to know, print -1. **Don't forget to change the value of $last$.**

It is guaranteed there will be at least one query of the second type.

Output

After every query of the second type, output the bitwise xor of the given subarray or -1 if it is still impossible to know.

Examples

input Copy

```
12
2 1 2
2 1 1073741822
1 0 3 4
2 0 0
2 3 3
2 0 3
1 6 7 3
2 4 4
1 0 2 1
2 0 0
2 4 4
2 0 0
```

output Copy

```
-1
-1
-1
-1
5
-1
6
3
5
```

input Copy

```
4
1 5 5 9
1 6 6 5
1 6 5 10
2 6 5
```

output Copy

```
12
```

Note

In the first example, the real queries (without being encoded) are:

- `12`
- `2 1 2`
- `2 0 1073741823`
- `1 1 2 5`
- `2 1 1`
- `2 2 2`
- `2 1 2`
- `1 2 3 6`
- `2 1 1`
- `1 1 3 0`
- `2 1 1`
- `2 2 2`
- `2 3 3`

- The answers for the first two queries are -1 because we don't have any such information on the array initially.
- The first update tells us $a_1 \oplus a_2 = 5$. Note that we still can't be certain about the values a_1 or a_2 independently (for example, it could be that $a_1 = 1, a_2 = 4$, and also $a_1 = 3, a_2 = 6$).
- After we receive all three updates, we have enough information to deduce a_1, a_2, a_3 independently.

In the second example, notice that after the first two updates we already know that $a_5 \oplus a_6 = 12$, so the third update is contradicting, and we ignore it.