

Problem 9: Dijkstra in Dungeon

4+7 Points

Problem ID: `explorer`

Rank: 3+3

Introduction

Laios, Marcille, Chilchuck, and Senshi are navigating through [the dungeon](#) to save Laios's sister Falin from the Red Dragon! Fortunately, Marcille can cast [clairvoyance](#) spells to scout out the dungeon, but she's low on mana and doesn't want to use any more mana than needed. She's also not very good at clairvoyance because she specializes in explosion spells instead. To find the shortest path there, they resurrected and enlisted the legendary computer scientist catboy [Edsger W. Dijkstra](#). Help the party find their way to the Red Dragon before it's too late!



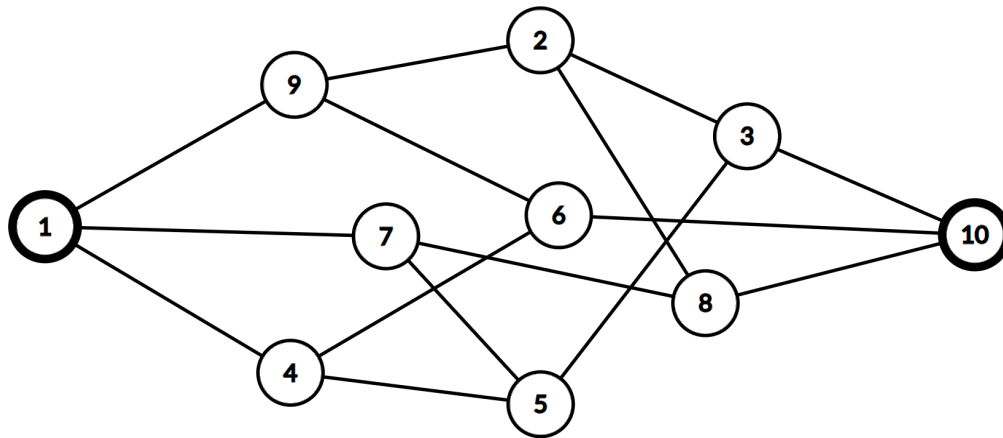
Problem Statement

This is an interactive problem! Please carefully read the Interaction Format before attempting.

The judge begins each test case by generating a connected [random 3-regular graph](#) with 500 vertices randomly labeled 1 to 500. In other words, of all possible graphs with these properties:

- The graph has exactly **500** vertices.
- The edges are undirected and unweighted.
- Every vertex has exactly **3** edges connecting it to exactly **3** other vertices.
- There always exists a path between every pair of vertices.

The judge generates a graph by choosing one of these graphs **at random**. This graph is initially hidden (**not given to your program**). Here's an example of a 3-regular graph with 10 vertices:



Your program must find the **length of the shortest path** from the vertex labeled 1 to the vertex labeled 500 in the hidden graph. To do this, your program can perform two types of queries:

- *Scan* a vertex to learn more about the graph:
 - Choose any vertex to scan. Of the three vertices connected to it, the judge chooses one **at random** and gives the label of that vertex back to your program.
 - *Scan* queries can be performed **any number of times** on **any vertex**.
- *Submit* your guess for the length of the shortest path:
 - If your guess is correct, the judge proceeds to the next test case.
 - If your guess is incorrect, you receive a wrong answer verdict.

Each test file contains **exactly 500 test cases** that your program will be run on. At the end of each test file, if the **average number of scan queries used per test case is no more than the threshold** as specified in the Constraints, your submission will receive a correct verdict. Otherwise, your submission will receive a wrong answer verdict.

Interaction Format

This is an interactive problem! Unlike regular problems, your program and the judge will run simultaneously. Please see the [contest guide](#) for more information. Please flush your buffer as instructed by [this post](#) when you output, or use our template code that handles it for you. If you run into technical issues with interaction, please let us know with a clarification request!

Begin by reading a single line containing a single integer **T** denoting the number of test cases. For each test case:

- Make any number of *scan* queries to learn about the graph. To make a *scan* query:
 - First, output a single line in the following format:
`SCAN v`
where *v* is an integer denoting the label of the vertex to scan.
 - Then, read a single line containing an integer denoting the label of a random vertex adjacent to the vertex labeled *v*.
- Make a single *answer* query to submit your guess. To make an *answer* query:
 - First, output a single line in the following format:
`SUBMIT d`
where *d* is an integer denoting your guess for the length of the shortest path.
 - Then, read a single line containing either `CORRECT` or `WRONG_ANSWER`
 - If `CORRECT`, the judge proceeds to the next test case.
 - If `WRONG_ANSWER`, your program should exit.

If at any point your program deviates from the interaction format (e.g. invalid query type, making a *scan* query with a negative number, etc.), the judge will send `WRONG_ANSWER`. If your program reads `WRONG_ANSWER` at any point, you should exit to receive a wrong answer verdict. If your program does not exit, you will receive a time limit exceeded verdict.

Note: After submitting, you can click your result to see submission details which includes an interaction log between your program and the judge for sample test cases:

time	problem	lang	result	cases
04:25	WTDS_M	PY3	WRONG- ANSWER	 

Constraints

Time limit: **5 seconds**

Memory limit: **256 MB**

T = 500

Note that **T** is equal to exactly 500, not less than or equal to.

Main Test Set

To pass, the average number of scan queries per test case in each test file must be no more than **750**.

Bonus Test Set

To pass, the average number of scan queries per test case in each test file must be no more than **150**.



Sample Interaction

The line spacing here is to emphasize the order in which interaction takes place only. Do not expect or output blank lines between each line of interaction. **Also, note that the value for T has been adjusted for sample purposes.** Real test cases will always have T at 500.

Sample Input

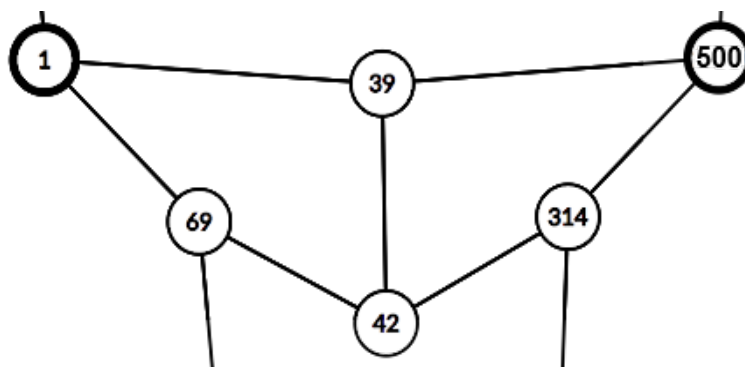
Sample Output

```
2 |
  | SCAN 1
69 |
  | SCAN 1
39 |
  | SCAN 1
39 |
  | SCAN 39
500 |
  | SCAN 314
500 |
  | SCAN 69
1 |
  | SUBMIT 2
CORRECT |
  | SCAN 500
1 |
  | SCAN 1
500 |
  | SUBMIT 1
CORRECT |
```

Sample Explanations

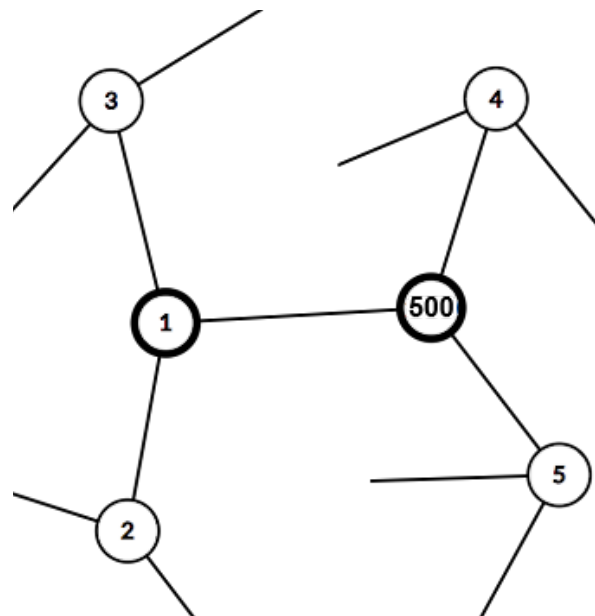
The judge begins by outputting 2, the number of test cases.

For test case #1, the judge generates the following graph. Since it is quite large, some vertices have been omitted from this illustration:



The program begins this test case by sending a *scan* query with $v = 1$. Vertex 1 is connected to vertex 39, vertex 69, and some other not shown vertex in the graph. The judge chooses at random from one of these and the program receives 69. The program then sends another 2 *scan* queries at vertex 1 and receives 39 twice. Note that the same vertex can be scanned multiple times and the same vertex can also be given by the judge multiple times. The program then scans vertices 39, 314, and 69 and receives 500, 500, and 1. Finally, the program sends a *submit* query with $d = 2$. Since this is correct, the judge responds `CORRECT` and proceeds to the next test case.

For test case #2, the judge generates the following graph. Since it is quite large, some vertices have been omitted from this illustration:



The program begins this test case by sending *scan* queries at vertices 500 and 1 and the judge responds with 1 and 500. The program then sends a *submit* query with 1. Since this is correct, the judge responds with `CORRECT`.

Now that all test cases have completed, we see that the program took 6 *scan* queries to complete the first test case and 2 *scan* queries to complete the second test case. Thus, the program used $(6 + 2) / 2 = 4$ queries on average. As this value is below the required average of **750** (and also below **150** for the bonus), the judge gives a `CORRECT` verdict. Upon receiving this line, the program exits.