A. Connect and Disconnect, 3 seconds, 256 megabytes, connect.in, connect.out

Do you know anything about DFS, Depth First Search? For example, using this method, you can determine whether a graph is connected or not in $O(E)$ time. You can even count the number of connected components in the same time.

Do you know anything about DSU, Disjoint Set Union? Using this data structure, you can process queries like "Add an edge to the graph" and "Count the number of connected components in the graph" fast.

And do you know how to solve Dynamic Connectivity Problem? In this problem, you have to process three types of queries fast:

1. Add an edge to the graph
2. Delete an edge from the graph
3. Count the number of connected components in the graph

Input

At the first moment, the graph is empty.

The first line of file contains two integers N and K—number of vertices and number of queries ($1 \le N \le 300\,000$, $0 \le K \le 300\,000$). Next K lines contain queries, one per line. There are three types of queries:

1. + u v: add an edge between vertices u and v. It is guaranteed that there is no such edge in the graph at the time of the query.
2. - u v: remove an edge between vertices u and v. It is guaranteed that this edge is present in the graph at the time of the query.
3. ?: count the number of connectivity components in the graph at the time of the query.

Vertices are numbered 1 through N. No query will have $u = v$. The graph is undirected.

Output

For each '?' query, output the number of connectivity components in the graph at the time of the query on a single line.

input

```
5 11
?
+ 1 2
+ 2 3
+ 3 4
+ 4 5
+ 5 1
?
- 2 3
?
- 4 5
?
```

output

```
5
1
1
2
```

B. GraphAero, 2 seconds, 256 megabytes, bridges.in, bridges.out

Given an undirected graph with N vertices's and M edges. You are to process K queries of adding new edges into the graph. After each query you should output the only number - amount of bridges in graph. Graph may contain loops and multiple edges.

Input

The first line of input contains two integers N and M: the number of vertices and edges, respectively. $1 \le N \le 10^5$, $1 \le M \le 10^5$. The following M lines contain edges, one per line. Two integers a_i and b_i follow, describing the edge. $1 \le a_i, b_i \le N$.

The next line of input contains only integer K: the number of requests. $1 \le K \le 10^5$. The following K lines contain requests, one per line. Two integers a_i and b_i follow, describing the edge to add. $1 \le a_i, b_i \le N$.

Output

Write K integers: the number of bridges in the graph after each request.

input

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

output

```
1
2
3
0
```

C. Bridges in a Tree, 5 seconds, 256 megabytes, bridges2.in, bridges2.out

December is here, it's time for... Exams!

Santa Claus

Boy Serezha has almost graduated. There are two huge tasks left: first, to pass the winter exams, and second, to write the research work. The exams are not a problem: Serezha has already tried to pass them a year ago. But the research work is much more complicated.

Serezha selected the research topic two years ago: it is the "Dynamic 2-Connectivity Problem". This is a problem about dynamically changing a graph and responding to queries like "the number of bridges in the graph at the given moment".

Serezha has invented several solutions. One of them, the simpliest one for coding, has an unpleasant subtask. Serezha wrote the code, but it seemed to be overcomplicated.

So, he wondered, if there's another simplier implementation. A shorter, but as fast as his one. To find out the answer for this question, he decided to offer this task to a University Championship.

There are only a few months left before the time to present the research work, and Serezha is going to code much more funny algorithms, and the time is running out. And if he does not manage to finish his work in time, he will have to pass the whole fifth year of study. Again.

Please help Serezha to face this unpleasant subtask.

Recall that a tree is an undirected connected graph without cycles. A bridge is an edge of an undirected graph that increases the number of its connected components when removed.

Given is a tree YourTree of N ($2 \le N \le 100\,000$) vertices. Your task is to calculate the number of bridges in the graphs which are produced by adding sets of K_i edges to the tree YourTree.

Input

The first line contains integers N and M ($2 \le N \le 100\,000$, $1 \le M \le 100\,000$): the number of vertices and the number of requests.

The second line contains the description of the tree YourTree: there are $N - 1$ integers $p_2, p_3, \ldots, p_N$ separated by single spaces which represent edges $(2, p_2), (3, p_3), \ldots, (N, p_N)$. It is guaranteed that these edges form a tree. You may assume that $1 \le p_i < i$.

M requests follow, one per line. The request number i is described by a non-negative integer K_i and K_i pairs of integers 1 through N. Each pair describes an edge to be added to the tree YourTree.

The sum K_i in all requests does not exceed $100\,000$.

Note that the graphs may contain loops and mul that a multiple edge can never be a bridge.

online now: 0

online now:

Output

For each request, write a single integer: the number of bridges in the graph produced by adding the given edges to the tree YourTree.

input

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

output

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

D. Bridges: The Final Battle,

2 seconds, 256 megabytes, **bridges3.in, bridges3.out**

Does your research work have any practical applications?

Frequently asked question

Serezha has almost finished his thesis. The subject hasn't changed since December: "Dynamic 2-Edge-Connectivity Problem". The algorithm has been invented and tested, the bound of $O(K \log K)$ has been proved. The only thing left is to write about "practical applications".

Well, does the problem "Dynamic 2-Edge-Connectivity Problem" indeed have any practical applications? That's not a simple question. Probably, it's even harder than the problem itself. Whatever, the thesis has to be done.

So, the first practical application: let us create a contest problem about it!

Given an undirected graph with no more than 10^5 vertices. Initially it does not contain any edges. You have to process requests ADD x y and DEL x y — to add and to remove edge from x to y, respectively.

After each request, you should find **the number of bridges** in the graph.

There are no multiedges and loops.

For every request to remove an edge, the corresponding edge exists.

The thesis has to be pretty hard to be written in five hours. So you are given five hints.

1. Requests to add or remove edge make the edge "alive" during some intervals of time.
2. Use "Divide and conquer" idea.
3. Compress components of biconnectivity.
4. Even having compressed biconnectivity components, the graph can be reduced provided there are few requests.
5. The solution in $O(K \log K)$ exists.

Input

The first line of input contains two integers N and K: the number of vertices and requests, respectively. $1 \le N \le 10^5$, $1 \le K \le 10^5$.

The following K lines contain requests, one per line. Each request starts with a word "ADD" or "DEL", depending on the type of the request. Two integers a_i and b_i follow, describing the edge to add or to remove. $1 \le a_i, b_i \le N, a_i \ne b_i$.

Output

Write K integers: the number of bridges in the graph after each request.

input

```
4 8
ADD 1 2
ADD 2 3
ADD 1 3
DEL 2 3
DEL 1 2
ADD 2 4
ADD 1 4
ADD 2 3
```

output

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

E. Disconnected Graph,

3 seconds, 256 megabytes, **disconnected.in, disconnected.out**

You are given a connected undirected graph and several small sets of its edges. For each set, you need to determine whether the graph stays connected with edges from the set removed.

Remember that a graph is connected when for every two distinct vertices there's a path connecting them.

Input

The first line of the input file contains two integers n and m ($1 \le n \le 10\,000$, $1 \le m \le 100\,000$), denoting the number of vertices and edges in the graph, respectively. Vertices are numbered from 1 to n.

The next m lines contain the description of the edges. Each line contains two integers a and b — the numbers of the vertices connected by this edge. Each pair of vertices is connected by at most one edge. No edge connects a vertex to itself. Edges are numbered from 1 to m in the order they are given in the input.

The next line contains an integer k ($1 \le k \le 100\,000$), denoting the number of small sets to test. The next k lines contain the descriptions of the small sets. Each line starts with an integer c ($1 \le c \le 4$), denoting the number of edges in the set, followed by c numbers of the edges from the set. The numbers of the edges inside one small set will be distinct.

Output

Output k lines, one per each given small set. The i-th line should contain "Connected" (without quotes), if removal of the corresponding small set leaves the graph connected, or "Disconnected" otherwise.

input

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

output

```
Connected
Disconnected
Connected
```