1. 问题描述与分析

1.在一棵二叉树中，分别需要找出值为x结点的孩子节点及确定值为x的结点数量。请问你会采用何种存储结构来实现这两个运算，说明理由，并构建算法完成验证。

2.抽象数据类型定义

typedef struct node {

//结构体定义

Belem data;

node\* lelem; //左孩子指针

node\* relem; //右孩子指针

}node, \* bnode;

3输入值范围

由于Belem定义为int ，所以输入的数值为整形

1. 问题解决思路

首先得创建好二叉树，我采用先序遍历的形式创建二叉树。确定值为x的结点数量，如果采用递归遍历的话不好计算，所以运用非递归的先序遍历来找到X的节点数量。

而要找到它的孩子节点，只要在以上的基础上加以判断输出就好了。

1. 算法及算法分析
2. 既有存放数据又有指针，结构体当然适合

2.

#include<iostream>

#include<stack>

#include<queue>

using namespace std;

//typedef char Status;

typedef int Belem;

typedef struct node {

//结构体定义

Belem data;

node\* lelem;

node\* relem;

}node, \* bnode;

void creat(bnode& p) { //先序遍历创建二叉树

int c;

cin >> c;

if (c == -1)

p = NULL;

else {

p = new node;

p->data = c;

creat(p->lelem);

creat(p->relem);

}

}

void Preoutput(bnode& p) { //先序遍历输出

if (p)

{

cout << p->data << " ";

Preoutput(p->lelem);

Preoutput(p->relem);

}

}

int PresortFindX(bnode& root,int m)

{ //运用非递归的先序遍历来找到X的节点数量

bnode p = root;

if (p == NULL)

return -1;

stack<bnode> s;

int count = 0;

while (!s.empty() || p)

{

while (p) {

//cout << p->data << " ";

if (p->data == m)

count++;

s.push(p);

p = p->lelem;

}

if (!s.empty()) {

p = s.top();

p = p->relem;

s.pop();

}

}

return count;

}

void Find\_X\_Child(bnode& root, int m)

{ //运用非递归的先序遍历来找到X的孩子节点并输出

bnode p = root;

if (p == NULL)

return;

stack<bnode> s;

//int count = 0;

while (!s.empty() || p)

{

while (p) {

//cout << p->data << " ";

if (p->data == m)

//count++;

{

if(p->lelem)

cout << p->lelem->data<<" ";

if(p->relem)

cout << p->relem->data << " ";

}

s.push(p);

p = p->lelem;

}

if (!s.empty()) {

p = s.top();

p = p->relem;

s.pop();

}

}

}

int main() {

node\* root;

cout << "请以先序遍历作为输入，子树若为空用‘-1’表示" << endl;

creat(root);

cout << "先序遍历结果" << endl;

Preoutput(root);

cout << endl;

cout << "请选择要查找的值" << endl;

int x;

cin >> x;

cout << "该值在二叉树节点数量为：" << endl;

cout<<PresortFindX(root, x);

cout << "该值孩子节点有：" << endl;

Find\_X\_Child(root, x);

}

3.

算法的时间复杂度和空间复杂度分析

递归算法空间复杂度（先序递归创建）：递归深度n\*每次递归所要的辅助空间，如果每次递归所需要的辅助空间为常数，则递归空间复杂度o（n）

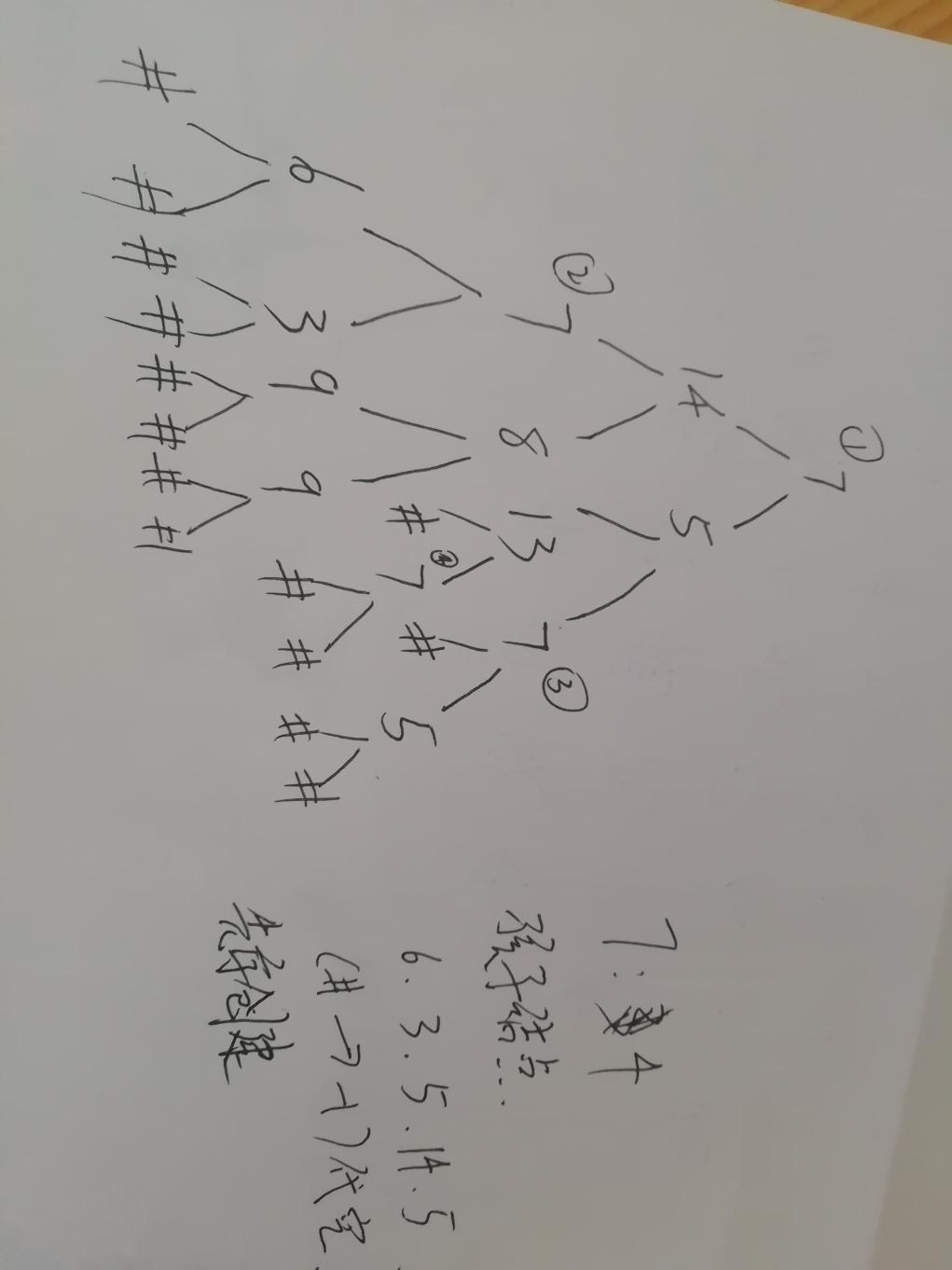
时间复杂度：O(2^n)

而非递归的遍历（找X，x的孩子） 由于每个节点都要进栈和出栈，所以****时间复杂度****为****O(N)****，同样****空间复杂度****也为****O(N)****，N为结点数

1. 测试

测试的二叉树应该考虑多种可能，所以二叉树的设计尤为重要。应该完整的囊括各个情况

设计测试用例：



运行截图：

