

## Nav

Input File	Output File	Time Limit	Memory Limit
<b>none</b>	<b>none</b>	4 seconds	1024 MiB

In the latest video game you are playing, you find yourself in a mysterious world. The world can be represented as a connected graph over  $n$  vertices, numbered from 0 to  $n - 1$ . There are  $m$  bidirectional edges in the graph, numbered from 0 to  $m - 1$ , each connecting a different pair of vertices. All edges have the same length. You have been told the exact layout of this graph.

To test your vast knowledge, you will have to solve  $t$  tests.

In each test, one vertex has been secretly chosen as the *prime* vertex, and your task is to identify which vertex it is. To help you do so, you have been given a device called the *Nav*. The Nav is designed to help you navigate to the prime vertex. It does so by allowing you to query it, in the following manner.

1. You input a vertex of your choosing, call it  $u$ . If  $u$  is the prime vertex, the Nav will return  $-1$ .
2. Otherwise, the Nav computes a **shortest path** from  $u$  to the prime vertex. Note that if there is more than one such path, the Nav can choose any of these.
3. The Nav returns an integer between 0 and  $m - 1$ , corresponding to a **single edge** that is a part of the computed shortest path.

Your task is to identify the prime vertex, using as few Nav queries as possible.

The fewer queries you use, the more points you will score for this problem. See the *Subtasks and Constraints* section below for further details.

### Subtasks and Constraints

For all subtasks, you are guaranteed that:

- $n \geq 1$
- $n - 1 \leq m \leq 100\,000$
- The provided graph will be connected
- Every edge will connect two *different* vertices
- No two edges connect the same pair of vertices.

For each subtask, there is a maximum number  $q$  of queries you can make to the Nav **per test**. Your score for a testcase is computed as follows:

- If you make more than  $q$  queries for any test, you will receive 0 points. This will be reported as **Not correct** in the contest system.
- If your program exceeds the time limit, you will receive 0 points.
- If your program does not successfully terminate, or fails to identify the prime vertex, you will receive a score of 0 points. This will be reported as **Not correct** in the contest system.
- Otherwise, if your program correctly identifies the prime vertex in all cases:
  - For Subtasks 1 to 3: you will score 100% of the marks for that subtask. This will be reported as **Correct** in the contest system.

– For Subtask 4, your score is computed, based on the number of queries  $q_{\max}$  that your program made in any test. Specifically, your score will be:

- \* 100%, if  $q_{\max} \leq 9$ ;
- \* 70%, if  $10 \leq q_{\max} \leq 12$ ; or
- \*  $\frac{2}{3} \times \frac{30 - q_{\max}}{q_{\max}}$ , if  $13 \leq q \leq 30$ .

Subtask	Points	Maximum Maximum			Additional constraints
		$t$	$n$	$q$	
1	7	750	$n \leq 300$	300	
2	15	250 000	$n \leq 100\,000$	17	$m = n - 1$ , and for each $0 \leq j \leq m - 1$ , edge $j$ connects vertex $j$ and vertex $j + 1$ . That is, the graph is a line.
3	26	750	$n \leq 1000$	10	$m = n - 1$ . That is, the graph is a tree.
4	52	750	$n \leq 300$	30	

Recall that your score for this problem is the sum of your scores for each subtask. Your score for a subtask is the maximum score obtained among any of your submissions. As such, you may wish to solve different subtasks in different submissions.

## Input / Output

This task has no input or output files. Instead, your solution must interact with the functions in the header file "nav.h". The provided functions are described in detail in the next section. **Do not output anything to stdout, or you will receive 0% points for the test case.**

## Implementation

You **must not** implement a `main` function. Instead, you should `#include "nav.h"` and implement the two functions `init` and `findPrime`, described below:

```
void init(int subtask, int n, int m, std::vector<int> A, std::vector<int> B);
```

where:

- `subtask` is the subtask number
- `n` is the number of vertices
- `m` is the number of edges
- For every  $0 \leq j \leq m - 1$ , edge  $j$  connects vertices `A[j]` and `B[j]`.
- This function will be called first to initialise your program.

```
int findPrime();
```

This function should return an integer between 0 and  $n - 1$ : the number corresponding to the prime vertex.

The grader will call this function  $t$  times, once for each test. Your program is not told  $t$ .

In your implementation, you may call the following function.

```
int nav(int u)
```

where:

- $u$  must be an integer, between 0 and  $n - 1$ , corresponding to the number of a vertex of your choosing.
- If  $u$  is the prime vertex, the function returns  $-1$ . Otherwise, the function returns an integer between 0 and  $m - 1$ , corresponding to an **edge** along a shortest path between  $u$  and the prime vertex.

If `nav` is called with an invalid value of  $u$ , the program will terminate, and you will score 0% for that testcase. This will be reported as `Not correct` in the contest system.

You may assume that over the course of a single test, the grader that is used does not change the identity of the prime vertex.

A template `nav.cpp` has been provided for your convenience.

## Experimentation

In order to experiment with your code on your own machine, first download the provided files `nav.h` and `grader.cpp`, which should be placed in the same directory as your code. Please note that the grader that is used may have **different** behaviour to the provided grader.

Compile your solution with:

```
g++ -std=c++11 -O2 -Wall nav.cpp grader.cpp -o nav
```

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

The compiled sample grader will read in the subtask number,  $n$  and  $m$  on the first line.

It will then read  $m$  more lines. From the  $i$ -th of these lines, the sample grader will read two integers `A[i]` and `B[i]`.

On the next line, it will read in the integer  $t$ , the number of tests, followed by  $t$  lines. From the  $j$ -th of these lines, it will read in the secret prime vertex for the  $j$ -th test.

For each test, the sample grader will call `findPrime()` and print out whether your program successfully found the prime vertex, along with the number of queries it made.

## Sample Grader Input and Sample Session

```
4 6 7
2 5
2 3
2 1
1 3
0 1
3 4
4 0
3
3
5
3
```

The grader begins by reading in the graph, and then interacts with your program.

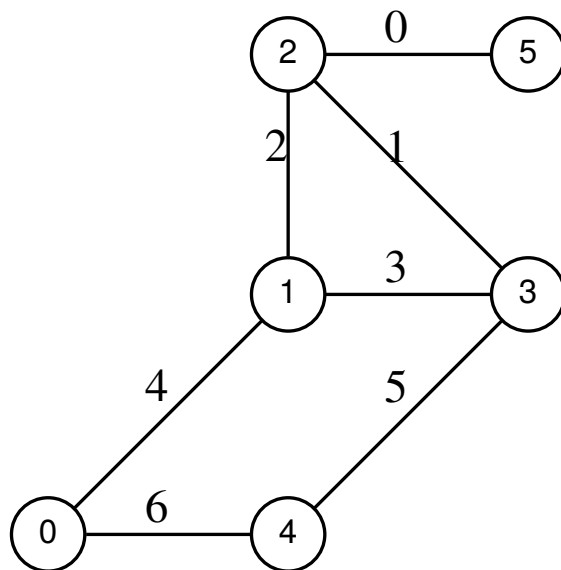


Figure 1: Sample Diagram

One possible interaction is shown below:

Grader	Student	Description
<code>init(4, 6, 7, [2, 2, 2, 1, 0, 3, 4], [5, 3, 1, 3, 1, 4, 0])</code>		The grader initialises your program. This sample case is for Subtask 4.
<code>findPrime()</code>		The grader starts the first test. The prime vertex is vertex 3
	<code>nav(0)</code>	Your program calls <code>nav</code> on vertex 0.
<code>nav</code> returns 6		The grader picks the shortest path $0 \rightarrow 4 \rightarrow 3$ , and chooses to return edge 6 (the edge between 0 and 4).
	<code>nav(2)</code>	Your program calls <code>nav</code> on vertex 2.
<code>nav</code> returns 1		The grader picks the shortest path $2 \rightarrow 3$ , and chooses to return edge 1 (the edge between 2 and 3).
	<code>nav(3)</code>	Your program calls <code>nav</code> on vertex 3.
<code>nav</code> returns -1		This is the prime vertex.
	You return 3	Your program guesses the prime vertex is vertex 3. This is correct.
<code>findPrime()</code>		The grader starts the second test. The prime vertex is vertex 5
	<code>nav(4)</code>	Your program calls <code>nav</code> on vertex 4.
<code>nav</code> returns 1		The grader picks the shortest path $4 \rightarrow 3 \rightarrow 2 \rightarrow 5$ , and chooses to return edge 1 (the edge between 2 and 3).

---

Grader	Student	Description
<code>findPrime()</code>	You return 1	Your program guesses the prime vertex is vertex 1. This is not correct.
		The grader starts the third test. The prime vertex is vertex 3 again.
	You return 3	Your program guesses the prime vertex is vertex 3, without making any calls to <code>nav</code> ! This is correct.

---

In this example, you have solved two of the three cases correctly, so you would score 0% points.