Homework #5

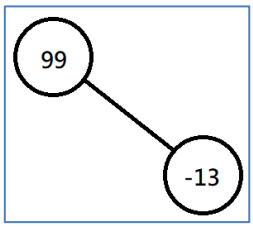
Simple Chain

(Due: 2021-12-15)

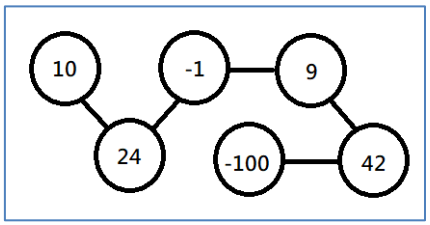
Overview

In this assignment, we define “Simple chain” as a path in a simple graph. A simple chain is basically a sequence of vertices of the form <V0, V1, V2, …… Vn> such that Vi and Vi+1 are connected by an edge. The vertices on the chain must be distinct, and the degree of each vertex should be exactly 2, except for the two endpoints.

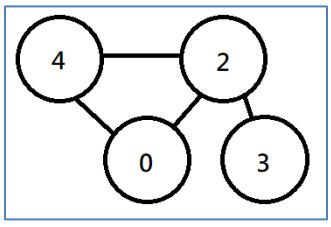
Take the following three graphs for example, graph A is a simple chain <99, -13> (or you can say <-13, 99>) and graph B is also a simple chain <10, 24, -1, 9, 42, -100> (<-100, 42, 9, -1, 24, 10>), while graph C is not.



Graph A



Graph B



Graph C

Consider a *simple graph with N vertices and N-1 edges*. Assume the vertices in the graph form a simple chain. Now, given an array of size N-1 representing the edges in the graph, please design an algorithm to restore the chain sequence and print the ith element in the sequence.

Input.txt

The first line of input is the number of test cases **T**. Each test case begins with two integer **N** and **I**, meaning that there are N vertices in the graph, and you must print the **I**th element in chain sequence (the index of the sequence starts from 0).
Following are N-1 pairs of integers separate by a single space. Each pair of integers <Ui, Vi> indicate that vertex Ui and vertex Vi are connected. It is guaranteed that every edge in the graph will be given exactly once, but the edges could appear in any order.

Let's take a look at the following test case. There are six vertices in the graph, and vertex <-1> and <9> are connected, <9> and <42> are connected and so on. this test case will form a simple chain exactly as the chain shown in graph B.

```
6 3
-1 9
9 42
-100 42
24 10
24 -1
```

Output.txt

For each test case, output the ith element in the chain sequence. Since there are two endpoints in the chain, choose the chain that starts from the larger value, for example, graph B's chain should be <10, 24, -1, 9, 42, -100> instead of <-100, 42, 9, -1, 24, 10>, and it's 3rd element is <-1>.

Sample Input

```
2
2 0
-1000000 1000000
6 5
24 10
-1 9
42 -100
24 -1
9 42
```

Sample Output

```
1000000
-100
```

Constraints

1 <= T <= 10

2 <= N <= 50000

0 <= I <= N-1

-10^7 <= value of each node <= 10^7

Preloaded Input Data

```
struct tTestCase {
	int n;
	int i;
	int edges[50000][2];
};

struct tTestData {
	int t;
	struct tTestCase testcase[10];
};
```