武汉纺织大学外经贸学院

工程学院

数据结构课程设计报告

2022 ~2023学年第1学期

|  |  |
| --- | --- |
|  |  |
| |  |  | | --- | --- | | **题目** |  | | **班级** |  |  |  |  |  | | --- | --- | --- | |  | 学号 | 姓名 | | **组长** |  |  | | **成员1** |  |  | | **成员2** |  |  | | **成员3** |  |  | | |

**提交课程报告要求**

提交的成果的内容必须由以下4个部分组成：

1．源程序：学生按照课程设计的具体要求所开发的所有源程序（应该放到一个文件夹中）；

2．程序的说明文件：（保存在.txt中）在说明文档中应该写明上交程序所在的目录，上交程序的主程序文件名，如果需要安装，要有程序的安装使用说明；

3．课程设计报告：（以word或pdf 文档提交，文件名要求按照 人数+组长姓名"命名，如文件名为"**3**刘备" 、“**1**马超”）按照本文档格式逐项（内容可扩展）填写，具体每项填写要求见正文。报告的排版要求：排版要求：行间距：1.25倍；纸型：A4；页眉，页脚，页边距：采用默认值；页码：Times New Roman，小五，靠右。

目录

[1. 题目与要求 4](#_Toc24554759)

[1.1. 问题描述 4](#_Toc24554760)

[1.2. 本系统涉及的知识点 4](#_Toc24554761)

[1.3. 功能要求 4](#_Toc24554762)

[2. 功能设计 4](#_Toc24554763)

[2.1. 数据结构定义 4](#_Toc24554764)

[2.2. 模块图 4](#_Toc24554765)

[3. 功能代码 4](#_Toc24554766)

[4. 调试与测试 5](#_Toc24554767)

[4.1. 调试分析 5](#_Toc24554768)

[4.2. 用户手册 5](#_Toc24554769)

[4.3. 测试过程 5](#_Toc24554770)

[5. 总结 5](#_Toc24554771)

[6. 参考文献 5](#_Toc24554772)

[7. 附录 6](#_Toc24554773)

[8. 项目分工 7](#_Toc24554774)

# 题目与要求

**赫夫曼树的构造及赫夫曼编码**

## 问题描述

赫夫曼树的构造：根据输入的权值动态生成赫夫曼树，使得权值大的靠近根目录，权值小的远离根目录

赫夫曼编码：从根节点开始到叶子结点，根据路径获取赫夫曼编码

## 本系统涉及的知识点

采用二叉链表，通过数据域和左右孩子结点构成

## 功能要求

创建树的结点：创建一个树的结点并将变量赋值。

创建赫夫曼树：通过权值数组构造一个赫夫曼树结构。

先序遍历：通过递归依次遍历根结点，左孩子结点，右孩子结点。

中序遍历：通过递归依次遍历左孩子结点，根结点，右孩子结点。

后序遍历：通过递归依次遍历左孩子结点，右孩子结点根结点，。

获得赫夫曼编码：通过根节点到叶子结点的路径，确定唯一的赫夫曼编码

# 功能设计

## 数据结构定义

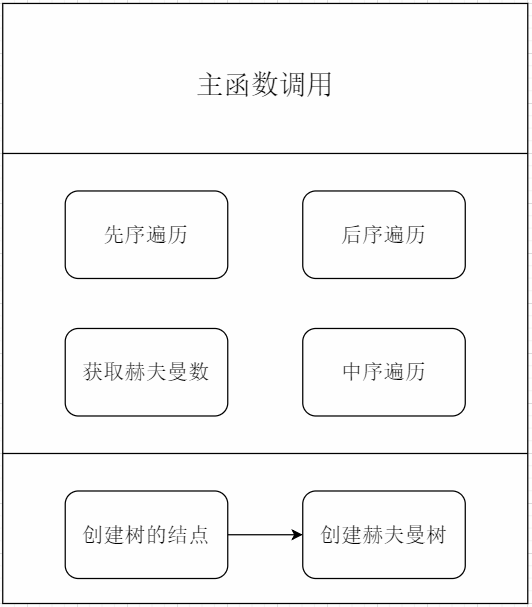
typedef struct BinaryTree {

int data;

BinaryTree \*Lchild, \*Rchild;

}BinaryTree;

## 模块图



# 算法设计

构造赫夫曼树：使用冒泡循环降序排序，每次都取最后两个数即最小的两个数，构成新的子树根结点为这两个数的和，循环直至遍历完成。

遍历：使用递归调用，如果左右孩子结点不为空就继续递归即到达叶子结点，直至遍历完整个树，不同的遍历只是递归位置不同。

输出赫夫曼数：递归调用，每次将输入的数据移位处理，如果不是叶子结点则进入递归，其中左孩子携带参数0，右孩子携带参数1，如果是叶子结点就输入数据。

# 调试与测试

## 调试分析

分析：完成的基本要求，但并没有做到数据的转换和逆转换。在构造赫夫曼树中的时间复杂度可以继续降低，如使用时间复杂度更低的排序算法

复杂度：核心的算法均是递归算法所以时间复杂度为O(n)，而构造赫夫曼树中使用的冒泡循环和while循环所以时间复杂度为O(n^3)。

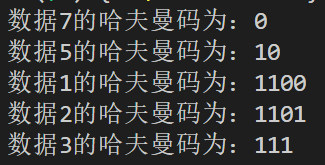
## 用户手册

首先需要一个int类型的权值数组，还要计算数组的长度

将数据输入到CreateHuffmanTree()中获取二叉树结构的赫夫曼树

其他功能函数只需要传入赫夫曼树结构即可

## 测试过程



# 总结

掌握了带权路径的计算方法、构造赫夫曼树的算法、二叉树链表存储方法，理解算法的时间复杂度。其中递归的循环中如何携带输入而不覆盖上一层查询了很多资料，最后用C++可以设置参数默认值的特性解决了第一次输入的问题。下一次改进应该在二叉树中添加一个权值参数，将权值和数据分开存储，可以实现更准确的赫夫曼数，也可以做到翻译和逆翻译。

# 参考文献

https://www.bilibili.com/video/BV1JZ4y1y7qB/?spm\_id\_from=333.337.search-card.all.click

https://blog.csdn.net/chengqiuming/article/details/115052522

# 附录

#include<iostream>

#include<math.h>

using namespace std;

typedef struct BinaryTree {

int data;

BinaryTree \*Lchild, \*Rchild;

}BinaryTree;

BinaryTree\* CreateTreeNode(int data);

BinaryTree\* CreateHuffmanTree(int data[], int size);

void ShowTreeMLR(BinaryTree\* root);

void ShowTreeLMR(BinaryTree\* root);

void ShowTreeLRM(BinaryTree\* root);

void GetLeafNumber(BinaryTree\* root, int number = 0);

int main() {

int data[] = {1, 5, 3, 2, 7};

int size = sizeof(data) / sizeof(data[0]);

BinaryTree\* HuffmanTree = CreateHuffmanTree(data, size);

//ShowTreeMLR(HuffmanTree);

GetLeafNumber(HuffmanTree);

return 0;

}

BinaryTree\* CreateTreeNode(int data) {

BinaryTree\* tree = new BinaryTree;

tree->data = data;

tree->Lchild = NULL;

tree->Rchild = NULL;

return tree;

}

BinaryTree\* CreateHuffmanTree(int data[], int size) {

int i, j;

BinaryTree\* HuffmanNode[size];

BinaryTree\* temp = CreateTreeNode(-1);

BinaryTree exchangeTemp;

for (i = 0; i < size; i++) {

HuffmanNode[i] = CreateTreeNode(data[i]);

}

while (size != 1) {

for (i = 0; i < size; i++) {

for (j = 0; j < size - 1 - i; j++) {

if (HuffmanNode[j]->data < HuffmanNode[j+1]->data) {

exchangeTemp = \*HuffmanNode[j];

\*HuffmanNode[j] = \*HuffmanNode[j+1];

\*HuffmanNode[j+1] = exchangeTemp;

}

}

}

temp = CreateTreeNode(HuffmanNode[size-1]->data + HuffmanNode[size-2]->data);

temp->Lchild = HuffmanNode[size-1];

temp->Rchild = HuffmanNode[size-2];

HuffmanNode[size-2] = temp;

size--;

}

return HuffmanNode[0];

}

void ShowTreeMLR(BinaryTree\* root) {

if (root != NULL) {

cout << root->data;

cout << '|';

ShowTreeMLR(root->Lchild);

ShowTreeMLR(root->Rchild);

}

}

void ShowTreeLMR(BinaryTree\* root) {

if (root != NULL) {

ShowTreeLMR(root->Lchild);

cout << root->data;

cout << '|';

ShowTreeLMR(root->Rchild);

}

}

void ShowTreeLRM(BinaryTree\* root) {

if (root != NULL) {

ShowTreeLRM(root->Lchild);

ShowTreeLRM(root->Rchild);

cout << root->data;

cout << '|';

}

}

void GetLeafNumber(BinaryTree\* root, int number) {

if (root != NULL) {

if (root->Lchild == NULL && root->Rchild == NULL) {

cout << "数据" << root->data << "的哈夫曼码为：";

cout << number << endl;

} else {

GetLeafNumber(root->Lchild, number\*10 + 0);

GetLeafNumber(root->Rchild, number\*10 + 1);

}

}

}

# 项目分工（每组不超过4人）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **成员姓名** | **承担任务** | **完成情况** | **贡献比例（%）** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |

小组成员签名：

日 期：