Flatten Binary Tree to Linked List

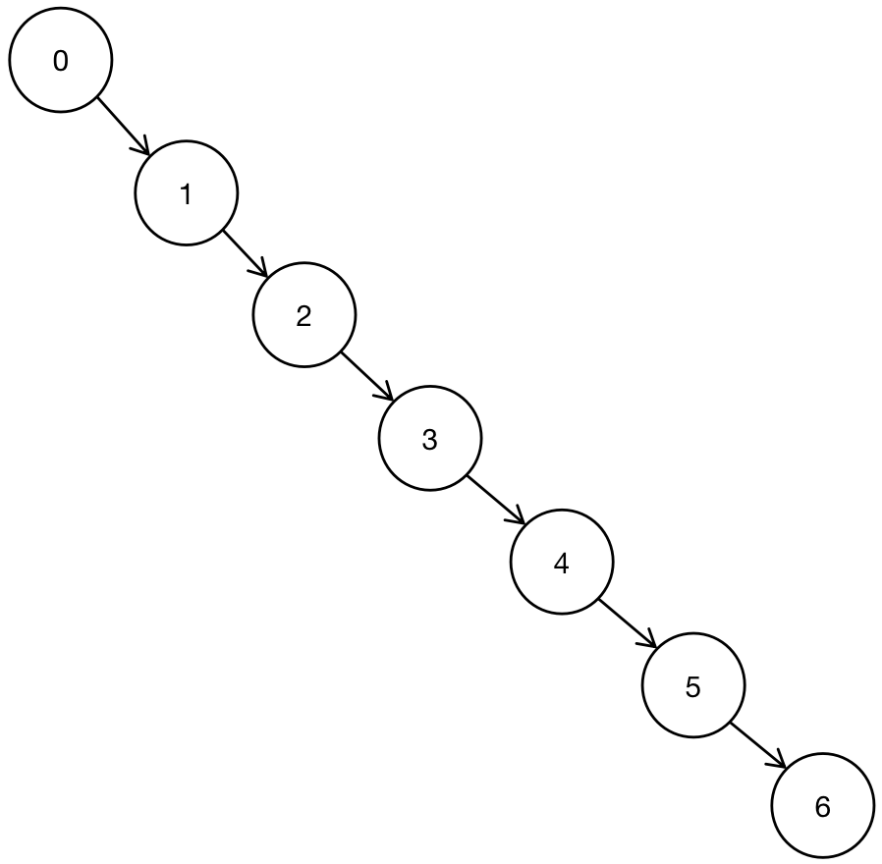
*Description:*

Given one Binary Tree, flatten it to a Linked List in - place.

*For example, given:*

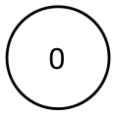


*Flattened Tree should look like:*



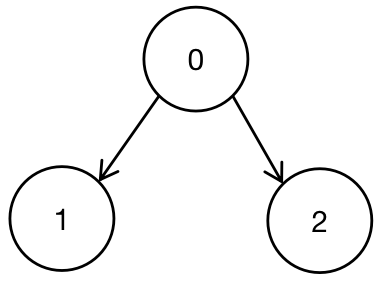
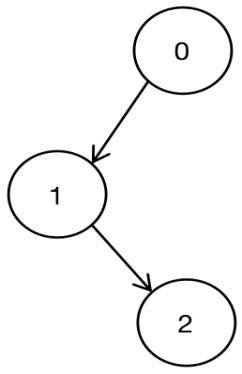
*Principle:*

* *Situation 0 ====> Left and Right Sub - Nodes are all NULL.*

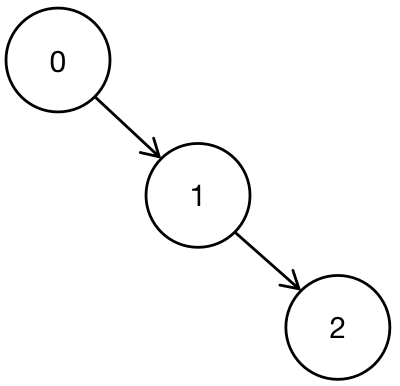
**

Left and Right Sub - Nodes are all NULL, then no need to move the node from left node to right node. Nothing would happen.

* *Situation 1 ====> Left and Right Sub - Nodes are all not NULL.*

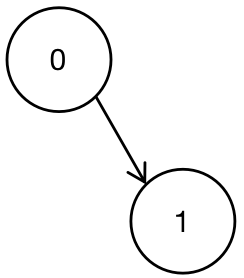
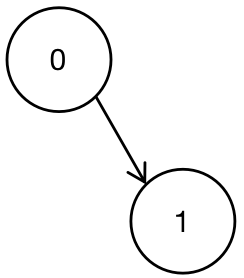
 

*Step 1:* Move Node 2 from Right Node of Node 0 to the Left Node of Node 1.



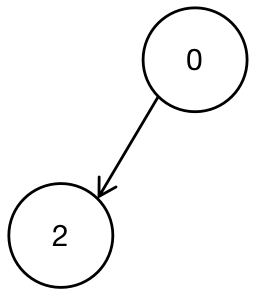
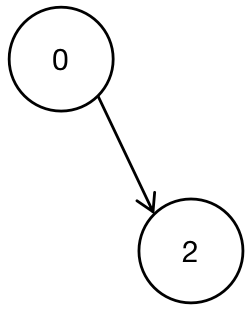
*Step 2:* Move Node 1 from Left Node of Node 0 to Right Node of Node 1.

* *Situation 2 ====> Left Node is NULL.*

No Operations.

* *Situation 3 ====> Right Node is NULL.*

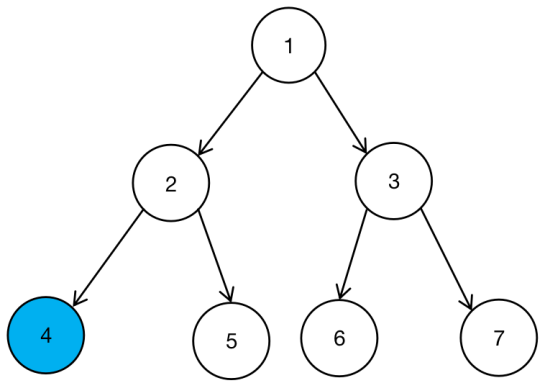
Move Left

Node of Node 0 to Right Node of Node 0.

*Analysis:*

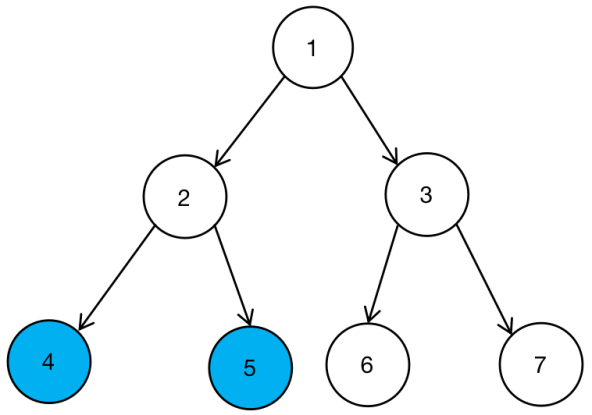
According to the example in the utmost, below is the analysis:

*The First Step -*



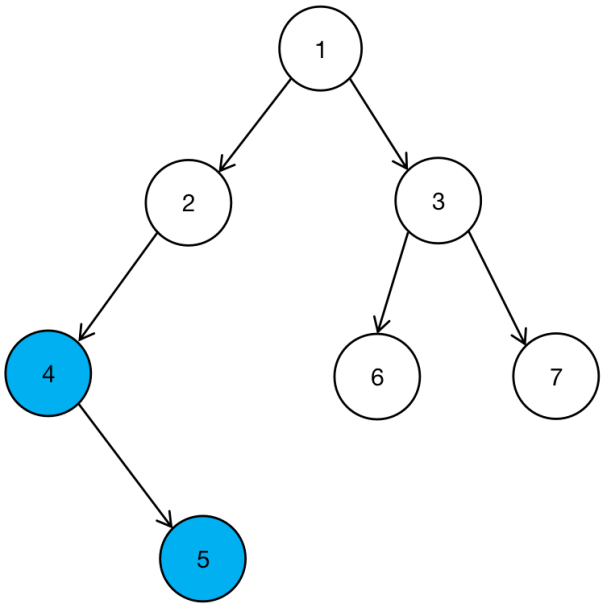
* *Find that Node 4 belongs to Situation 0.*

*The Second Step -*



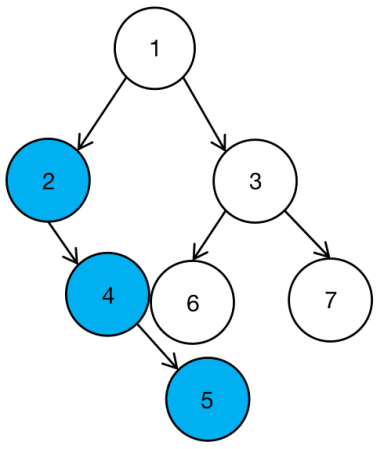
* *Find that Node 5 belongs to Situation 0.*

*The Third Step -*

**

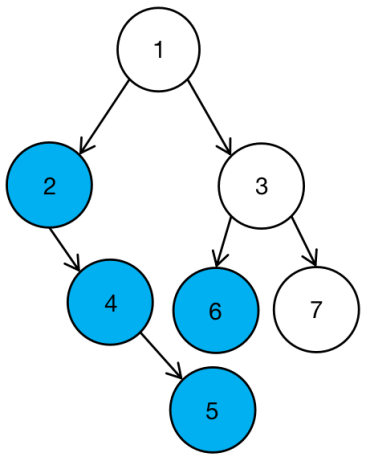
* *Check the situation of Node 2:*
* *First, goes through all Right Sub - Nodes of Node 4.*
* *Second, connects Right Node of Node 2 to the last Right Node of Node 4.*
* *Third, breaks up the Left Node of Node 2, and re - connects it to the Right Node of Node 2.*

*The Forth Step -*

**

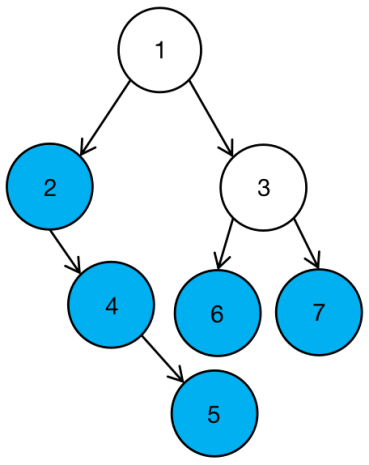
*After finishing Third Step, it turns to the situation above.*

*The Fifth Step -*

**

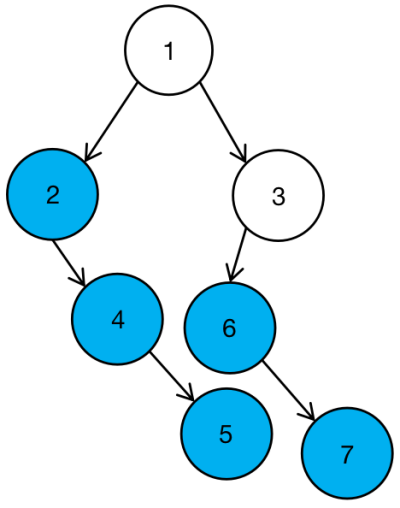
* *Check that Node 6 belongs to Situation 0.*

*The Sixth Step -*

**

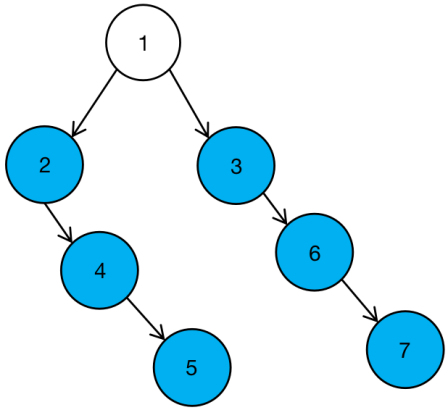
* *Check that Node 7 belongs to situation 0.*

*The Seventh Step -*

**

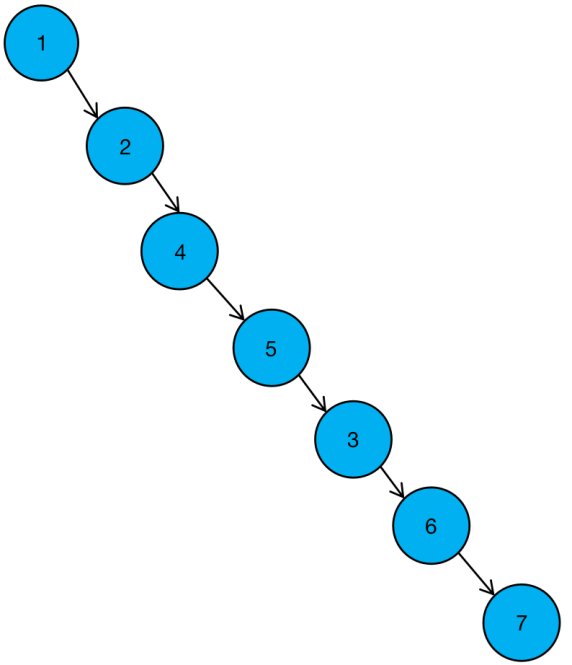
* *Check the situation of Node 3:*
* *First, goes through all Right Sub - Nodes of Node 6;*
* *Second, connects all Right Sub - Nodes of Node 3 to the last Right Node of Node 6.*
* *Third, breaks up the Left Node of Node 3, and reconnect it to Node 3.*

*The Eighth Step -*

**

* *Check the situation of Node 1:*
* *First, goes through all Right Sub - Nodes of Node 2;*
* *Second, connects the Right Node 3 all to the last Right Node of Node 5.*
* *Third, breaks up the Left Node of Node 1, and reconnect it to the Right Node of Node 1.*

*The Ninth Step -*

**

*Binary Tree has been flattened to the Linked List.*

*Code:*

void FlattenBinaryTree ( BinaryTreeNode \* root )

{

if ( root == NULL )

return;

*// Function:*

*// Flatten Binary Tree into Linked List Tree*

*// Steps:*

*// First, Flatten the left Sub - Tree of Root.*

*// Second, Flatten the right Sub - Tree of Root.*

FlattenBinaryTree ( root -> left );

FlattenBinaryTree ( root -> right );

*// If Left Node of Root equals to NULL, just skip the Current Flatten Procedure.*

if ( root -> left == NULL )

return;

*// Move all Children from the Right Tree of Root to the Left Tree of Root.*

*// Reset the Left Tree of Root.*

BinaryTreeNode \* node = root -> left;

while ( node -> right != NULL ) node = node -> right;

node -> right = root -> right;

root -> right = root -> left;

root -> left = NULL;

return;

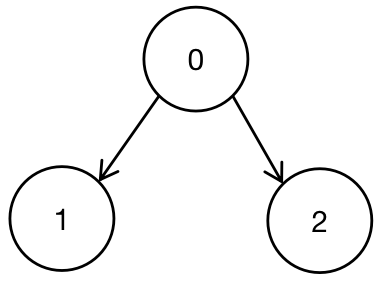
}

*Recursive Calling*

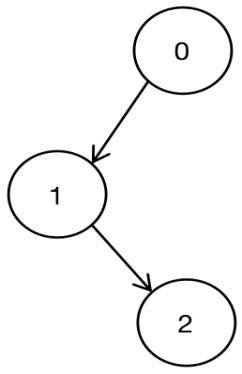
*Instruction - FlattenBinaryTree*

1. *Used as the whole procedure to re - adjust Root Node.*
2. *Concatenate the Newly Setting Right Sub-Tree with Right Child of Root Node.*
3. *Re - adjust the Left Child Node of Root Node.*

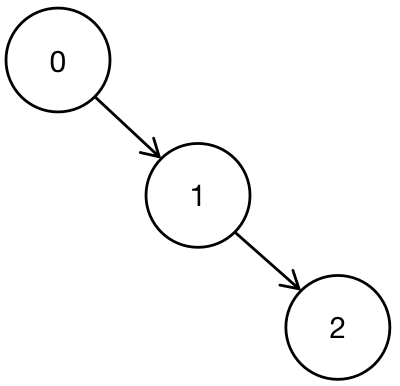
*Step 1:*



*Step 2:*



*Step 3:*



void GetFlattenBinaryTree ( )

{

FlattenBinaryTree ( root, NULL );

}

BinaryTreeNode \* FlattenBinaryTree ( BinaryTreeNode \*root, BinaryTreeNode \*tail )

{

if ( tail == NULL )

return tail;

*// FlattenBinaryTree ( BinaryTreeNode \*root, BinaryTreeNode \*tail ) is used to help // define the right root node of root. We need to return BinaryTreeNode \**

*// pointer back to the upper level.*

root -> right = *FlattenBinaryTree ( root -> left, FlattenBinaryTree ( root -> right, tail ) )* ;

root -> left = NULL;

return root;

}