《数据结构与算法》实验报告

实验四：二叉树和数组

**教 师：潘晔**

**学 生：梁书恺**

**学 号：2022040906023**

**时 间：10.9**

**地 点：科B119**

**一、ex4-1**

1. **问题分析：**
2. **题意理解**

传入一个字符串，用大写字母表示结点，用\*表示空指针，以先序框架建立起二叉树，再以中序、后序遍历二叉树，最后求二叉树的深度和宽度

1. **数据结构设计**

typedef struct BiNode

{

char data;

BiNode \*lchild, \*rchild;

} BiNode;

1. **关键算法思路**

// 先序创建二叉树

BiNode \*CreateBiTree(BiNode \*T, char \*str)

{

static int i = 0;

char ch = str[i++];

if (ch == '\*')

T = NULL;

else

{

T = (BiNode \*)malloc(sizeof(BiNode));

if (T == NULL)

exit(2);

T->data = ch;

T->lchild = CreateBiTree(T->lchild, str);

T->rchild = CreateBiTree(T->rchild, str);

}

return T;

}

1. **健壮性设计**

通过调用统一出错处理函数ErrorHandle打印出错信息并跳出程序。

1. **性能分析**

时间复杂度：O(n);

空间复杂度：O(n);

1. **测试数据与运行结果截图：**

传入字符串："ABD\*\*E\*\*CF\*\*\*"

输出：

中序遍历二叉树：D B E A F C

后序遍历二叉树：D E B F C A

二叉树的深度为：3

二叉树的宽度为：3

传入字符串："ABD\*\*E\*\*CF\*1\*"

输出：

输入不合法！

1. **上机时遇到的问题**
2. **问题现象：** 创建二叉树时异常 **原因：**先序框架递归逻辑有误 **解决办法：**修改先序创建时的递归逻辑
3. **问题现象：** 二叉树的宽度输出错误 **原因：**不会求二叉树宽度 **解决办法：**网上搜索算法求二叉树宽度
4. **程序代码**

ex1.c

ex1.h

**二、ex1-2**

1. **问题分析：**
2. **题意理解**

定义两个三元组构成的稀疏矩阵A，B，将两个矩阵相加，返回其结果。

1. **数据结构设计**

typedef struct Triple

{

unsigned int row, col;

int data;

} Triple;

typedef struct Matrix

{

Triple data[10];

unsigned int row, col, num;

} Matrix;

1. **关键算法思路**

// 三元组稀疏矩阵相加

while (i < A->num && j < B->num)

{

if (A->data[i].row < B->data[j].row)

{

C->data[k].row = A->data[i].row;

C->data[k].col = A->data[i].col;

C->data[k].data = A->data[i].data;

C->num++;

i++;

k++;

}

else if (A->data[i].row > B->data[j].row)

{

…

}

else

{

if (A->data[i].col < B->data[j].col)

{

…

}

else if (A->data[i].col > B->data[j].col)

{

…

}

else

{

…

}

}

}

if (i < A->num)

{

while (i < A->num)

{

…

}

}

if (j < B->num)

{

while (j < B->num)

{

…

}

}

1. **健壮性设计**

若矩阵维数不同，则给出提示，并取较大的维数作为新矩阵的维数。

1. **性能分析**

时间复杂度：O(n);

空间复杂度：O(n);

1. **测试数据与运行结果截图：**

Matrix A = {{{0, 0, 1}, {1, 1, 3}, {1, 2, 5}, {2, 3, 4}}, 3, 3, 4};

Matrix B = {{{0, 0, 2}, {1, 1, 4}, {2, 1, 3}, {2, 2, 5}}, 3, 3, 4};

输出：

0 0 3

1 1 7

1 2 5

2 1 3

2 2 5

2 3 4

Matrix A = {{{0, 0, 1}, {1, 1, 3}, {1, 2, 5}, {2, 3, 4}}, 3, 3, 4};

Matrix B = {{{0, 0, 2}, {1, 1, 4}, {2, 1, 3}, {2, 2, 5}, {3, 1, 7}}, 3, 4, 5};

输出：

注意：矩阵大小不同！

0 0 3

1 1 7

1 2 5

2 1 3

2 2 5

2 3 4

3 1 7

1. **上机时遇到的问题**
2. **问题现象：** 打印时不显示元素 **原因：**在进行加法时忘记对新矩阵C的数量++ **解决办法：**在加和时对新数组的num++
3. **问题现象：** 最后几个元素没有加 **原因：**在主加法循环时，会有一个矩阵的元素没有被完全遍历 **解决办法：**在之后遍历剩余的元素
4. **程序代码**

ex2.c

ex2.h

**三、ex1-3**

1. **问题分析：**
2. **题意理解**

传入一组整数，建立哈夫曼树，并输出权值对应的二进制编码。

1. **数据结构设计**

typedef struct

{

unsigned int weight;

int parent, lchild, rchild;

} HuffmanTree;

1. **关键算法思路**

HuffmanTree \*CreateHuffmanTree(int w[], int n)

{

// 初始化哈夫曼树

HuffmanTree \*T = (HuffmanTree \*)malloc(sizeof(HuffmanTree) \* (2 \* n - 1));

for (int i = 0; i < 2 \* n - 1; i++)

{

T[i].parent = -1;

T[i].lchild = -1;

T[i].rchild = -1;

}

// 创建哈夫曼树

for (int i = n; i < 2 \* n - 1; i++)

{

int min1 = 99999, min2 = 99999;

int x1 = -1, x2 = -1;

for (int j = 0; j < i; j++)

{

if (T[j].parent == -1)

{

if (w[j] < min1)

{

min2 = min1;

x2 = x1;

min1 = w[j];

x1 = j;

}

else if (w[j] < min2)

{

min2 = w[j];

x2 = j;

}

}

}

T[i].weight = min1 + min2;

T[i].lchild = x1;

T[i].rchild = x2;

T[x1].parent = i;

T[x2].parent = i;

}

return T;

}

1. **健壮性设计**

若内存分配失败，则提示并退出程序。

1. **性能分析**

时间复杂度：O(1);

空间复杂度：O(1);

1. **测试数据与运行结果截图：**

输入：5, 29, 7, 8, 14, 23, 3, 11

输出：

1

00

0010

1010

00

1

0

110

1. **上机时遇到的问题**

**1. 问题现象：** 数组溢出 **原因：**在定义时使用无符号型，在赋值时赋入负数 **解决办法：**定义为有符号型数

1. **程序代码**

ex3.c

ex3.h

**小结**

本章主要学习二叉树和数组。二叉树的各种操作逻辑比较复杂，在遍历时，要注意递归逻辑和退出条件。在三元组矩阵相加时，要将原矩阵中的每一个元素都加到新矩阵中，做到不重不漏。