Lecture 34 Binary Trees

1. <https://leetcode.com/problems/binary-tree-right-side-view/>

class Solution {

public:

vector<int> rightSideView(TreeNode\* root) {

if(root==NULL)

{

return {};

}

queue<TreeNode\*> q;

q.push(root);

vector<int> ans;

while(!q.empty())

{

int n = q.size();

for(int i=1; i<=n; i++)

{

TreeNode \*f = q.front();

q.pop();

if(i==n)

{

ans.push\_back(f->val);

}

if(f->left)

{

q.push(f->left);

}

if(f->right)

{

q.push(f->right);

}

}

}

return ans;

}

};

1. <https://leetcode.com/problems/invert-binary-tree/>

class Solution {

public:

TreeNode\* invertTree(TreeNode\* root) {

if(root==NULL)

{

return NULL;

}

invertTree(root->left);

invertTree(root->right);

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

return root;

}

};

1. <https://leetcode.com/problems/subtree-of-another-tree/>

class Solution {

bool isEquals(TreeNode \*s, TreeNode \*t)

{

if(!s && !t)

{

return true;

}

if(!s || !t)

{

return false;

}

if(s->val!=t->val)

{

return false;

}

return isEquals(s->left, t->left) && isEquals(s->right, t->right);

}

public:

bool isSubtree(TreeNode\* s, TreeNode\* t) {

if(!s && !t)

{

return true;

}

if(!s || !t)

{

return false;

}

return isEquals(s, t) || isSubtree(s->left, t) || isSubtree(s->right, t);

}

};

1. <https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/>

class Solution {

public:

vector<vector<int>> zigzagLevelOrder(TreeNode\* root) {

if(root==NULL)

{

return {};

}

stack<TreeNode\*> s1;

stack<TreeNode\*> s2;

s1.push(root);

vector<vector<int>> res;

vector<int> tempAns;

while(!s1.empty() || !s2.empty())

{

while(!s1.empty())

{

TreeNode \*temp = s1.top();

s1.pop();

tempAns.push\_back(temp->val);

if(temp->left)

{

s2.push(temp->left);

}

if(temp->right)

{

s2.push(temp->right);

}

}

res.push\_back(tempAns);

tempAns.clear();

while(!s2.empty())

{

TreeNode \*temp = s2.top();

s2.pop();

tempAns.push\_back(temp->val);

if(temp->right)

{

s1.push(temp->right);

}

if(temp->left)

{

s1.push(temp->left);

}

}

if(tempAns.size() > 0)

{

res.push\_back(tempAns);

tempAns.clear();

}

}

return res;

}

};