Lecture 33 Binary Trees

1. Level Order Traversal BFS
2. #include <iostream>
3. #include<queue>
4. using namespace std;
5. class node{
6. public:
7. int data;
8. node\*left;
9. node\*right;
10. node(int d){
11. data = d;
12. left = NULL;
13. right = NULL;
14. }
15. };
16. node\* buildTree(){
17. int d;
18. cin>>d;
19. if(d==-1){
20. return NULL;
21. }
22. node \* root = new node(d);
23. root->left = buildTree();
24. root->right = buildTree();
25. return root;
26. }
27. void bfs(node \*root)
28. {
29. queue<node\*> q;
30. q.push(root);
31. while(!q.empty())
32. {
33. node \* f = q.front();
34. q.pop();
35. cout<<f->data<<", ";
36. if(f->left)
37. {
38. q.push(f->left);
39. }
40. if(f->right)
41. {
42. q.push(f->right);
43. }
44. }
45. return;
46. }
47. int count(node\* root)
48. {
49. if(root==NULL)
50. {
51. return 0;
52. }
53. return root->data + count(root->left) + count(root->right);
54. }
55. int main(){
56. node\* root = buildTree();
57. bfs(root);
58. cout<<endl;
59. cout<<count(root);
60. return 0;
61. }

2. <https://leetcode.com/problems/same-tree/>

class Solution {

public:

bool isSameTree(TreeNode\* p, TreeNode\* q) {

if(p==NULL && q==NULL)

{

return true;

}

else if(p==NULL || q==NULL)

{

return false;

}

else if(p->val!=q->val)

{

return false;

}

return isSameTree(p->left, q->left) && isSameTree(p->right, q->right);

}

};

3. <https://leetcode.com/problems/symmetric-tree/>

class Solution {

bool symm(TreeNode\* left, TreeNode\* right)

{

if(left==NULL && right==NULL)

{

return true;

}

if(left==NULL || right==NULL)

{

return false;

}

if(left->val!=right->val)

{

return false;

}

return symm(left->left, right->right) && symm(left->right, right->left);

}

public:

bool isSymmetric(TreeNode\* root) {

if(root==NULL)

{

return true;

}

return symm(root->left, root->right);

}

};