

HEDGE MAZE II

Perrito has found himself stuck in a hedge maze!

He knows 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 the passageways are bidirectional, then it is possible to travel between any pair of rooms using the passageways. However, to make the maze more difficult, each passageway **can only be travelled in one direction**. Because of this, it is not necessarily possible to travel between every pair of rooms.

Unfortunately, Perrito does not know where these passageways are and his first goal is to discover all the passageways. That is, he must determine the $N - 1$ pairs of rooms that are connected by passageways, and the direction of each passageway.

To do this, the maze designer has allowed Perrito to ask *queries*. In each query, Perrito provides the maze designer with two non-empty arrays A and B :

- The arrays A and B each contain integers from 1 to N , inclusive.
- The arrays must be disjoint and contain distinct integers. That is, each integer can appear at most once across both arrays.

The maze designer will answer each query with a single integer: the number of pairs of integers (a, b) , where a is in the array A and b is in the array B , for which there exists a path in the maze from room a to room b .

Let $|A|$ represent the number of integers in the array A , and $|B|$ represent the number of integers in the array B . Then, the *cost* of the query is $|A| \times |B|$. Perrito is allowed to ask queries with a total cost of up to 10 000 000. Additionally, your score depends on the *number* of queries asked. See the scoring section for details.

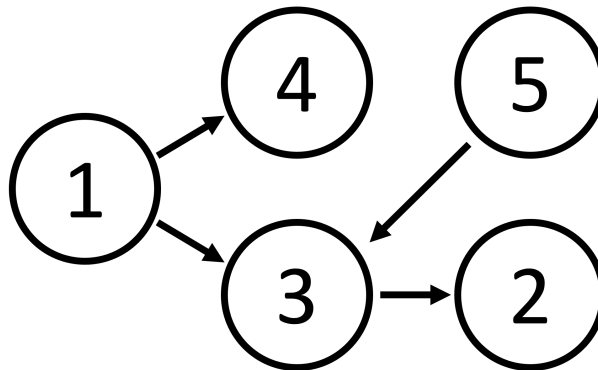


Figure 1: An example maze

Figure 1 has a maze with $N = 5$ rooms. If Perrito asked a query with $A = [1, 3]$ and $B = [2, 4]$, then the maze designer would answer with 3:

- There exists a path from room 1 to room 2, room 1 to room 4, and room 3 to room 2.
- There does not exist a path from room 3 to room 4.

The cost of this query is $2 \times 2 = 4$.

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.

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

```
int query(vector<int> A, vector<int> B);
```

- `A` and `B`: Your chosen arrays A and B .
- A and B must both be non-empty.
- A and B must contain integers from 1 to N inclusive, and each integer can appear at most once across both of the arrays.
- **The sum of $|A| \times |B|$ across all queries can be at most 10 000 000.**
- This function returns the number of pairs of integers (a, b) , where a is in the array A and b is in the array B , such that there exists a path in the maze from room a to room b .
- **Your score depends on the number of times this function is called. See the scoring section for details.**

```
void report_passageway(int i, int j);
```

- You should call this function if there is a directed passageway from room i to room j .
- You should call this function exactly $N - 1$ times: once for each passageway in the maze.

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 answers to all queries are based on a fixed input.

Subtasks and Constraints

For all subtasks:

- $3 \leq N \leq 1\,000$.
- There are $N - 1$ passageways in the maze.
- If passageways are made bidirectional, then it is possible to travel from any room to any other room using the passageways.

Additional constraints for each subtask are given below.

Subtask	Points	Additional constraints
1	10	Each passageway has one of its endpoints at room 1.
2	18	There exists a room i , such that every passageway has one of its endpoints at room i .
3	22	There exists a room i , such that for every other room j , there exists a path from room i to room j .
4	50	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`:

- Subtask 1, 2, and 3: You score all the points for these subtasks if $Q \leq 13\,000$, and 0 points otherwise.
- Subtask 4:
 - If $Q \leq 10\,000$, your score is 50. Otherwise,
 - If $Q \leq 11\,000$, your score is 43. Otherwise,
 - If $Q \leq 12\,000$, your score is 36. Otherwise,
 - If $Q \leq 13\,000$, your score is 31. Otherwise,
 - If $Q \leq 999\,000$, your score is 20. Otherwise,
 - Your score is 0.

Your score for a subtask will be the **minimum** score of all test cases 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++17 -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 i th such line contains two integers a and b , representing a one-way passageway from room a to room b .

Note that the sample grader does not check whether the provided passageways form a valid input.

Sample Grader Input & Sample Session

The sample grader is supplied with the following input:

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

This corresponds to the diagram on the first page.

One possible interaction is described below:

Grader	Student	Description
<code>map_maze(5)</code>		The grader calls your program.
	<code>query({1, 3}, {2, 4})</code>	You ask a query with $A = \{1, 3\}$ and $B = \{2, 4\}$.
returns 3		The grader responds with 3. This is explained in the example on the first page.
	<code>report_passageway(1, 4)</code>	You report that there is a passageway from room 1 to room 4. This is correct.
	<code>query({5}, {1})</code>	You ask a query with $A = \{5\}$ and $B = \{1\}$.
returns 0		There is not a path from room 5 to room 1, and so the grader returns 0.
	<code>report_passageway(1, 5)</code>	You report that there is a passageway from room 1 to room 5. This is incorrect, and your program will score 0.
