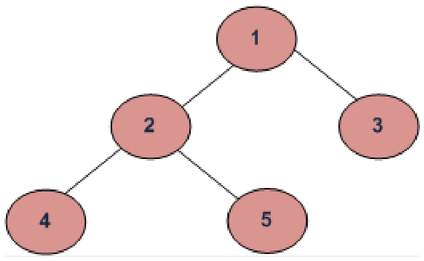
Iterative Method to find Height of Binary Tree

There are two conventions to define height of Binary Tree
1) Number of nodes on longest path from root to the deepest node.
2) Number of edges on longest path from root to the deepest node.

In this post, the first convention is followed. For example, height of the below tree is 3.



Example Tree

Recursive method to find height of Binary Tree is discussed here. How to find height without recursion? We can use level order traversal to find height without recursion. The idea is to traverse level by level. Whenever move down to a level, increment height by 1 (height is initialized as 0). Count number of nodes at each level, stop traversing when count of nodes at next level is 0.
Following is detailed algorithm to find level order traversal using queue.

```
Create a queue.
Push root into the queue.
height = 0
Loop
 nodeCount = size of queue

        // If number of nodes at this level is 0, return height
 if nodeCount is 0
  return Height;
 else
  increase Height

        // Remove nodes of this level and add nodes of
        // next level
 while (nodeCount > 0)
  pop node from front
  push its children to queue
  decrease nodeCount
       // At this point, queue has nodes of next level
```

Following is the implementation of above algorithm.

C++

```
/* Program to find height of the tree by Iterative Method */
#include <iostream>
#include <queue>
using namespace std;

// A Binary Tree Node
struct node
{
    struct node *left;
    int data;
    struct node *right;
};

// Iterative method to find height of Bianry Tree
int treeHeight(node *root)
{
    // Base Case
    if (root == NULL)
        return 0;

    // Create an empty queue for level order tarversal
    queue<node *> q;

    // Enqueue Root and initialize height
    q.push(root);
    int height = 0;

    while (1)
    {
        // nodeCount (queue size) indicates number of nodes
        // at current lelvel.
        int nodeCount = q.size();
        if (nodeCount == 0)
            return height;

        height++;

        // Dequeue all nodes of current level and Enqueue all
        // nodes of next level
        while (nodeCount > 0)
        {
            node *node = q.front();
            q.pop();
            if (node->left != NULL)
                q.push(node->left);
            if (node->right != NULL)
                q.push(node->right);
            nodeCount--;
        }
    }
}

// Utility function to create a new tree node
node* newNode(int data)
{
    node *temp = new node;
    temp->data = data;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

// Driver program to test above functions
int main()
{
    // Let us create binary tree shown in above diagram
    node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    cout << "Height of tree is " << treeHeight(root);
    return 0;
}
```

Java

```
// An iterative java program to find height of binary tree

import java.util.LinkedList;
import java.util.Queue;
import java.util.Queue;

// A binary tree node
class Node
{
    int data;
    Node left, right;

    Node(int item)
    {
        data = item;
        left = right;
    }
}

class BinaryTree
{
    Node root;

    // Iterative method to find height of Bianry Tree
    int treeHeight(Node node)
    {
        // Base Case
        if (node == null)
            return 0;

        // Create an empty queue for level order tarversal
        Queue<Node> q = new LinkedList();

        // Enqueue Root and initialize height
        q.add(node);
        int height = 0;

        while (1 == 1)
        {
            // nodeCount (queue size) indicates number of nodes
            // at current lelvel.
            int nodeCount = q.size();
            if (nodeCount == 0)
                return height;
            height++;

            // Dequeue all nodes of current level and Enqueue all
            // nodes of next level
            while (nodeCount > 0)
            {
                Node newnode = q.peek();
                q.remove();
                if (newnode.left != null)
                    q.add(newnode.left);
                if (newnode.right != null)
                    q.add(newnode.right);
                nodeCount--;
            }
        }
    }

    // Driver program to test above functions
    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();

        // Let us create a binary tree shown in above diagram
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        System.out.println("Height of tree is " + tree.treeHeight(tree.root));
    }
}

// This code has been contributed by Mayank Jaiswal
```

Python

```
# Program to find height of tree by Iteration Method

# A binary tree node
class Node:

    # Constructor to create new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Iterative method to find height of Binary Tree
def treeHeight(root):

    # Base Case
    if root is None:
        return 0

    # Create a empty queue for level order traversal
    q = []

    # Enqueue Root and Initialize Height
    q.append(root)
    height = 0

    while(True):

        # nodeCount(queue size) indicates number of nodes
        # at current level
        nodeCount = len(q)
        if nodeCount == 0 :
            return height

        height += 1

        # Dequeue all nodes of current level and Enqueue
        # all nodes of next level
        while(nodeCount > 0):
            node = q[0]
            q.pop(0)
            if node.left is not None:
                q.append(node.left)
            if node.right is not None:
                q.append(node.right)

            nodeCount -= 1


# Driver program to test above function
# Let us create binary tree shown in above diagram
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

print "Height of tree is", treeHeight(root)

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Output:

```
Height of tree is 3
```

Time Complexity: O(n) where n is number of nodes in given binary tree.