# 实验三：树型结构及其应用

### 一．实验目的

掌握树的存储结构和基本操作，重点巩固和体会二叉树的遍历及应用。

### 二．实验内容

(1) 题目1

按层次遍历建立二叉树，并输出该二叉树的前序遍历、 中序遍历和后序遍历的序列。

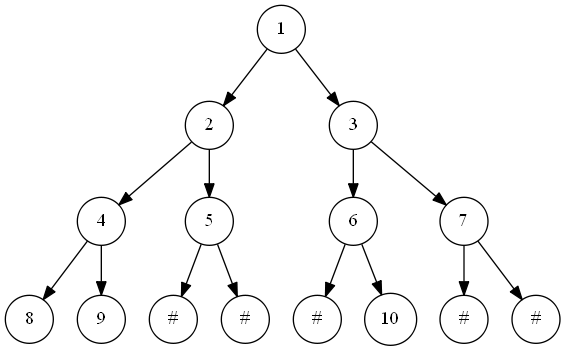


图1

以图1所示二叉树为例，第一行输入为15，表示节点的个数，第二行输入为1 2 3 4 5 6 7 8 9 # # # 10 # #，其中，#表示空，表示层次遍历序列。该案例应输出：

前序遍历：1 2 4 8 9 5 3 6 10 7

中序遍历：8 4 9 2 5 1 6 10 3 7

后序遍历：8 9 4 5 2 10 6 7 3 1

同一序列输出之间用空格隔开，不同序列换行输出

注：文件读写已经为你写好，序列1 2 3 4 5 6 7 8 9 # # # 10 # # 将被转换成int数组1 2 3 4 5 6 7 8 9 -1 -1 -1 10 -1 -1，其中，-1表示空。

注：只输入一个 # 时，需要返回空二叉树！

**提示**：1. 可以参考“树-第二部分”PPT 第13页内容！2. 可以使用队列来实现层序遍历构建二叉树，也可以通过计算索引来完成构建二叉树。

(2) 题目2

给定一棵二叉树，**路径**定义为从树的根节点到叶子结点的任意路径，求取该二叉树的最大路径和，**路径和**定义为一条路径中各节点的权值之和。

**注**：只输入一个#时，需要返回0！

以图1为例，一共有5条路径，分别为：

1. 1 2 4 8
2. 1 2 4 9
3. 1 2 5
4. 1 3 6 10
5. 1 3 7

最大路径和为20，为第四条路径1+3+6+10=20。

(3) 题目3

给定一棵二叉树，求取该二叉树的所有左子叶权重之和，左子叶被定义为二叉树叶子结点中属于左子树的节点。

**注**：只输入一个#时，需要返回0！

以图1为例，左孩子分别有2 4 6 8，但是2 4 6 都不是叶子结点，只有8才是左子叶，所以该树的左子叶权重之和为8。

(4) 题目4

给定求取该树的镜像，即翻转该二叉树，如图1的结果如图2所示。

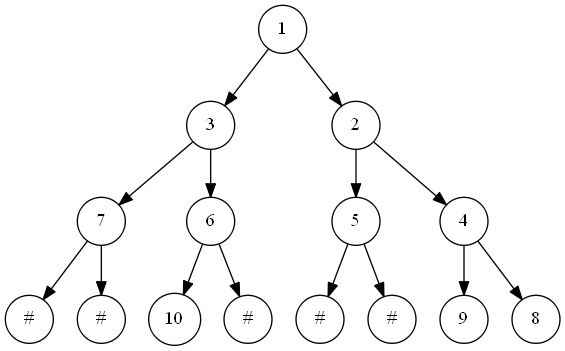


图2

并输出翻转后二叉树的中序遍历。

**注**：在实验中会对题目1中构造的二叉树进行测试！

### 实验要求

1. 提供模板main.c，也可以自己编写程序。
2. 根据实验结果撰写实验报告。

### 函数说明

1. 队列封装（已经提供）

说明：队列使用链表的头删尾插法，有一个dummyHead

ListNodePtr createListNode(TreeNodePtr, ListNodePtr next): 创建链表节点，即队列元素。

QueuePtr InitQueue()：队列初始化。

void EnQueue(QueuePtr queue, TreeNodePtr node)：在队列的尾部添加一个元素的副本

void DeQueue(QueuePtr queue)：删除队列中的第一个元素

bool QueueEmpty(QueuePtr queue)：判断队列中是否有元素，如果为空返回true。

TreeNodePtr GetHead(QueuePtr queue)：返回队列中第一个元素的引用

1. 二叉树（需要自己完成）

TreeNodePtr createTreeNode(int val, TreeNodePtr left, TreeNodePtr right)：创建二叉树结点

TreeNodePtr createTreeWithLevelOrder(int \*data, int size)：通过层次遍历来构建二叉树，并返回二叉树的头结点

void preOrderTraverse(TreeNodePtr root)：打印输出二叉树前序遍历序列

void inOrderTraverse(TreeNodePtr root)：打印输出二叉树中序遍历序列

void postOrderTraverse(TreeNodePtr root)：打印输出二叉树后序遍历序列

int maxPathSum(TreeNodePtr root, int sum)：求取二叉树的最大路径和

int sumOfLeftLeaves(TreeNodePtr root)：求取二叉树的所有左子叶权重之和

TreeNodePtr invertTree(TreeNodePtr root)：求取二叉树的镜像

1. Io/其他（已经提供）

int max(int a, int b)：返回二者间最大值

void getDigits(char \*buff, int \*data)：将二叉树层次遍历的char序列转换成可以计算int序列

void createDotFile(char \*filename, TreeNodePtr root, int MaxSize)：创建dot文件用于graphviz可视化，需要安装该软件，下载地址：<https://graphviz.org/>

void plot(TreeNodePtr tree\_root, int i, int size, char \*name)：调用系统命令使用.dot文件绘制二叉树.png：命令如下：dot -Tpng xx.dot -o xx.png，通过配置变量use\_graphviz来选择是否进行绘制。

### Graphviz说明（不计分）

1. 为了实现二叉树的可视化需要下载graphviz软件，下载地址：<https://graphviz.org/>。
2. 进入网站后选择对应的操作系统版本进行安装。

图形用户界面, 文本, 应用程序

描述已自动生成

1. 一般图片绘制

使用语法：

diagraph G{}：定义图纸

nodeId[attr1=val1, attri2=val2]：定义节点及相关属性

node1->node2[attr1=val1]：定义有向连接及相关属性

更多语法请参考：<http://www.graphviz.org/doc/info/>

示例代码：

digraph g {

a->b[color=cyan];

b->c[style=dotted];

c->a[color=indigo];

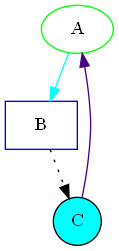
a[shape=oval, label="A", color=green];

b[shape=rect, label="B", color=blue];

c[shape=circle, label="C", style=filled, fillcolor=cyan];

}

结果图如下：



1. 二叉树的绘制

对于二叉树由于有左右子叶的存在，所以我们有时需要绘制一些节点，然后设置其属性为隐藏。比如，有一个节点只有左节点，没有右节点，那么，我们绘制的时候，可以同时绘制左右节点，然后设置有节点的属性为隐藏。

二叉树示例的代码及结果图：

digraph G {

124 [shape=circle, label="1"];

124->126;

\_n2 [shape=circle, label="#", style=invis];

124->\_n2 [style=invis, weight=10];

126 [shape=circle, label="2"];

126->128;

\_n4 [shape=circle, label="#", style=invis];

126->\_n4 [style=invis, weight=10];

126->129;

128 [shape=circle, label="3"];

\_n6 [shape=circle, label="#", style=invis];

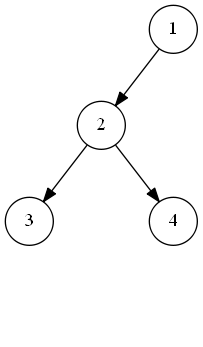
128->\_n6 [style=invis, weight=10];

129 [shape=circle, label="4"];

\_n8 [shape=circle, label="#", style=invis];

129->\_n8 [style=invis, weight=10];

}



1. 图片生成

将代码保存为 tree.dot 文件，使用cmd或其他终端（terminal）执行

dot -Tpng tree.dot -o tree.png

以生成 .png 图片