Populating Next Right Pointers in Each Node II

Follow up for problem "*Populating Next Right Pointers in Each Node*".

What if the given tree could be any binary tree? Would your previous solution still work?

Note:

- You may only use constant extra space.

For example,
Given the following binary tree,

```
         1
       /  \
      2    3
     / \    \
    4   5    7
```

After calling your function, the tree should look like:

```
         1 -> NULL
       /  \
      2 -> 3 -> NULL
     / \    \
    4-> 5 -> 7 -> NULL
```

Solution 1

Just share my iterative solution with O(1) space and O(n) Time complexity

```java
public class Solution {

    //based on level order traversal
    public void connect(TreeLinkNode root) {

        TreeLinkNode head = null; //head of the next level
        TreeLinkNode prev = null; //the leading node on the next level
        TreeLinkNode cur = root;  //current node of current level

        while (cur != null) {

            while (cur != null) { //iterate on the current level
                //left child
                if (cur.left != null) {
                    if (prev != null) {
                        prev.next = cur.left;
                    } else {
                        head = cur.left;
                    }
                    prev = cur.left;
                }
                //right child
                if (cur.right != null) {
                    if (prev != null) {
                        prev.next = cur.right;
                    } else {
                        head = cur.right;
                    }
                    prev = cur.right;
                }
                //move to next node
                cur = cur.next;
            }

            //move to next level
            cur = head;
            head = null;
            prev = null;
        }

    }
}
```

written by flashstone original link here

Solution 2

The idea is simple: level-order traversal. You can see the following code:

```
public class Solution {
    public void connect(TreeLinkNode root) {

        while(root != null){
            TreeLinkNode tempChild = new TreeLinkNode(0);
            TreeLinkNode currentChild = tempChild;
            while(root!=null){
                if(root.left != null) { currentChild.next = root.left; currentChild = currentChild.next;}
                if(root.right != null) { currentChild.next = root.right; currentChild = currentChild.next;}
                root = root.next;
            }
            root = tempChild.next;
        }
    }
}
```

written by davidtan1890 original link here

Solution 3

Thanks for liji94188 for adding the explanation:

It's a BFS traversal. now pointer is the current level traveler and head is the left most element at next level and the tail is the right most element at next level till now. We move now pointer at current level and populate the the next-link at its children level. (Here the gist is we can move now to its next because this relationship was already populated in the previous round).

```
void connect(TreeLinkNode *root) {
    TreeLinkNode *now, *tail, *head;

    now = root;
    head = tail = NULL;
    while(now)
    {
        if (now->left)
            if (tail) tail = tail->next =now->left;
            else head = tail = now->left;
        if (now->right)
            if (tail) tail = tail->next =now->right;
            else head = tail = now->right;
        if(!(now = now->next))
        {
            now = head;
            head = tail=NULL;
        }
    }
}
```

written by aileengw original link here