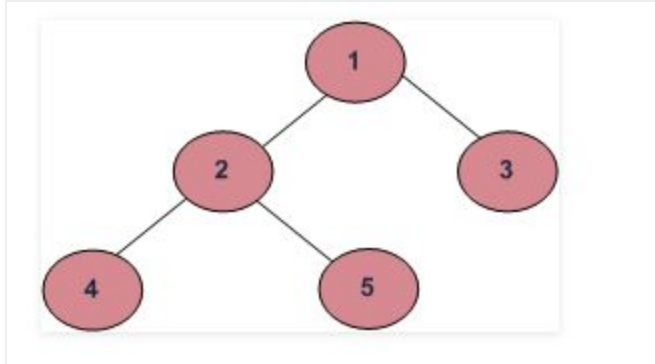


Problem Description

Given a binary tree, return the zigzag level order traversal of its nodes' values.
(i.e. from left to right, then right to left for the next level and alternate for every level).
For the tree given, zigzag traversal will be:



1
3 2
4 5

Input format

Line 1 has the number of test cases (T)

Line 2 to X: First Test Case details with the binary tree structure (refer section below for the format)

Line X+1 to Y: Second Test case details and so on.

Output format

For each test case, print K lines, where K is the height of the tree. In each line print the values of the zigzag traversed node values.

Constraints

$1 \leq T \leq 1000$

$1 \leq \text{Number of nodes in a Tree} \leq 10000$

$0 \leq \text{Value of each node} \leq 10^9$

It's guaranteed that the sum of the number of tree nodes across all test cases will be less than 500000.

Sample Input 1

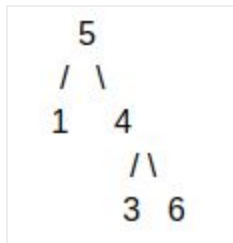
```
1
5
5 1 4 3 6
1 2 3
2 -1 -1
3 4 5
4 -1 -1
5 -1 -1
```

Sample Output 1

```
5
4 1
3 6
```

Explanation 1

Treeview



Instructions to create custom input for a Binary Tree

In order to specify a binary tree that can be used as custom input to the problem, you'll need to follow this convention.

- Line 1: Number of nodes in the Binary Tree (N)
- Line 2: N space separated node values. The position of the Nodes on this line will be used to refer to them in the below lines, starting from 1.
- Line 3 to N+2: Lines specifying the child nodes for each of the N nodes

Format of each line (space separated): Parent_node Left_child_node Right_child_node

* Parent_node - Node at this Position on Line 2 is the Node to which we are assigning the Left and Right child here

* Left_child_node - Node at this position on Line 2 is the left child. Specify -1 if there is no Left child.

* Right_child_node - Node at this position on Line 2 is the right child. Specify -1 if there is no Right child

Example1

If you want to create a Tree that looks like this:

```
  2
 /\
3 7
 /\
8 9
```

Your input would be:

5	→ Number of Nodes
2 3 7 8 9	→ Node values
1 2 3	→ Node 1(value 2) and its child nodes (left child value 3 and right child value 7)
2 4 5	→ Node 2(value 3) and its child nodes (left child value 8 and right child value 9)
3 -1 -1	→ Node 3(value 7) and its child nodes (left and right child are Null i.e. Leaf Node)
4 -1 -1	→ Node 4(value 8) and its child nodes (left and right child are Null i.e. Leaf Node)
5 -1 -1	→ Node 5(value 9) and its child nodes (left and right child are Null i.e. Leaf Node)