1. Populating Next Right Pointers in Each Node

Medium

1370130Add to ListShare

You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

struct Node {  
 int val;  
 Node \*left;  
 Node \*right;  
 Node \*next;  
}

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.

**Follow up:**

* You may only use constant extra space.
* Recursive approach is fine, you may assume implicit stack space does not count as extra space for this problem.

**Example 1:**



img

Input: root = [1,2,3,4,5,6,7]  
Output: [1,#,2,3,#,4,5,6,7,#]  
Explanation: Given the above perfect binary tree (Figure A), your function should populate each next pointer to point to its next right node, just like in Figure B. The serialized output is in level order as connected by the next pointers, with '#' signifying the end of each level.

**Solution**

层序遍历

/\*  
// Definition for a Node.  
class Node {  
public:  
 int val;  
 Node\* left;  
 Node\* right;  
 Node\* next;  
  
 Node() : val(0), left(NULL), right(NULL), next(NULL) {}  
  
 Node(int \_val) : val(\_val), left(NULL), right(NULL), next(NULL) {}  
  
 Node(int \_val, Node\* \_left, Node\* \_right, Node\* \_next)  
 : val(\_val), left(\_left), right(\_right), next(\_next) {}  
};  
\*/  
class Solution {  
public:  
 Node\* connect(Node\* root) {  
 if(root == NULL)return root;  
 queue<Node\*>q;  
 int cur = 1, next = 0;  
 q.push(root);  
 Node\* pre = new Node;  
 while(!q.empty()){  
 Node\* top = q.front();  
 q.pop();  
 if(top->left){  
 q.push(top->left);  
 next++;  
 }  
 if(top->right){  
 q.push(top->right);  
 next++;  
 }  
   
 pre->next = top;  
 if(cur != 1)pre = top;  
 else pre = new Node();  
   
 cur--;  
 if(cur == 0){  
 cur = next;  
 next = 0;  
 }  
 }  
 return root;  
 }  
};