Populating Next Right Pointers in Each Node

*Description:*

Given a binary tree:

*struct TreeLinkNode {*

*int value;*

*TreeLinkNode \*left, \*right, \*next;*

*TreeLinkNode (int x) : value(x), left(NULL), right(NULL), next(NULL) { }*

*};*

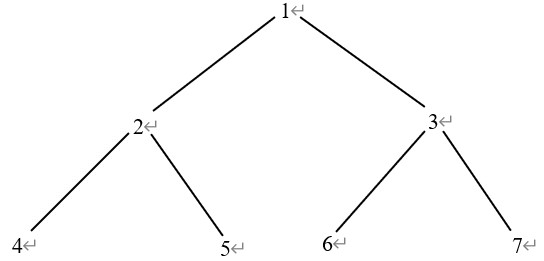
Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.

Initially, all next pointers are set to NULL.

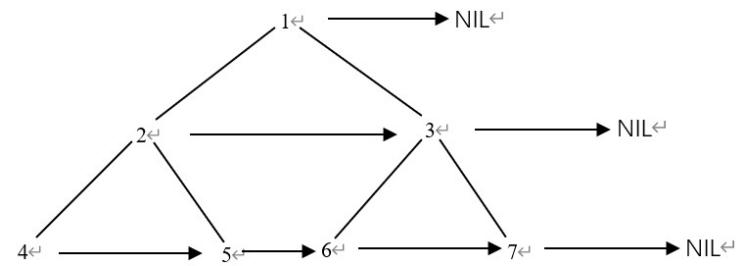
*Note:*

* You may only use constant extra space.
* You may assume that it is a *Perfect Binary Tree* (ie, all leaves are at the same level, and every parent has two children).

For example, Given the following *Perfect Binary Tree*,



After calling your function, the tree should look like:



*Analysis:*

In this example, we use the *Level Display Algorithm* to visit all nodes in the same level.

Use the data structure *Queue* as the intermediate data structure to save all nodes in each level and set the next pointer of the current node, for the tail node in the same level, set it to NULL.

*Code - Non-recursively:*

*Procedure:*

* Push the root node into queue.
* Set the first index as front() and the end index as end().
* Pop all nodes among the first to the end index.
* Set the next node of all nodes from the first to end index.
* Go to the next round and jump to the step 2.

class Solution {

public:

Node\* connect (Node\* root) {

if (root == NULL) return NULL;

// Push the root node from level 1 into Queue.

vector<Node \*> temp;

Node \* node;

Node \* next\_node;

temp.push\_back(root);

int size;

while (!temp.empty()) {

size = temp.size() - 1;

while (size >= 0) {

// Set the next node of all current nodes.

node = temp[0];

temp.erase(temp.begin());

size --;

if (size >= 0)

next\_node = temp[0];

else

next\_node = NULL;

node->next = next\_node;

// Push all nodes from same level into queue.

if (node->left != NULL)

temp.push\_back(node->left);

if (node->right != NULL)

temp.push\_back(node->right);

}

}

return root;

}

};

*Code - Recursively:*

struct TreeLinkNode {

int value;

TreeLinkNode \* left;

TreeLinkNode \* right;

TreeLinkNode \* next;

TreeLinkNode (int x) : value(x), left(NULL), right(NULL), next(NULL) { }

};

void ConnectNext(TreeLinkNode \* root) {

ConnectNext(root, NULL);

}

void ConnectNext(TreeLinkNode \* root, TreeLinkNode \* sibling) {

if (root == NULL)

return;

else

root->next = sibling;

ConnectNext(root->left, root->right);

if (sibling != NULL)

ConnectNext(root->right, sibling->left);

else

ConnectNext(root->right, NULL);

}