

Three Keys of Wisdom

Story

The well-known dream team, Andela and her cat Luna, has once again found itself in a situation from the world of songs by the Italian band *Rhapsody of Fire*.

Andela and Luna set out on an epic journey through the Enchanted Lands. According to an ancient legend, the lands are ruled by N magical places, numbered $\{0, 1, \dots, N - 1\}$, connected by $N - 1$ mysterious paths. These paths are so ancient that all maps of them have long since disappeared into the fog of oblivion. What is known for certain is that the mysterious paths are bidirectional, and that each magical place is connected to at most 18 other magical places.

Somewhere deep in enchanted forests, dusty valleys and dark swamps, three keys of wisdom are hidden. But without knowing the paths, neither Andela nor Prince Arvald's army can find them.

As you already know, Luna is far from an ordinary cat. She can sense the age of magical places. When Andela shows her three places u , v and w , Luna senses the place where people from those three places used to gather: the unique place that lies simultaneously on the path from u to v , on the path from u to w , and on the path from v to w . Keep in mind that this place may even be one of u , v or w .

Andela knows that this is her only chance. Using Luna's sense, she must reconstruct the entire network of paths in the Enchanted Lands and map all $N - 1$ paths. Only then can she understand the secrets of civilizations long gone. But mistakes are costly! Andela must not make even a single mistake when she attempts to map a path.

Task

This is an **interactive task**. Therefore, **standard input and output are not used**. Instead, the contestant communicates with the system and tries to determine the final answer. If your code uses standard input/output, it is not guaranteed to receive points, regardless of whether it is otherwise correct.

You need to implement the function `solve(N)`, which may call:

- `meetingPlace(u, v, w)` -- Luna senses the meeting place of u , v and w . The function returns the index of a magical place. It may be called at most 100 000 times. A call is valid

only if u , v and w are pairwise distinct indices from the set $\{0, 1, \dots, N - 1\}$.

If the allowed number of calls is exceeded, or if u , v and w are not pairwise distinct, the solution is immediately judged as `Wrong Answer`.

- `mapPath(u, v)` -- Andela maps the path between places u and v . Every path (edge in the graph) must be reported exactly once, and only if the path really exists. If the reported path does not actually exist, the solution is immediately judged as `Wrong Answer`.

Submission Instructions

In the statement section of the task, there are three files: `grader.cpp`, `grader.h` and `kljucevi.cpp`. You need to modify the file `kljucevi.cpp` and submit that file.

Note: Your submission must contain the line `#include "grader.h"`.

Using the Grader

To test your code, you should compile and run the sample grader. It is located in the file `grader.cpp`. This file expects an implementation of the `solve` function, so you need to copy everything from your `kljucevi.cpp` file and paste it at the very top of `grader.cpp`. Then compile and run `grader.cpp` like any other C++ file.

Input for the Sample Grader

When you run `grader.cpp`, you must provide input for the grader.

The first line of input contains the number N -- the number of magical places.

Each of the next $N - 1$ lines contains two values u_i and v_i ($0 \leq i \leq N - 2$), the indices of two magical places between which there is a path.

Note: The solution itself **must not** read from standard input or communicate with other files. The grader receives the input data.

Output of the Sample Grader

When the program finishes successfully, the sample grader prints:

- If the solution is correct, it prints `Accepted` and returns the number of calls to `meetingPlace`.
- If the solution is incorrect, it prints `Wrong Answer`.

Note: The solution is **not printed** to standard output. Instead, it is constructed through calls to `mapPath`.

Constraints

- $3 \leq N \leq 2\,000$
- $0 \leq u_i < v_i \leq N - 1$ ($0 \leq i \leq N - 2$)
- Each magical place is connected to at most 18 other magical places.

Subtasks

Subtask 1 (3 points)

- $N \leq 7$

Subtask 2 (11 points)

- $N \leq 50$

Subtask 3 (23 points)

- $N \leq 300$

Subtask 4 (63 points)

- No additional constraints.

Example (for the sample grader)

Input 1

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

Function Calls 1

```
meetingPlace(0, 1, 2)    // returns 0
meetingPlace(0, 3, 4)    // returns 1
mapPath(1, 3)            // returns nothing
mapPath(0, 2)            // returns nothing
mapPath(1, 4)            // returns nothing
mapPath(0, 1)            // returns nothing
```

Output 1

Accepted: 2