G2. Baudelaire (hard version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes

This is the Hard Version of the problem. The only difference between the two versions is that in the Hard Version the tree may be of any shape.

This problem is interactive.

Baudelaire is very rich, so he bought a tree of size n, rooted at some arbitrary node. Additionally, every node has a value of 1 or -1.

Cow the Nerd saw the tree and fell in love with it. However, computer science doesn't pay him enough, so he can't afford to buy it. Baudelaire decided to play a game with Cow the Nerd, and if he won, he would gift him the tree.

Cow the Nerd does not know which node is the root, and he doesn't know the values of the nodes either. However, he can ask Baudelaire queries of two types:

- $1\ k\ u_1\ u_2\ \ldots\ u_k$: Let $f(u)$ be the sum of the values of all nodes in the path from the root of the tree to node u. Cow the Nerd may choose an integer k $(1 \leq k \leq n)$ and k nodes $u_1, u_2, \ldots, u_k$, and he will receive the value $f(u_1) + f(u_2) + \ldots + f(u_k)$.
- $2\ u$: Baudelaire will toggle the value of node u. Specifically, if the value of u is 1, it will become -1, and vice versa.

Cow the Nerd wins if he guesses the value of every node correctly (the values of the final tree, **after** performing the queries) within $n + 200$ total queries. Can you help him win?

Input

The first line of the input contains a single integer t $(1 \leq t \leq 100)$, the number of test cases.

The first line of each test case contains a single integer n $(2 \leq n \leq 10^3)$, the size of the tree.

Each of the next $n - 1$ lines contains two integers u and v $(1 \leq u, v \leq n, u \neq v)$, denoting an edge between nodes u and v in the tree.

It is guaranteed that the sum of n over all test cases does not exceed 10^3 and that each graph provided is a valid tree.

Interaction

To ask a query of type 1, output a line in the following format (without the quotes):

- "`? 1 k` $u_1\ u_2\ \ldots\ u_k$", $(1 \leq k, u_i \leq n)$

The jury will return a single integer, $f(u_1) + f(u_2) + \ldots + f(u_k)$.

To ask a query of type 2, output a line in the following format:

- "`? 2` u" $(1 \leq u \leq n)$

The jury will toggle the value of node u: if its value is 1, it will become -1 and vice versa.

When you have found the answer, output a single line in the following format:

- "`!` $v_1, v_2, \ldots, v_n$" ($v_i = 1$ or $v_i = -1$, and v_i is the value of node i after performing the queries)

After that, proceed to process the next test case or terminate the program if it was the last test case. Printing the answer does not count as a query.

The interactor is **not** adaptive, meaning that the values of the tree are known before the participant asks the queries.

If your program makes more than $n + 200$ queries, your program should immediately terminate to receive the verdict `Wrong Answer`. Otherwise, you can get an arbitrary verdict because your

solution will continue to read from a closed stream.

After printing a query do not forget to output the end of line and flush the output. Otherwise, you may get the `Idleness Limit Exceeded` verdict. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Hacks

For hacks, use the following format.

The first line should contain a single integer t $(1 \le t \le 100)$ — the number of test cases.

The first line of each test case must contain exactly two integers n and $root$ $(2 \le n \le 10^3, 1 \le root \le n)$ — the size of the tree and the root of the tree.

The second line of each test case must contain exactly n integers $a_1, a_2, \ldots, a_n$ $(|a_i| = 1)$ — where a_i is the value of node i.

Each of the following $n - 1$ lines must contain exactly two integers u and v $(1 \le u, v \le n)$ — denoting an edge of the tree between nodes u and v.

The sum of n over all test cases must not exceed 10^3 and every graph provided must be a valid tree.

Example

input Copy

```
3
4
1 4
4 2
2 3

1

-1


-5

-5

2
1 2

2

7
1 2
2 7
7 3
7 4
7 5
7 6

-1
```

output Copy

```
? 1 3 1 2 4

? 1 2 3 1

? 2 4
? 1 3 1 2 4

? 1 2 3 1

! -1 -1 -1 -1


? 1 1 1
```

```
! 1 1




? 1 1 1

! -1 1 1 1 1 1 -1
```

Note

In the first example, the root of the tree is node 4 and the values are: $[-1, -1, -1, 1]$ (the i-th value is the value of node i).

Initially, $f(1) = 0, f(2) = 0, f(3) = -1, f(4) = 1$. Therefore, the answer to our first query is $f(1) + f(2) + f(4) = 1$, and of the second query is $f(3) + f(1) = -1$.

After toggling the value of node 4, the values are $[-1, -1, -1, -1]$. In addition, $f(1) = -2, f(2) = -2, f(3) = -3, f(4) = -1$. Therefore, $f(1) + f(2) + f(4) = -5$ and $f(3) + f(1) = -5$.

We answer that the final values of the nodes are $[-1, -1, -1, -1]$, which is correct. Notice that we report the values of the nodes **after** the changes, and not before.

In the second example, the root of the tree is 2 and the initial values are $[1, 1]$.

In the last example, the root of the tree is 1 and the initial values are $[-1, 1, 1, 1, 1, 1, -1]$.

Note that this is just an explanation of how the queries work, and it is not supposed to use any specific strategy to solve the problem.