

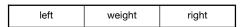
本小节内容

2014年41题真题讲解

2014 年 41 题真题讲解

2014年

41. (13分) 二叉树的带权路径长度 (WPL) 是二叉树中所有叶结点的带权路径长度 之和。给定一棵二叉树 T, 采用二叉链表存储, 结点结构如下:



其中叶结点的 weight 域保存该结点的非负权值。设 root 为指向 T 的根结点的指针,请设计 求 T 的 WPL 的算法,要求:

- 1) 给出算法的基本设计思想。
- 2) 使用 C 或 C++语言, 给出二叉树结点的数据类型定义。
- 3) 根据设计思想,采用C或C++语言描述算法,关键之处给出注释。

答案解析:

树的带权路径长度(Weighted Path Length of Tree, 简记为 WPL

考查二叉树的带权路径长度,二叉树的带权路径长度为**每个叶子结点的深度与权值之积的总和**,可以使用先序遍历或层次遍历解决问题。

- 1) 算法的基本设计思想:
- ①基于先序递归遍历的算法思想是用一个 static 变量记录 wpl, 把每个结点的深度作为递归函数的一个参数传递, 算法步骤如下:

若该结点是叶子结点, 那么变量 wpl 加上该结点的深度与权值之积;

若该结点非叶子结点,那么若左子树不为空,对左子树调用递归算法,若右子树不为空,对右子树调用递归算法,深度参数均为本结点的深度参数加一;

最后返回计算出的 wpl 即可。

②基于层次遍历的算法思想是使用队列进行层次遍历, 并记录当前的层数,

当遍历到叶子结点时, 累计 wpl;

当遍历到非叶子结点时对该结点的把该结点的子树加入队列;

当某结点为该层的最后一个结点时, 层数自增 1;

队列空时遍历结束,返回 wpl (层序遍历代码比较复杂,因此我们用先序遍历)

2) 二叉树结点的数据类型定义如下

typedef int BiElemType:

typedef struct BiTNode{

BiElemType weight;

struct BiTNode *lchild;

struct BiTNode *rchild;

}BiTNode,*BiTree;

```
3) 代码实现如下:
#include <stdio.h>
#include <stdlib.h>
typedef int BiElemType;
typedef struct BiTNode{
    BiElemType weight;//c 就是书籍上的 data
    struct BiTNode *lchild;
    struct BiTNode *rchild;
}BiTNode,*BiTree;
typedef struct tag{
    BiTree p;//树的某一个结点的地址值
    struct tag *pnext;
}tag_t,*ptag_t;
int wp1_PreOrder(BiTree root, int deep) {
    //wp1 称为静态局部变量,类似于全局变量,只会初始化1次
    static int wp1 = 0;
    if (root->lchild == NULL && root->rchild == NULL)
         //若为叶子结点, 累积 wp1
         wp1 += deep * root->weight;
    if (root->lchild != NULL)
         wp1_PreOrder(root->lchild, deep + 1);
    if (root->rchild != NULL)
         wp1_PreOrder(root->rchild, deep + 1);
    return wp1;
int WPL(BiTree root) {
    return wp1_PreOrder(root, 0);
//对于初试,只需要编写 WPL 和 wp1_PreOrder 两个子函数即可得满分,不需要
写下面的 main 函数
//《王道 C 督学营》课程
int main()
    BiTree pnew;
    int i,j,pos;
    char c;
    BiTree tree=NULL;//树根
    ptag_t phead=NULL,ptail=NULL,listpnew=NULL,pcur=NULL;//phead 就是队列头, ptail 就是队列尾
    //abcdefghij
```

长注微信公众号:王道在线 王道论坛网址: www.cskaoyan.com

```
while(scanf("%c",&c))
    if(c=='\n')
    {
        break;
    pnew=(BiTree)calloc(1,sizeof(BiTNode));//calloc 申请空间并对空间进行初始化,赋值为 0
    pnew->weight=c;//数据放进去
    listpnew=(ptag_t)calloc(1,sizeof(tag_t));//给队列结点申请空间
    listpnew->p=pnew;
    if(NULL==tree)
        tree=pnew;//树的根
        phead=listpnew;//队列头
        ptail=listpnew;//队列尾
        pcur=listpnew;
        continue;
    }else{
        ptail->pnext=listpnew;//新结点放入链表,通过尾插法
        ptail=listpnew;//ptail 指向队列尾部
    }//pcur 始终指向要插入的结点的位置
    if(NULL==pcur->p->lchild)//如何把新结点放入树
        pcur->p->lchild=pnew;//把新结点放到要插入结点的左边
    }else if(NULL==pcur->p->rchild)
        pcur->p->rchild=pnew;//把新结点放到要插入结点的右边
        pcur=pcur->pnext;//左右都放了结点后,pcur 指向队列的下一个
printf("%d\n", WPL(tree));
return 0;
```