Hedge Maze III

Jacques has trapped you in a maze — a hedge maze!

You know that the maze has N rooms, numbered from 1 to N, and that there are $N-1$ passageways that each connect two different rooms. If you follow the passageways in either direction, it is possible to travel between any pair of rooms. However, Jacques has not told you where these passageways are — your goal is to discover all of them.

To help you escape, Jacques has agreed to answer *queries*. In each query, you provide two distinct rooms A and B. Jacques will tell you the room with the **lowest index** that lies on the unique path from room A to room B, **excluding** rooms A and B themselves. If rooms A and B are directly connected by a passageway (so that no other rooms lie between them), Jacques responds with -1.

You must recover the entire structure of the maze using a limited number of queries. See the scoring section for details.

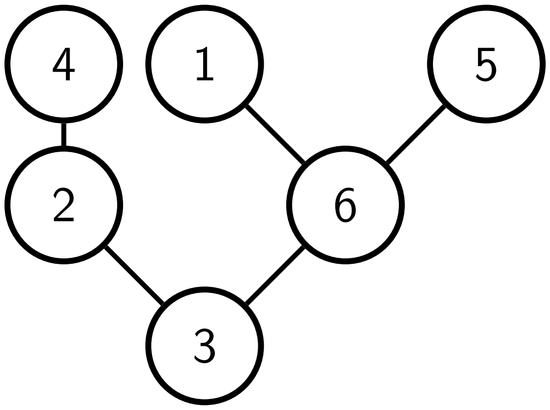

Figure 1: An example maze

Figure 1 shows a maze with $N = 6$ rooms and passageways between $(1, 6)$, $(5, 6)$, $(2, 3)$, $(3, 6)$, and $(2, 4)$. For example:

- If you ask a query with $A = 1$ and $B = 5$, the path from room 1 to room 5 passes through room 6. The only room between them is room 6, so Jacques responds with 6.
- If you ask a query with $A = 1$ and $B = 2$, the path passes through rooms 6 and 3. The lowest index between them is 3, so Jacques responds with 3.
- If you ask a query with $A = 2$ and $B = 4$, rooms 2 and 4 are directly connected. Jacques responds with -1.

Implementation Details

In this task, **do not read from standard input nor write to standard output.** Do not interact with any files. Do not implement a `main` function. Instead, begin your program by including the header file `maze.h` (`#include "maze.h"`) and interact with it as described below.

Functions

You must implement the function `map_maze`, which is called once in every test case:

```
void map_maze(int N);
```

- `N` is the number of rooms in the maze.
- From this function you can call `query`.
- You must call `report_passageway` exactly $N - 1$ times: once for each passageway in the maze.

From `map_maze`, you can make calls to the following functions:

```
int query(int A, int B);
```

- `A` and `B` must be two distinct rooms, each between 1 and N inclusive.
- This function returns the room with the lowest index on the path from A to B, excluding A and B.
- If A and B are directly connected by a passageway, this function instead returns -1.
- **Your score depends on the number of times this function is called. See the scoring section for details**.

```
void report_passageway(int A, int B);
```

- You should call this function for each passageway between rooms A and B.
- You must call this function exactly $N - 1$ times: once for each passageway in the maze.
- Each passageway is bidirectional; it does not matter whether you report (A, B) or (B, A).

If you violate any of the conditions listed above, your program will be judged as incorrect and you will receive 0% of the points for the test case.

The grader is not adaptive. This means that the answers to all queries are based on a fixed input.

Subtasks and Constraints

For all subtasks:

- $2 \leq N \leq 1\,000$.
- There are $N - 1$ passageways in the maze.
- It is possible to travel between any pair of rooms using the passageways.

Additional constraints for each subtask are given below.

Subtask	Points	Additional constraints
1	25	For every room i, there are less than 10 rooms between room i and room N.
2	20	The maze forms a line: that is, every room is connected to at most 2 other rooms.
3	15	For all $i < N$, room i is connected to at most 2 other rooms. Room N may be connected to any number of other rooms.
4	40	No additional constraints.

Scoring

If your solution fails to find all the passageways, or violates any of the conditions in the Implementation Details section, your score will be 0.

Otherwise, let Q be the number of calls that your solution makes to `query`:

- If $Q \leq 15\,000$, you will receive 100% for the test case.
- If $Q \leq 100\,000$, you will receive 20% for the test case.
- If $Q \leq 1\,000\,000$, you will receive 10% for the test case.
- Otherwise, you will receive 0% for the test case.

Your score for a subtask will be the **minimum** score of all test cases in the subtask, multiplied by the number of points you can score in the subtask.

Experimentation

In order to experiment with your code on your own machine, first download the provided files `maze.cpp`, `maze.h` and `grader.cpp`.

You should modify `maze.cpp`, which contains a basic example implementation.

Compile your solution with:

```
g++ -std=c++20 -O2 -Wall maze.cpp grader.cpp -o maze
```

This will create an executable `maze` which you can run with `./maze`. If you have trouble compiling, please send a message in the Communication section of the contest website.

The provided grader reads from standard input in the following format:

- The first line of input contains N.
- The next $N-1$ lines of input describe the passageways. The ith such line contains two integers a and b, representing a passageway between rooms a and b.

Note that the sample grader checks whether the provided passageways form a valid input.

Sample Grader Input & Sample Session

The sample grader is supplied with the following input:

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

This corresponds to the diagram on the first page.

One possible interaction is described below:

Grader	Student	Description
`map_maze(6)`		The grader calls your program.
	`query(1, 5)`	You ask a query for the path from room 1 to room 5.
returns `6`		The path passes through room 6, so Jacques responds with 6.
	`query(1, 2)`	You ask a query for the path from room 1 to room 2.
returns `3`		The path passes through rooms 6 and 3. The lowest index is 3.
	`query(2, 4)`	You ask a query for the path from room 2 to room 4.
returns `-1`		Rooms 2 and 4 are directly connected, so Jacques responds with -1.
	`report_passageway(1, 6)`	You report a passageway between rooms 1 and 6. This is correct.
	...	(You continue until all 5 passageways are reported.)