南开大学

Nankai University

南　开　大　学

网 络 空 间 安 全 学 院

数据结构实验报告

第四次实验作业

罗功成 1910487

年级：2019 级

专业：信息安全

指导教师：王玮

2021 年 11 月 25 日

摘要

在键值不重复，节点值可以为负数的非空二叉树，实现找到 p,q 节点公共祖先，遍历打印公共祖先的所有路径，并求最大路径和。

关键字：非空二叉树的链表实现，查找，比较，复杂度分析

目录

一、 实验流程

(一) 实验要求

1. 找到 p,q 节点公共祖先，一个节点可以是另一个节点的祖先.
2. 输出以上一问中以公共祖先为根节点的到所有叶节点的所有路径。
3. 输出以上一问中以公共祖先为根节点的子树的最大路径和。注意，路径不一定要通过该子树的根节点。

(二) 实验原理和思路

1. 首先要根据各个元素值不重复的特性，完成 p,q 元素的搜索功能 searchnode（只需要将遍历过程中的节点 value 和 p,q 元素值进行比较，相等时返回当前节点，即可得到 p,q 所在的节点）
2. 确定最近共同祖先：getlastcommonancestor 函数。设计思路是，从根节点出发，同时考虑它左孩子和右孩子。当左孩子和右孩子同时被确定为就是 p,q 对应的节点，那么当前的节点就是 p,q 的最近公共祖先；否则，将进行递归调用，即沿着现在的节点左孩子和右孩子分别向下找，此时把根节点设置为两个孩子，继续比较他们的孩子，直至找到 p,q 或者是遇到终结符号后停止。
3. 遍历打印以公共祖先为根的子树所有路径：用到的是 printpath 函数。将上一问找到的公共祖先节点作为函数中新的根节点，分别向左孩子和右孩子进行遍历，之后递归调用，将孩子节点作为新的节点，继续向下查找。当遇到的不是终结的节点时，把当前节点值保留在 path 数组中, 等遇到终止节点时，将会停止记录。
4. 计算以最近公共祖先为根节点的子树最大路径和：构建 solution 类。将上一问找到的公共祖先节点作为函数中新的根节点，指定任意一个在子树的节点作为开始，此时分别沿着 left,right 两个指针遍历路径，然后比较向左和向右的值，取最大，之后再递归调用，直至到空节点。最后，每个节点中得到的路径进行比较，将当前节点值和累加后的向左向右的路径较大值相加，从而得到最大路径。
5. 注意该树是按照先根方式（DLR）构建的，0 作为表示空节点的字符

(三) 实验核心代码

二叉树的链表实现和用到的函数

```
struct BinaryTreeNode
{
        BinaryTreeNode* left;
        BinaryTreeNode* right;
        int value;
};

//存储路径的数组
vector<int> path(10, 0);//用0表示为空

```

```
//创建二叉树
void createTree(BinaryTreeNode*& root)
{
        int val;
        cin >> val;
        if (val != 0)
        {
                root = new BinaryTreeNode;
                root->value = val;
                root->left = NULL;
                root->right = NULL;
                createTree(root->left);
                createTree(root->right);
        }
        else
                return;
}
//找公共祖先
BinaryTreeNode* GetLastCommonAncestor(BinaryTreeNode* root, BinaryTreeNode*
    node1, BinaryTreeNode* node2)
{
        if (root == NULL || node1 == NULL || node2 == NULL)
                return NULL;

        if (node1 == root || node2 == root)
                return root;

        BinaryTreeNode* cur = NULL;

        BinaryTreeNode* left_lca = GetLastCommonAncestor(root->left, node1,
            node2);
        BinaryTreeNode* right_lca = GetLastCommonAncestor(root->right, node1,
             node2);
        if (left_lca && right_lca)
                return root;
        if (left_lca == NULL)
                return right_lca;
        else
                return left_lca;
}

struct BinaryTreeNode* searchNode(struct BinaryTreeNode*node,int x) {
        struct BinaryTreeNode* endNode = NULL;
        if (node == NULL) {
                return NULL;
        }

        if (node->value == x) {
```

```
56                  endNode = node;
57          }
58          else {
59
60                  if (endNode == NULL) {
61                          endNode = searchNode(node->left ,x);
62                  }
63                  if (endNode == NULL) {
64                          endNode = searchNode(node->right ,x);
65                  }
66          }
67          return endNode;
68  }
69  //求最大路径和
70  class Solution {
71  public:
72          int dfs(BinaryTreeNode* root, int& res) {//dfs函数的作用是返回从当前
                结点往左或者往右走的单条最大路径
73                  if (root == NULL) { //如果扫到空结点，返回0（路径为0）
74                          return 0;
75                  }
76                  int left = max(dfs(root->left, res), 0);//left表示从左孩子往
                        左或者往右走的单条最大路径，再与0比较取较大的那个值
77                  int right = max(dfs(root->right, res), 0);//right同理
78                  res = max(res, root->value + left + right);//res是用来保存全
                        局中出现的最大路径的
79                                  //由于题中意思是可以不从根结点出发
80                                  //所以题意中的结果就是当前结点的值+左孩子往左
                                        或者往右走的单条最大路径+右孩子往左或者往
                                        右走的单条最大路径
81                  //要注意的是我们这边的left和right是经过处理的，与0比较过了，
                        所以不会出现当前结点的值+左右孩子的值之后小于当前结点的情
                        况
82                  return root->value + max(left, right);//dfs的作用是返回从当前
                        结点往左或者往右走的单条最大路径，所以要用自身+上左右孩子
                        中大的那条路径
83
84          }
85          int maxPathSum(BinaryTreeNode* root) {
86                  int res = INT_MIN;//用INT_MIN来解决树中全是负数的情况
87                  dfs(root, res);
88                  return res;
89          }
90  };
91
92
93  //递归实现输出从根节点到叶子结束的所有路径
94  void printPath(BinaryTreeNode* root, int pos)
```

```
{
        if (root == NULL)
                return;
        path[pos] = root->value;
        if (root->left == NULL && root->right == NULL)
        {
                auto it = path.begin();
                auto end = find(path.begin(), path.end(), 0);
                while (it != end)
                {
                        cout << *it++ << " ";
                }
                cout << endl;
        }
        else
        {
                if (root->left != NULL)
                {
                        printPath(root->left, pos + 1);
                }
                if (root->right != NULL)
                {
                        printPath(root->right, pos + 1);
                }
        }
        path[pos] = 0;
}
```

初始化和寻找最近公共祖先

```
    BinaryTreeNode* root = NULL;
        cout << "输入各节点, 空值用0来表示" << endl;
        createTree(root);
        cout << "请输入p,q节点" << endl;
        int p, q; cin >> p >> q;
        BinaryTreeNode* node1 = searchNode(root,p);
        BinaryTreeNode* node2 = searchNode(root,q);
        BinaryTreeNode* ancestor = GetLastCommonAncestor(root, node1, node2);
            //实现对p,q最近祖先的查找。
        cout << "p,q节点的祖先是: "<<ancestor->value << endl;
```

遍历打印以公共祖先为根节点的所有路径

```
    cout << "打印祖先的各条路径" << endl;
        printPath(ancestor, 0);//两个参数: 最近祖先节点和从0开始的路径下标
        //遍历打印祖先所有节点路径。
```

求出最大路径和

```
    int m;
        Solution max;//实例化一个类max求最大路径,max调用求最大路径的函数。
        max.dfs(ancestor, m);//两个参数：最近祖先节点，存放最大值的m
        cout << "最大路径和为"<<m<<endl;
```

（四） 实验结果与分析

注意，该二叉树是按照 DLR 的方式先序构造的二叉树。

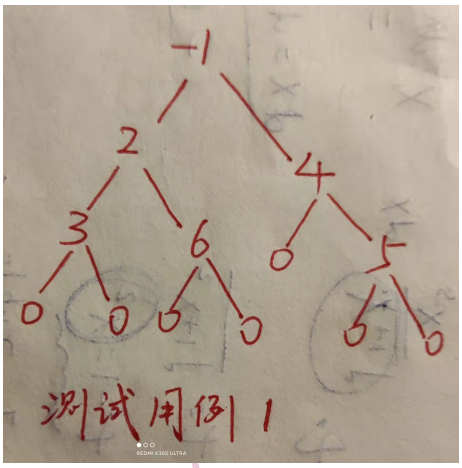

图 1：输入以下的二叉树

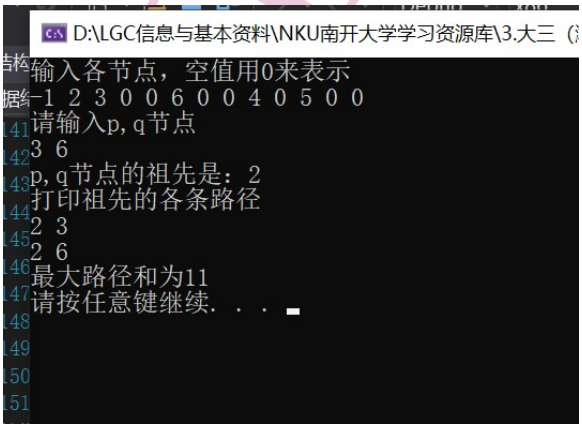

图 2：输出对应的公共祖先，所有路径与最大路径和

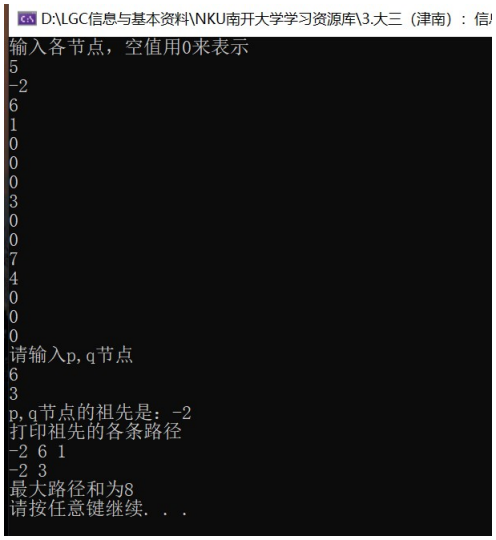

图 3: 助教指定的测试用例的表达结果

可见，以上代码可以满足本次实验的各项要求。

二、 实验探究问题解析

时间复杂度：O（n）

插入二叉树各个元素过程为 O（n），对二叉树的遍历查找和求最大值取决于是否是构造的是平衡二叉树，所以介于 O（logn）到 O（n）之间，综上为 O（n）

空间复杂度：O(n)

在计算最大值和公共祖先时，只是中间变量，不影响空间复杂度；

在遍历打印过程中，创建了一个可变大小的数组 path，用来分别保存各条路径。

SP=n*(addr(path)+sizeof(int)).

三、 总结和收获

1. 对二叉树的链表实现方法，二叉树的遍历，查找对应节点等相关知识得到了巩固和提高。

2. 对链表，指针，类的编写和实例化，函数调用等编程能力得到提高。