



8、编写按层次顺序（同一层自左至右）遍历二叉树的算法。

void levelOrderTraversal(TreeNode\* root) {

if (!root) return;

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

int size = q.size();

for (int i = 0; i < size; ++i) {

auto node = q.front();

q.pop();

cout << node->val << " ";

if (node->left) q.push(node->left);

if (node->right) q.push(node->right);

}

cout << endl;

}

}

9、编写算法判别给定二叉树是否为完全二叉树。

bool isCompleteTree(Node\* root) {

if (root == nullptr) {

return true;

}

queue<Node\*> q;

q.push(root);

bool has\_missing = false;

while (!q.empty()) {

Node\* node = q.front();

q.pop();

if (has\_missing && (node->left != nullptr || node->right != nullptr)) {

return false;

}

if (node->left == nullptr && node->right != nullptr) {

return false;

}

if (node->left != nullptr) {

q.push(node->left);

}

if (node->right != nullptr) {

q.push(node->right);

}

if (node->left == nullptr || node->right == nullptr) {

has\_missing = true;

}

}

return true;

}