

# 河海大学

## 算法与数据结构课程设计报告



课设题目：图书馆管理系统、哈夫曼编码

小组组长：瞿晟

小组成员：王鑫，夏洋，刘雅婷，向景荣

指导老师：刘颜君

报告日期：2019.6.10

# 目录

一. 需求分析.....	3
1.1 开发背景及意义.....	3
1.2 设计题目与要求.....	3
二. 概要设计.....	4
2.1 图书管理系统.....	4
2.2 哈夫曼编码.....	5
三. 详细设计.....	6
3.1 图书管理系统.....	6
3.1.1 存储结构.....	6
3.1.2 采编入库.....	6
3.1.3 添加学生信息.....	8
3.1.4 按书号查询图书信息.....	9
3.1.5 按书名查询图书信息.....	10
3.1.6 按作者查询图书信息.....	11
3.1.7 显示学生信息.....	12
3.1.8 借阅书籍.....	13
3.1.9 归还书籍.....	16
3.2 哈夫曼编码.....	19
3.2.1 存储结构.....	19
3.2.2 计算字符的权值.....	20
3.2.3 初始化哈夫曼树.....	22
3.2.4 构造哈夫曼树.....	24
3.2.5 将一个字符反转.....	25
3.2.6 哈夫曼编码.....	25
3.2.7 哈夫曼译码.....	27
3.2.8 用户输入字符.....	29
3.2.9 用户输入解码字符.....	30
3.2.10 计算压缩比.....	30
四. 调试分析.....	31
4.1 图书管理系统.....	31
4.1.1 欢迎界面.....	31
4.1.2 采编入库.....	31
4.1.3 添加学生信息.....	32
4.1.4 查询图书信息.....	33
4.1.5 显示学生信息.....	36
4.1.6 借阅书籍.....	37
4.1.7 归还书籍.....	38
4.1.8 退出.....	40
4.1.9 程序分析.....	40
4.2 哈夫曼编码.....	40
4.2.1 欢迎界面.....	40
4.2.2 计算字符的权值.....	41

4.2.3 哈夫曼编码.....	42
4.2.4 用户输入字符.....	43
4.2.5 用户输入解码字符.....	44
4.2.6 退出.....	45
4.2.7 程序分析.....	45
<b>五. 课程设计总结.....</b>	<b>45</b>
5.1 项目分工.....	45
5.2 心得体会.....	46

## 一. 需求分析

### 1.1 系统开发背景及意义

图书管理系统：

图书管理作为计算机应用的一个分支，有着手工管理无法比拟的优点，如检索迅速、查找方便、可靠性高、存储量大、保密性好、寿命长、成本低等。这些有点能够极大地提高图书管理地效率。因此，开发一套能够为用户提供充足信息和快捷地查询手段地图书管理系统，将是非常必要地，也是十分及时的。

图书管理系统需要满足查找图书、借书、还书等操作，以及为图书管理系统中添加学生信息的操作。

哈夫曼编码：

在当今信息爆炸时代，如何采用有效的数据压缩技术来节省数据文件的存储空间和计算机网络的传送时间已越来越引起人们的重视。哈夫曼编码正是一种应用广泛且非常有效的数据压缩技术。

哈夫曼编码的应用很广泛，利用哈夫曼树求得的用于通信的二进制编码称为哈夫曼编码。树中从根到每个叶子都有一条路径，对路径上的各分支约定：指向左子树的分支表示“0”码，指向右子树的分支表示“1”码，取每个路径上的“0”或“1”的序列作为各个对应的字符的编码，这就是哈夫曼编码。

通常我们把数据压缩的过程称为编码，解压缩的过程称为解码。电报通信是传递文字的二进制形式的字符串。但在信息传递时，总希望总长度尽可能最短，即采用最短码。

### 1.2 设计题目与要求

图书管理基本业务模拟：

- 1) 书的登记内容包括书号、书名、著作者、现存量和库存量；
- 2) 建立索引表（线性表）以提高查找效率；

3) 主要功能如下:

a) 采编入库: 新购一种书, 确定书号后, 登记到图书帐目表中, 如果表中已有, 则只将库存量增加;

b) 借阅: 如果一种书的现存量大于 0, 则借出一本, 登记借阅者的书证号和归还期限, 改变现存量;

c) 归还: 注销对借阅者的登记, 改变该书的现存量。

输出形式: 能按书号、书名、著作者查找库存的书籍信息

能按学生的借书证号显示学生信息和借阅信息

书籍入库

借书功能实现

还书功能实现

采用哈夫曼编码进行文件压缩:

给定一个文本文件, 统计其中字符使用频率, 建立哈夫曼树, 设计哈夫曼编码与译码方案, 计算压缩比。

## 二. 概要设计

### 2.1 图书管理系统

分为 7 个模块:

1) 添加图书信息

输入图书的一些信息: 书号、书名、作者、存量、总量, 将图书添加进书库, 若有相同书号的图书则增加其存量和总量, 否则直接添加, 并以 txt 文件形式保存。

2) 添加学生信息

输入学生的一些信息: 学号 (借书证号)、学号、性别、院系, 默认为未借书, 将学生信息添加进学生信息库, 并以 txt 文件形式保存。

3) 查询图书信息

以书号、书名、作者三种方法查询书籍, 查询到后显示该书籍。

4) 显示学生信息

输入学生的学号, 显示该学生的全部信息, 若无该学生则显示“查无此人”。

5) 借阅书籍

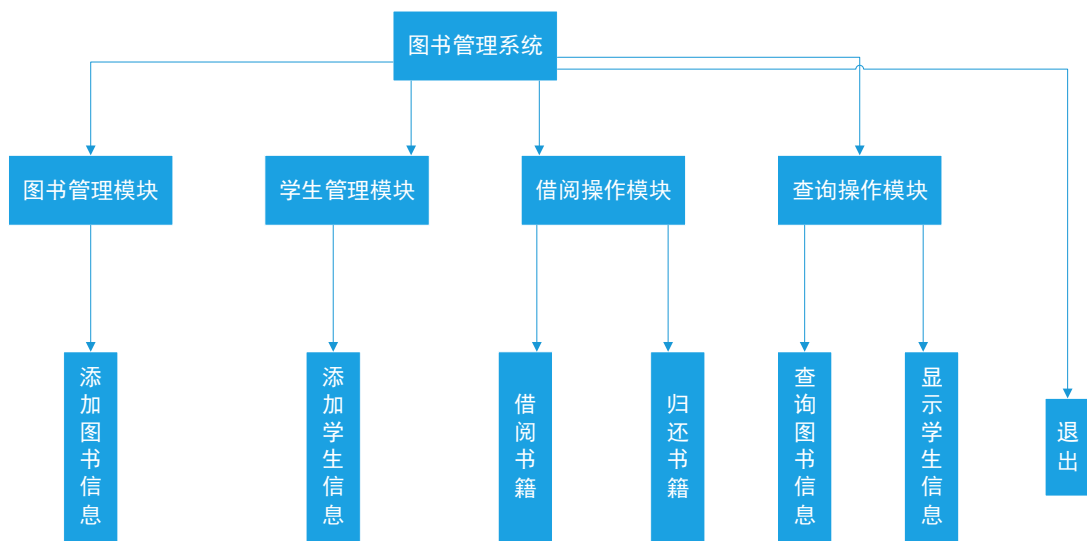
输入学生借书证号以及书籍的书号, 在学生的借书信息中增加该书号, 该书的存量减一, 若无该学生或该图书则显示错误。

6) 归还书籍

输入学生借书证号以及书籍的书号, 在学生的借书信息中删去该书号, 该书的存量加一, 若无该学生或该图书则显示错误。

7) 退出

退出图书管理系统并保存学生和图书信息。



## 2.2 哈夫曼编码

分为个模块：

### 1) 计算字符的权值

读取已存储的文本文件，计算出其中各字符的权值（频次），并用一个 txt 文件保存。

### 2) 初始化哈夫曼树

对每个结点赋字符的值和权值，并计算字符个数。

### 3) 构造哈夫曼树

每次将最小权值的两个结点合并，新节点的权值为两个结点权值之和，形成一颗哈夫曼树。

### 4) 哈夫曼编码

从根节点出发，左子节点为“0”，右子节点为“1”，直到叶子结点，对每个字符进行编码。

### 5) 哈夫曼译码

从根节点开始搜索编码，到叶子结点停止输出字符，直到编码全部读取。

### 6) 用户输入字符

提供一个输入字符的界面，对用户输入的字符进行哈夫曼编码并输出。

### 7) 用户输入编码

提供一个输入编码的界面，对用户输入的字符进行哈夫曼译码并输出。

### 8) 计算压缩比

计算从文本文件读入的文本所形成的编码文件大小  $m$ ，文本文件未编码前大小为  $n$ ，以公式  $(n-m)/n$  来计算压缩比。

## 三. 详细设计

### 3.1 图书管理系统

#### 3.1.1 存储结构

图书的基本信息为书号、书名、作者、存量以及总量。

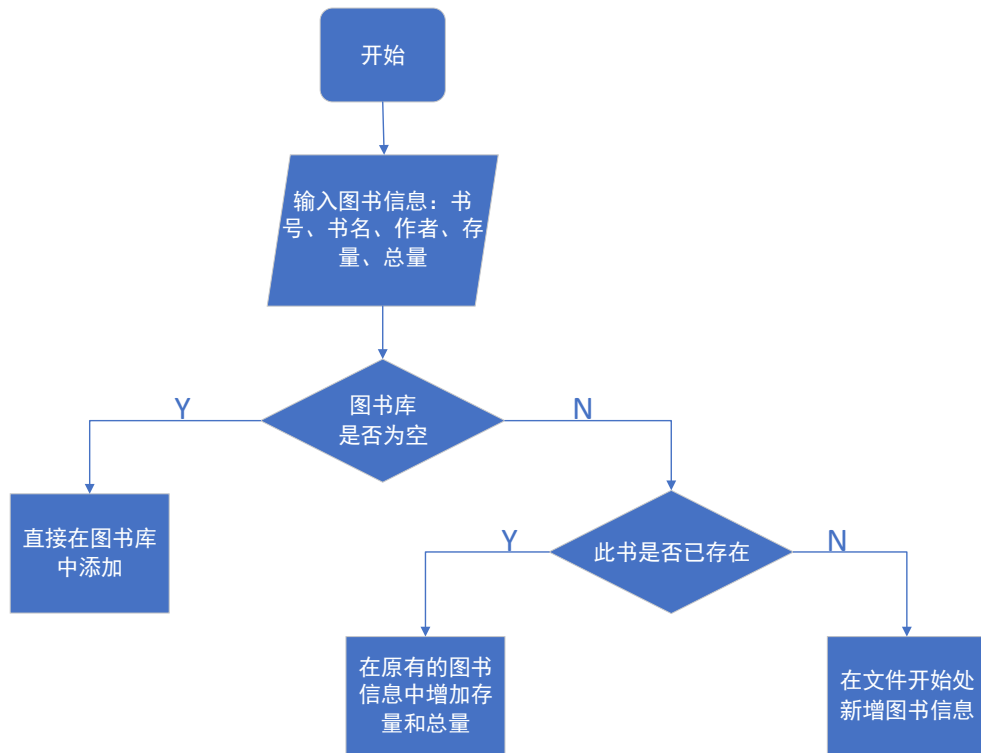
```
typedef struct book{
    long call;           //书号
    char bookname[max];  //书名
    char author[max];    //作者
    int now;             //存量
    int total;           //总量
} book;
```

学生的基本信息为学号（借书证号）、姓名、性别、院系、借阅的书籍书号（最多五本）。

```
typedef struct student{
    long number;         //学号
    char name[max];      //姓名
    char sex[max];       //性别
    char department[max]; //院系
    long Borrowed[maxBook]; //借阅的书籍
} student;
```

#### 3.1.2 采编入库

添加图书入库，保存在 Book.txt 文件中，当有新书入库时，创建一个 ChangedBook.txt 文件，将新书放置于文件开始处，删除原有的 Book.txt 文件，将 ChangedBook.txt 文件改名为 Book.txt。



//采编入库

```

void addBook() {
    book bk;    //原有的书籍
    book addbk; //新加入的书籍
    char ch;    //读取文件第一个字符，判断文件是否为空
    FILE *fp;   //源文件
    FILE *fq;   //替换文件
    int isTrue = 0; //此书是否存在
    int flag; //文件是否为空
    if ((fp = fopen("book.txt", "a+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    ch = fgetc(fp);
    if (ch == EOF) //若不为空，则修改 flag
        flag = 0;
    else
        flag = 1;
    if ((fq = fopen("ChangedBook.txt", "w+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    printf("*****输入你要写入的图书信息*****\n");
    printf("依次输入：\n          书号，书名，作者，存量，总量\n");
    //输入新加入的图书
  
```

```

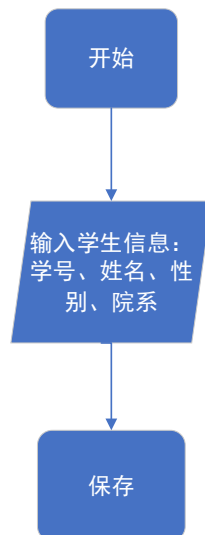
        scanf("%ld %s %s %d %d", &addbk.call, addbk.bookname, addbk.author,
&addbk.now, &addbk.total);
        if (flag != 0) { //若文件不为空
            fseek(fp, 0L, 0);
            while (!feof(fp)) {
                fscanf(fp, "%ld %s %s %d %d\n", &bk.call, bk.bookname, bk.author,
&bk.now, &bk.total);
                if (addbk.call == bk.call) { //若书已存在, 则加存量与总量
                    bk.now += addbk.now; //存量增加
                    bk.total += addbk.total; //总量增加
                    isTrue = 1; //图书存在
                }
                //转移到替换文件 fq 中
                fprintf(fq, "%ld %s %s %d %d\n", bk.call, bk.bookname, bk.author,
bk.now, bk.total);
            }
            if(isTrue == 0) { //若图书不存在
                fprintf(fq, "%ld %s %s %d %d\n", addbk.call, addbk.bookname,
addbk.author, addbk.now, addbk.total);
            }
        }
        else{ //若文件为空, 直接添加
            fprintf(fq, "%ld %s %s %d %d\n", addbk.call, addbk.bookname,
addbk.author, addbk.now, addbk.total);
        }
        printf("添加成功\n");
        if (fclose(fp)) {
            printf("关闭文件失败\n");
            exit(0);
        }
        if (fclose(fq)) {
            printf("关闭文件失败\n");
            exit(0);
        }
    }
}

```

### 3.1.3 添加学生信息

输入要添加的学生信息：学号、姓名、性别、院系，并初始化借书情况为无（即借阅书籍处全为0），保存在 Student.txt 文件中。





//添加学生信息

```

void addStudent() {
    student stu;    //新加入的学生
    FILE *fp;    //存储学生的文件
    int flag = 0; //初始化为未借书
    if ((fp = fopen("student.txt", "a")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    fseek(fp, 0L, 0);
    printf("*****输入你要写入的学生信息*****\n");
    printf("依次输入：\n          学号，姓名，性别，院系\n");
    //读取新加入的学生
    scanf("%ld %s %s %s", &stu.number, stu.name, stu.sex, stu.department);
    //存储新加入的学生
    fprintf(fp, "%ld %s %s %s %ld %ld %ld %ld %ld\n", stu.number, stu.name,
stu.sex, stu.department, flag, flag, flag, flag, flag);
    printf("添加成功\n");
    if (fclose(fp)) {
        printf("关闭文件失败\n");
        exit(0);
    }
}

```

### 3.1.4 按书号查询图书信息

输入图书的书号，在 Book.txt 文件中每次读入一行来搜索此书，若找到则输出图书信息，若未找到则显示错误。

//按书号查询图书信息

```

void searchBook_call() {
    book bk;    //已存储的图书
    long value; //查询的图书书号
    FILE *fp;   //图书文件
    int flag = 0; //是否找到书, 找到为 1, 未找到则为 0
    if ((fp = fopen("book.txt", "r")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    printf("输入查询的图书书号: ");
    //输入查询的图书书号
    scanf("%d", &value);
    while (!feof(fp)) {
        //遍历输入已储存的图书
        fscanf(fp, "%ld %s %s %d %d", &bk.call, bk.bookname, bk.author,
&bk.now, &bk.total);
        //查询到了
        if (value == bk.call) {
            //输出图书信息
            printf("依次输出: 书号, 书名, 作者, 存量, 总量\n %ld %s %s %d %d\n", bk.call, bk.bookname, bk.author, bk.now, bk.total);
            flag = 1;    //已找到此书
            break;
        }
    }
    if (flag == 0) {
        printf("未找到此书\n");
    }
    if (fclose(fp)) {
        printf("关闭文件失败\n");
        exit(0);
    }
}
}

```

### 3.1.5 按书名查询图书信息

输入图书的书名, 在 Book.txt 文件中每次读入一行来搜索此书, 若找到则输出图书信息, 若未找到则显示错误。

*//按书名查询图书信息*

```

void searchBook_bookname() {
    book bk;    //已存储的图书
    char bkname[max]; //要查询的图书书名

```

```

FILE *fp;    //已存储的图书文件
int flag = 0;    //是否找到
if ((fp = fopen("book.txt", "r")) == NULL) {
    printf("打开文件错误\n");
    exit(0);
}
printf("输入你要查询的图书的书名: ");
//输入图书书名
scanf("%s", bkname);
while (!feof(fp)) {
    //遍历查找已存储的图书
    fscanf(fp, "%ld %s %s %d %d", &bk.call, bk.bookname, bk.author,
&bk.now, &bk.total);
    //找到了
    if (strcmp(bk.bookname, bkname) == 0) {
        //输出图书信息
        printf("依次输出: 书号, 书名, 作者, 存量, 总量
\n %ld %s %s %d %d\n", bk.call, bk.bookname, bk.author, bk.now, bk.total);
        flag = 1;    //已找到
        break;
    }
}
if (flag == 0) {
    printf("未找到此书\n");
}
if (fclose(fp)) {
    printf("关闭文件失败\n");
    exit(0);
}
}

```

### 3.1.6 按作者查询图书信息

输入图书的作者, 在 Book.txt 文件中每次读入一行来搜索此书, 若找到则输出图书信息, 若未找到则显示错误。

```

//按作者查询图书信息
void searchBook_author() {
    book bk;    //已存储的图书
    char name[max]; //要查找的作者
    int flag = 0;    //是否找到
    FILE *fp;    //已存储的图书文件
    if ((fp = fopen("book.txt", "r")) == NULL) {

```

```

        printf("打开文件错误\n");
        exit(0);
    }
    printf("输入你要查询的图书的作者: ");
    //输入图书作者
    scanf("%s", name);
    while (!feof(fp)) {
        //遍历查找已存储的图书
        fscanf(fp, "%ld %s %s %d %d", &bk.call, bk.bookname, bk.author,
&bk.now, &bk.total);
        //找到了
        if (strcmp(bk.author, name) == 0) {
            //输出图书信息
            printf("依次输出: 书号, 书名, 作者, 存量, 总量
\n %ld %s %s %d %d\n", bk.call, bk.bookname, bk.author, bk.now, bk.total);
            flag = 1;    //已找到
            break;
        }
    }
    if (flag == 0) {
        printf("未找到此书\n");
    }
    if (fclose(fp)) {
        printf("关闭文件失败\n");
        exit(0);
    }
}

```

### 3.1.7 显示学生信息

输入学生学号（借书证号），在 Student.txt 中每次读取一行，若找到则输出学生信息，若未找到则显示错误。

*//用学生的借书证号（学号）显示学生信息*

```

void showStudent() {
    student stu;    //已存储的学生
    long value;    //要查找的学生学号
    int flag = 0;    //是否找到学生
    FILE *fp;    //已存储的学生信息
    if ((fp = fopen("student.txt", "r")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    printf("输入学生借书证号: ");
}

```

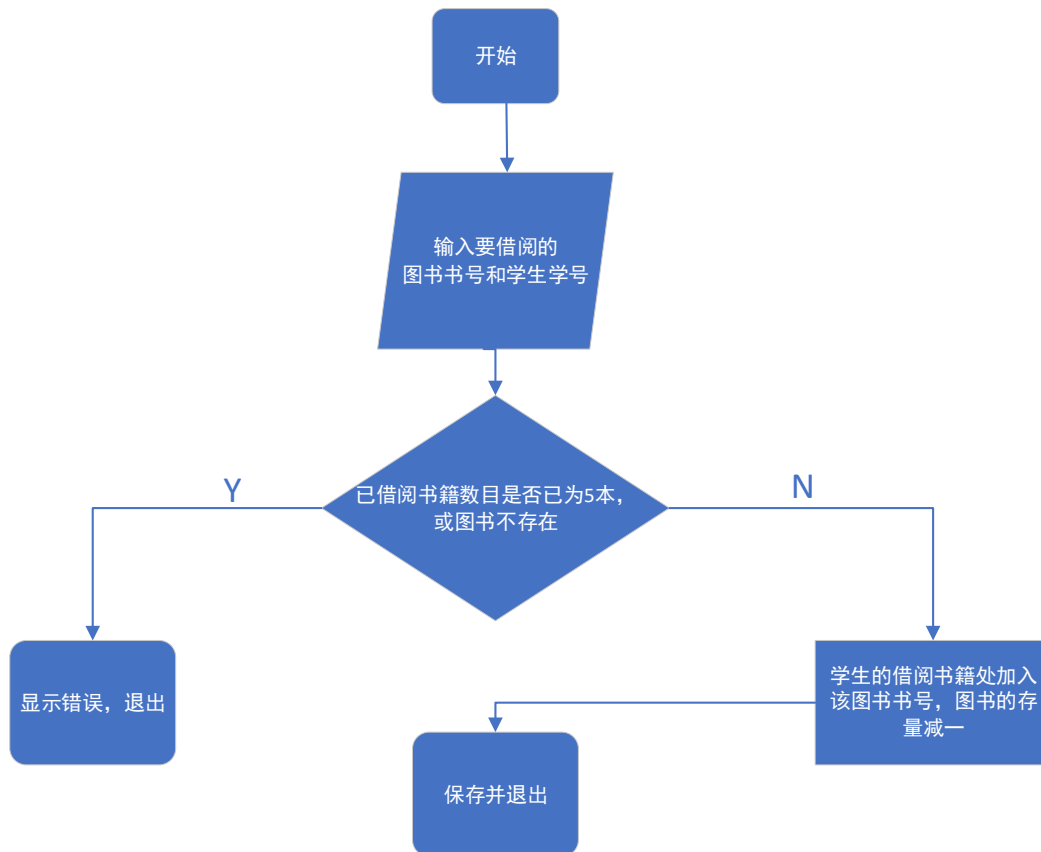
```

//输入学生学号
scanf("%ld", &value);
while (!feof(fp)){
    //遍历已存储的学生查找
    fscanf(fp, "%ld %s %s %s %f %f %f %f %f", &stu.number, stu.name,
stu.sex, stu.department, &stu.Borrowed[0], &stu.Borrowed[1], &stu.Borrowed[2],
&stu.Borrowed[3], &stu.Borrowed[4]);
    //找到了
    if (value == stu.number){
        int i;
        //输出学号, 姓名, 院系
        printf("依次输出: 学号, 姓名, 性别, 院系, 借阅书籍:
\n%ld %s %s %s", stu.number, stu.name, stu.sex, stu.department);
        //输出借阅书籍
        for (i = 0; (stu.Borrowed[i] != 0) && (i < 5); i++){
            printf(" %ld\n", stu.Borrowed[i]);
        }
        if (i == 0) //没有借书
            printf("\n");
        flag = 1;    //已找到
        break;
    }
}
if (flag == 0){
    printf("未找到此人\n");
}
if (fclose(fp)){
    printf("关闭文件失败\n");
    exit(0);
}
}

```

### 3.1.8 借阅书籍

分别输入要借阅的图书书号、学生学号，若该学生已借阅书籍数目已为 5 本，或没有此书，则显示错误。否则，在该学生的借阅书籍中加入该图书的书号，并将图书的存量减一。



*//借阅书籍*

```

void borrowBook() {
    book bk;    //已存储的图书
    student stu; //已存储的学生
    long number; //要借书的学生学号
    long booknumber; //要借的图书书号
    int flag = 0; //是否有这本书, 是否有此人
    FILE *fp;    //原图书文件
    FILE *fq;    //原学生文件
    FILE *fc;    //改变后的书籍存储文件
    FILE *fb;    //改变后的学生信息存储文件
    if ((fp = fopen("book.txt", "r+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    if ((fq = fopen("student.txt", "r+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    if ((fc = fopen("copyBook.txt", "w+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
}
  
```

```

}
if((fb = fopen("copyStudent.txt", "w+")) == NULL) {
    printf("打开文件错误\n");
    exit(0);
}
printf("输入要借阅的书籍书号: ");
scanf("%ld", &booknumber);
printf("输入你的学号: ");
scanf("%ld", &number);
while (!feof(fp)) {
    //遍历查找图书
    fscanf(fp, "%ld %s %s %ld %ld\n", &bk.call, bk.bookname, bk.author,
&bk.now, &bk.total);
    //找到了
    if (booknumber == bk.call) {
        bk.now--; //图书存量减少
        flag = 1; //已找到
    }
    //转移到替换文件 fc 中
    fprintf(fc, "%ld %s %s %ld %ld\n",
bk.call, bk.bookname, bk.author, bk.now, bk.total);
}
if(flag == 0) {
    printf("没有这本书\n");
    return;
}
if (fclose(fp)) {
    printf("关闭文件失败\n");
    exit(0);
}
if (fclose(fc)) {
    printf("关闭文件失败\n");
    exit(0);
}
while (!feof(fq)) {
    //遍历查找学生
    fscanf(fq, "%ld %s %s %s %ld %ld %ld %ld %ld\n", &stu.number, stu.name,
stu.sex, stu.department, &stu.Borrowed[0], &stu.Borrowed[1], &stu.Borrowed[2],
&stu.Borrowed[3], &stu.Borrowed[4]);
    //找到了
    if (number == stu.number) {
        flag = 1; //已找到
        int i = 0;
        while ((i<5) && (stu.Borrowed[i] != 0))

```

```

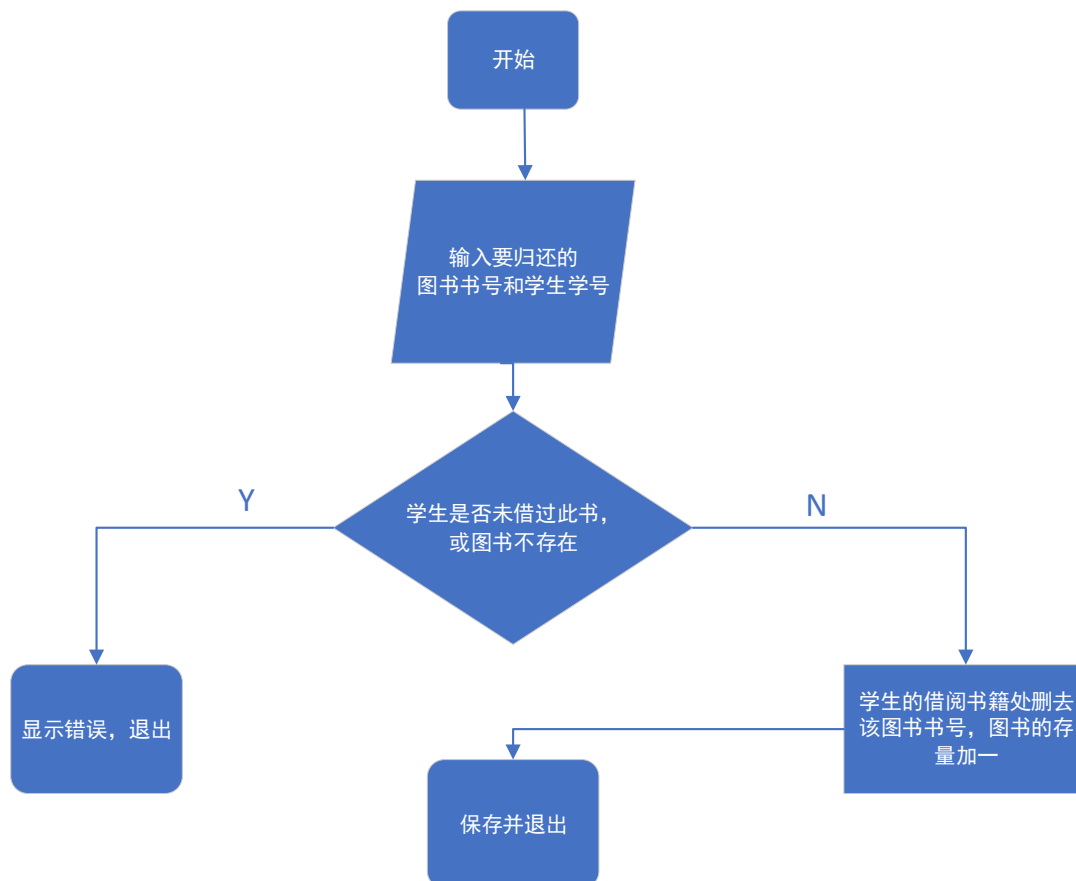
        i++;
    if(i>4) {
        printf("借书数量超过最大值");
        return;
    }
    stu.Borrowed[i] = booknumber;    //改变借阅书籍
}
//转移到替换文件 fb 中
fprintf(fb, "%ld %s %s %s %ld %ld %ld %ld %ld\n", stu.number, stu.name,
stu.sex, stu.department, stu.Borrowed[0], stu.Borrowed[1], stu.Borrowed[2],
stu.Borrowed[3], stu.Borrowed[4]);
}
if(flag == 0) {
    printf("没有这个人\n");
    return;
}
if (fclose(fq)) {
    printf("关闭文件失败\n");
    exit(0);
}
if (fclose(fb)) {
    printf("关闭文件失败\n");
    exit(0);
}
printf("借书成功\n");
}

```

### 3.1.9 归还书籍

分别输入要归还的图书书号、学生学号，若该学生未借过此书，或不存在此书，则显示错误。否则，在该学生的借阅书籍中删去该图书的书号，并将图书的存量加一。





//归还书籍

```

void returnBook() {
    book bk;    //已存储的图书
    student stu; //已存储的学生
    long number; //要还书的学生学号
    long booknumber; //要还的书籍书号
    int flag = 0; //是否有此书, 是否借了此书, 是否有此人
    FILE *fp; //原图书文件
    FILE *fq; //原学生文件
    FILE *fc; //替换的图书文件
    FILE *fb; //替换的学生文件
    if ((fp = fopen("book.txt", "r+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    if ((fq = fopen("student.txt", "r+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    if ((fc = fopen("copyBook.txt", "w+")) == NULL) {
        printf("打开文件错误\n");
    }
}
  
```

```

        exit(0);
    }
    if ((fb = fopen("copyStudent.txt", "w+")) == NULL) {
        printf("打开文件错误\n");
        exit(0);
    }
    printf("输入要归还的书籍书号: ");
    scanf("%ld", &booknumber);
    printf("输入你的学号: ");
    scanf("%ld", &number);
    while (!feof(fp)) {
        //遍历查找图书
        fscanf(fp, "%ld %s %s %ld %ld\n", &bk.call, bk.bookname, bk.author,
&bk.now, &bk.total);
        //找到了
        if (booknumber == bk.call) {
            bk.now++; //书籍存量增加
            flag = 1; //已找到
        }
        //转移到替换文件 fc 中
        fprintf(fc, "%ld %s %s %ld %ld\n", bk.call, bk.bookname, bk.author,
bk.now, bk.total);
    }
    if(flag == 0) {
        printf("没有这本书\n");
        return;
    }
    if (fclose(fp)) {
        printf("关闭文件失败\n");
        exit(0);
    }
    if (fclose(fc)) {
        printf("关闭文件失败\n");
        exit(0);
    }
    while (!feof(fq)) {
        //遍历查找学生
        fscanf(fq, "%ld %s %s %s %ld %ld %ld %ld %ld\n", &stu.number, stu.name,
stu.sex, stu.department, &stu.Borrowed[0], &stu.Borrowed[1], &stu.Borrowed[2],
&stu.Borrowed[3], &stu.Borrowed[4]);
        //找到了
        if (number == stu.number) {
            flag = 1; //已找到
            int i = 0;

```

```

        while ((i<5) && (stu.Borrowed[i] != booknumber))    //借的书与此书不
相同
        {
            i++;
            if(i>4) {
                printf("你没有借这本书\n");
                return;
            }
            stu.Borrowed[i] = 0;    //重置已借书籍
        }
        //转移到替换文件 fb 中
        fprintf(fb, "%ld %s %s %s %ld %ld %ld %ld %ld\n", stu.number, stu.name,
stu.sex, stu.department, stu.Borrowed[0], stu.Borrowed[1], stu.Borrowed[2],
stu.Borrowed[3], stu.Borrowed[4]);
    }
    if(flag == 0) {
        printf("你没有借这本书\n");
        return;
    }
    if (fclose(fq)) {
        printf("关闭文件失败\n");
        exit(0);
    }
    if (fclose(fb)) {
        printf("关闭文件失败\n");
        exit(0);
    }
    printf("还书成功\n");
}

```

## 3.2 哈夫曼编码

### 3.2.1 存储结构

哈夫曼结点的信息为：字符、权值、父节点、左子节点、右子节点。

```

typedef struct {
    char ch;    //存储的字符
    int weight; //权值，这个字符出现的频率
    int parent; //父节点
    int left;   //左子节点
    int right;  //右子节点
} HuffNode;

```

哈夫曼编码信息：哈夫曼编码、编码开始处。

```
typedef struct{
    char code[MAXNUM]; //Huffman 编码
    int start;
}HuffCode;
```

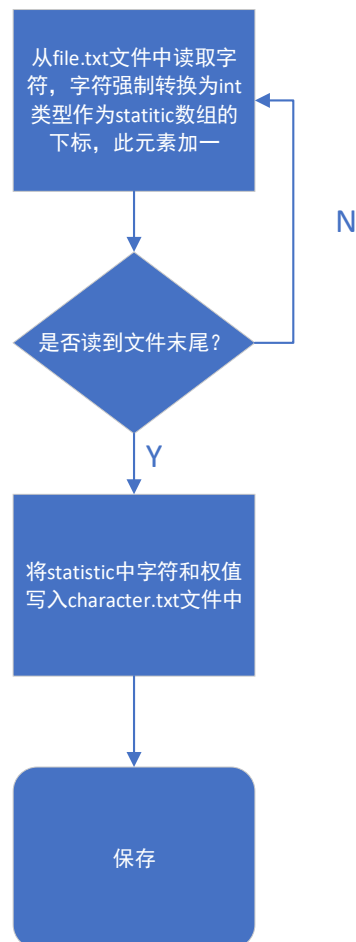
另使用两个数组来分别存放哈夫曼树和每个字符的对应编码，并用  $n$  表示字符个数。

```
HuffNode ht[MAXNUM*2]; //存放哈夫曼树
HuffCode hcd[MAXNUM]; //存放 ht 数组中对应的字符的编码
int n; //字符的个数
```

### 3. 2. 2 计算字符的权值

使用名为 `statistic` 的数组来存储权值，从 `file.txt` 文件中读取字符，每读入一个字符，将其强制转换为 `int` 类型作为 `statistic` 数组的下标，并令此元素加一，从而将字符的权值保存在 `statistic` 数组里。

遍历 `statistic` 数组，将其打印到 `character.txt` 文件中保存。



```

//计算字符的权值
void count() {
    char file[MAXCHAR]; //存储字符
    int statistic[CHARNUM]; //存储权值
    int i;
    char c; //记录读到的字符
    FILE *fp, *fq;
    for(i=0; i<CHARNUM; i++) {
        statistic[i] = 0;
    }
    if((fp = fopen("file.txt", "r")) == NULL)
    {
        printf("打开文件失败\n");
        exit(0);
    }
    if((fq = fopen("character.txt", "w+")) == NULL)
    {
        printf("打开文件失败\n");
        exit(0);
    }
    while((c = getc(fp)) != EOF) { //记录权值
        statistic[(unsigned int)c]++;
    }
    if(fclose(fp)) {
        printf("关闭文件 1 失败");
        exit(0);
    }
    for(i=0; i<CHARNUM; i++) {
        if(statistic[i] != 0) { //若字符出现过, 则加入
            file[i] = (char)i;
            fprintf(fq, "%c%d\n", file[i], statistic[i]);
        }
    }
    if(fclose(fq)) {
        printf("关闭文件失败");
        exit(0);
    }
    printf("已将 file.txt 的字符及权值写入 character.txt\n");
}

```

//初始化哈夫曼树 ht

```

void initHt() {
    FILE * fp; //存储编码字符和权值的文件
    char ch; //文件中字符

```

```

int i = 0;
int j;
//character.txt 中读出要编码的字符和权值
if((fp = fopen("character.txt", "r")) == NULL) {
    printf("打开文件失败\n");
    exit(0);
}
//初始化根结点
ht[i].left = ht[i].right = ht[i].parent=-1;
//遍历读取编码文件
while((ch = fgetc(fp))!=EOF) {
    if(ch == '\n') {    //初始化下一个结点
        i++;
        ht[i].left = ht[i].right = ht[i].parent=-1;
    }
    //结点的权值赋值
    else if(ch>='0' && ch<='9')
        ht[i].weight = ht[i].weight*10 + ch-'0';
    //结点的字符赋值
    else
        ht[i].ch = ch;
}
n = i+1;    //计算字符个数
//初始化后 n-1 个结点
for(j=n; j<2*n-1; j++) {
    ht[j].parent = ht[j].left = ht[j].right = -1;
    ht[j].weight = -1;
}
if(fclose(fp)) {
    printf("关闭文件错误");
    exit(0);
}
printf("已将 character.txt 字符权值写入哈夫曼结点中\n");
}

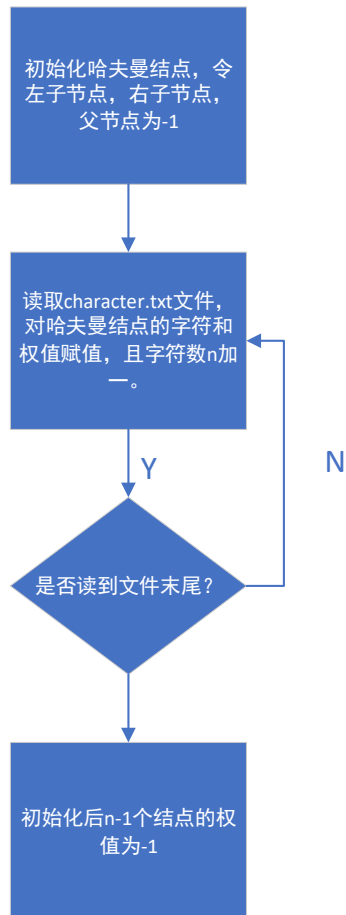
```

### 3.2.3 初始化哈夫曼树

先初始化哈夫曼结点的左子节点，右子节点，父节点都为-1。

遍历 character.txt 文件，对前 n 个结点的字符和权值赋值，并记录字符数目 n。

赋后 n-1 个结点的权值为-1。



```

//初始化哈夫曼树 ht
void initHt() {
    FILE * fp; //存储编码字符和权值的文件
    char ch;    //文件中字符
    int i = 0;
    int j;
    //character.txt 中读出要编码的字符和权值
    if((fp = fopen("character.txt", "r")) == NULL) {
        printf("打开文件失败\n");
        exit(0);
    }
    //初始化根结点
    ht[i].left = ht[i].right = ht[i].parent = -1;
    //遍历读取编码文件
    while((ch = fgetc(fp)) != EOF) {
        if(ch == '\n') { //初始化下一个结点
            i++;
            ht[i].left = ht[i].right = ht[i].parent = -1;
        }
        //结点的权值赋值

```

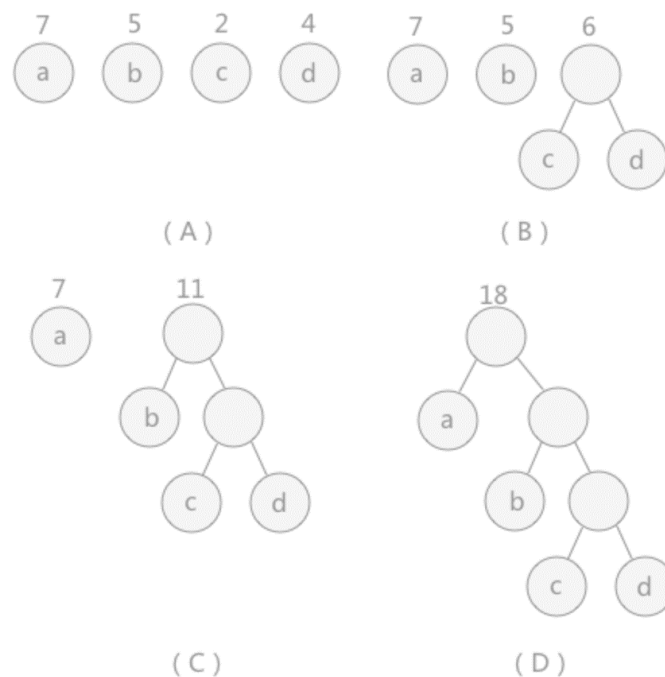
```

else if(ch>='0' && ch<='9')
    ht[i].weight = ht[i].weight*10 + ch-'0';
//结点的字符赋值
else
    ht[i].ch = ch;
}
n = i+1;    //计算字符个数
//初始化后 n-1 个结点
for(j=n; j<2*n-1; j++) {
    ht[j].parent = ht[j].left = ht[j].right = -1;
    ht[j].weight = -1;
}
if(fclose(fp)) {
    printf("关闭文件错误");
    exit(0);
}
printf("已将 character.txt 字符权值写入哈夫曼结点中\n");
}

```

### 3.2.4 构造哈夫曼树

每次在前  $n$  个结点中选取权值最小的两个结点，合并成一个新结点保存在  $ht$  数组的后  $n-1$  个元素中，直到  $ht$  数组的第  $2n-1$  个元素被赋值。



//构造哈夫曼树,看成有  $n$  棵树,选择权值最小的两棵树合并

```

void createHuffTree() {

```



```

int i, j, m1, m2; //m1 为最小权值, m2 为次小权值
int minI, minJ; //minI 为权值最小结点, minJ 为权值次小结点
minI = minJ = -1; //minI < minJ
for(i=0; i<n-1; i++) {
    m1 = m2 = MAXNUM;
    minI = minJ = -1;
    for(j=0; j<n+i; j++) { //找两个最小权的无父节点的结点
        if(ht[j].weight < m1 && ht[j].parent == -1) {
            m2 = m1;
            minJ = minI;
            m1 = ht[j].weight;
            minI = j;
        }
        else if(ht[j].weight < m2 && ht[j].parent == -1) {
            m2 = ht[j].weight;
            minJ = j;
        }
    }
    ht[minI].parent = ht[minJ].parent = n+i;
    ht[n+i].weight = m1+m2;
    ht[n+i].left = minI;
    ht[n+i].right = minJ;
}
ht[2*n-2].parent = -1;
printf("已创建哈夫曼树\n");
}

```

### 3.2.5 将一个字符反转

```

//将一个字符串反转
void reverse(char *str) {
    int i, j;
    char ch;
    for(i=0, j = strlen(str)-1; i<j; i++, j--) {
        ch = str[i];
        str[i] = str[j];
        str[j] = ch;
    }
}

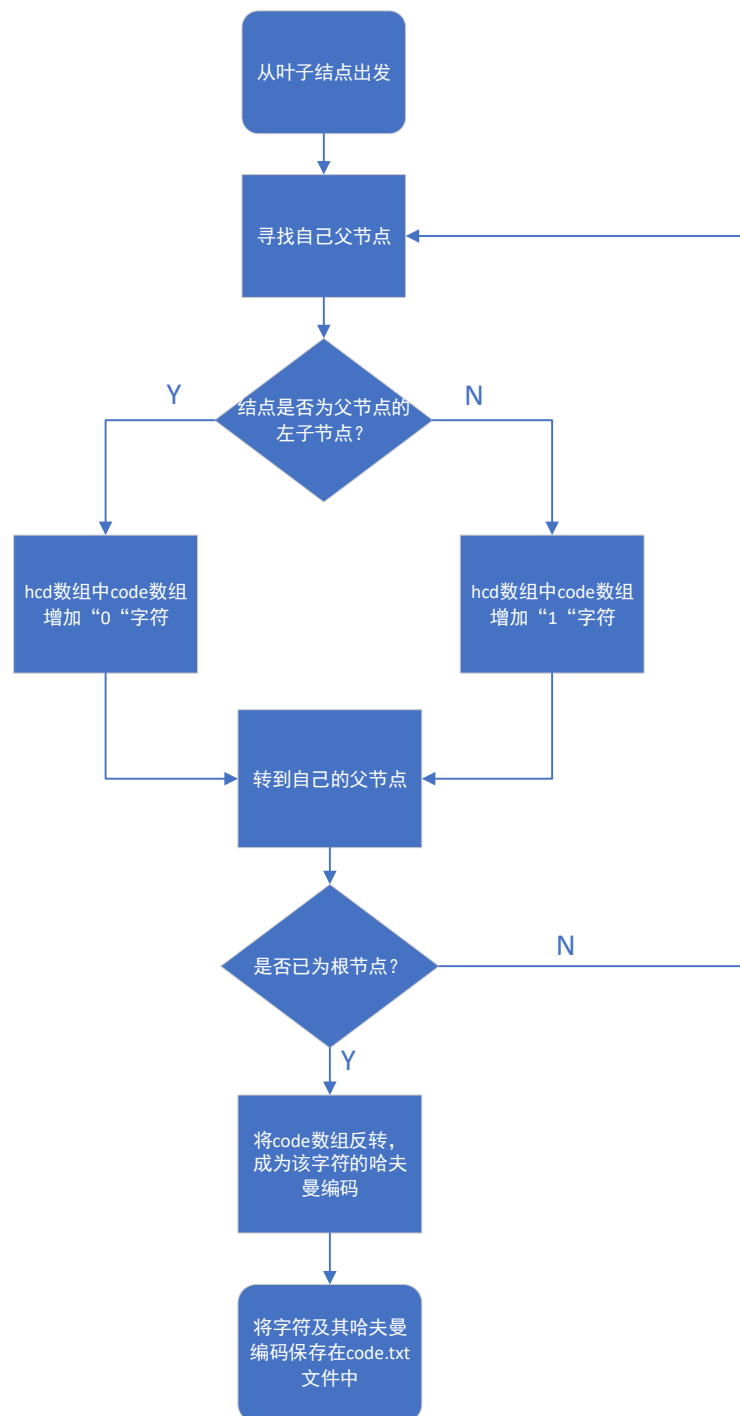
```

### 3.2.6 哈夫曼编码

使用 hcd 数组来存储字符的哈夫曼编码，从叶子结点从下往上遍历 ht 哈夫曼树，若为左子节点则编码为“0”，若为右子节点则编码为“1”，直到根节点，最后将获得的编码串

反转，则得到该字符的哈夫曼编码。

将字符和它的哈夫曼编码写入到 code.txt 文件中保存。



*//哈夫曼编码，通过父节点从下往上找*

```
void createHuffCode() {  
    int i, j, length;  
    FILE * fp; //存放哈夫曼编码的文件  
    for(i=0; i<n; i++) {
```

```

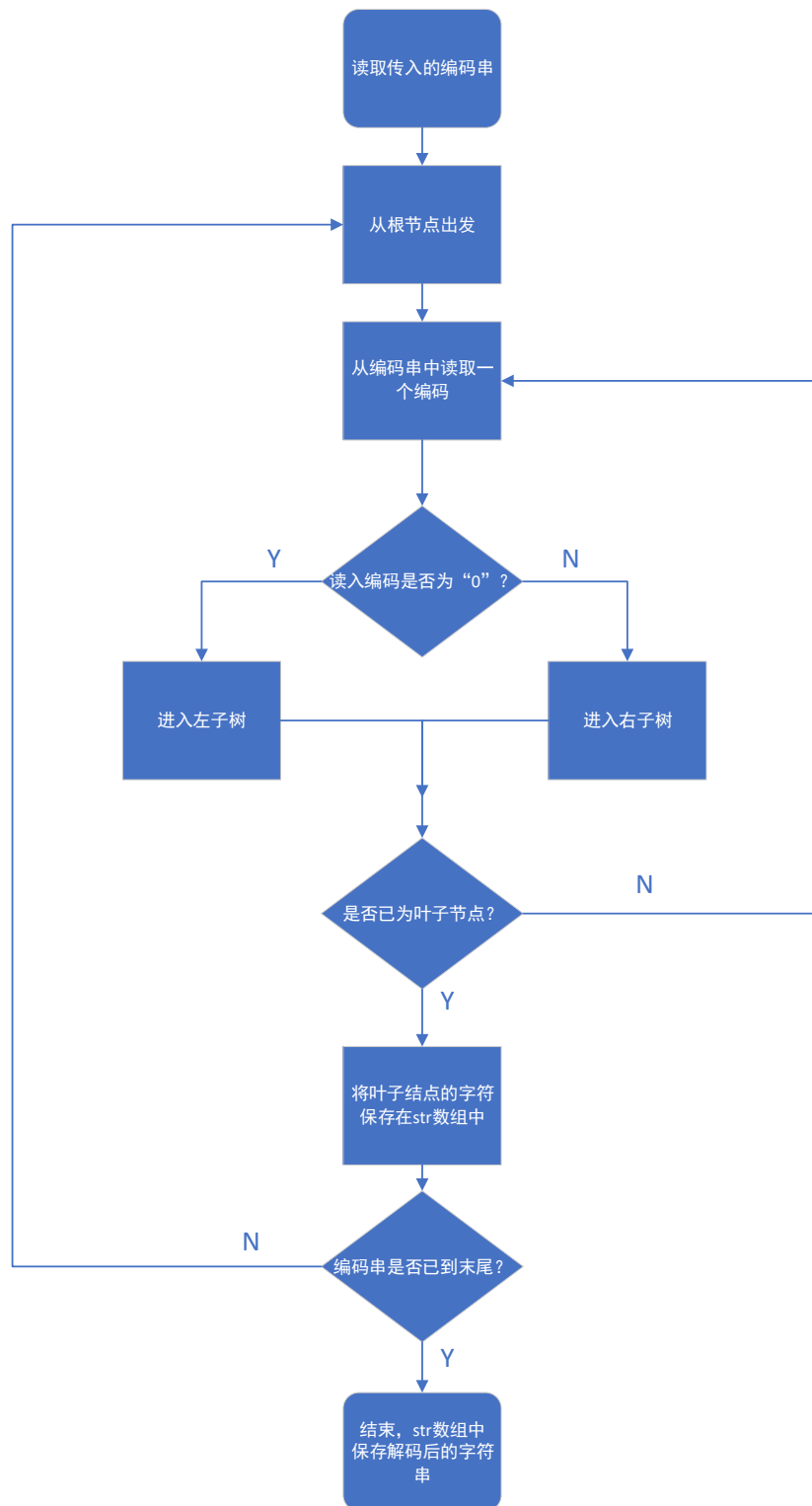
length = 0;
j = i;
//给每个字符进行编码
while(ht[j].parent != -1) {
    //若 j 为左子结点, 则编码 0
    if(ht[ht[j].parent].left == j) {
        hcd[i].code[length++] = 0+'0';
    }
    //若 j 为右子结点, 则编码 1
    else{
        hcd[i].code[length++] = 1+'0';
    }
    j = ht[j].parent;    //不断向上
}
hcd[i].start = hcd[i].code[length-1]-'0'; //记录开始位置
hcd[i].code[length] = '\0';    //最后为结束符
reverse(hcd[i].code);    //反转来完成编码
}
if((fp=fopen("code.txt", "w+"))==NULL) {
    printf("打开文件错误\n");
    exit(0);
}
//把 hcd 字符编码写入文件 code.txt 中
for(i=0; i<n; i++) {
    fputc(ht[i].ch, fp);
    fputc(" ", fp);
    fputs(hcd[i].code, fp);
    fputc('\n', fp);
}
if(fclose(fp)) {
    printf("can not close the file character.txt");
    exit(0);
}
printf("已创建哈夫曼编码, 保存在 code.txt 中\n");
}

```

### 3.2.7 哈夫曼译码

从哈夫曼树 ht 的根节点出发, 读到“0”编码则转到其左子节点, 读到“1”编码则转到其右子节点, 直到读到叶子节点, 将叶子节点的字符保存在 str 数组中。

若要读的编码串还没到末尾, 则继续从根节点出发, 继续上述步骤。



*//哈夫曼译码，每次都从根节点开始搜索1*

```

int releaseHuffCode(char *str, char* code) {
    int root = 2*n-2;
    int length = 0, i = 0;
    //遍历解码
    while(code[i]) {

```

```

        //编码为 0 则为左子树
        if(code[i] == '0')
            root = ht[root].left;
        //编码为 1 则为右子树
        else if(code[i] == '1')
            root = ht[root].right;
        //解码失败
        else
            return 0;
        //若为树叶
        if(ht[root].left == -1 && ht[root].right == -1) {
            str[length++] = ht[root].ch;    //赋值
            root = 2*n-2;    //重新从根节点往下
        }
        i++;
    }
    str[length]='\0';
    //解码成功
    if(root==2*n-2)
        return 1;
    return 0;
}

```

### 3.2.8 用户输入字符

获取用户输入的字符串，用哈夫曼编码将其编码并输出。

```

//用户输入字符
void encode() {
    int i = 0, j;
    char str[500];    //存储字符
    char code[5000]={'\0'};    //存储编码
    printf("\n 请输入要编码的字符串 (length<50)\n");
    gets(str);
    while(str[i]) {
        //遍历寻找字符
        for(j=0; j<n; j++) {
            //找到了字符
            if(str[i] == ht[j].ch) {
                strcat(code, hcd[j].code);    //连接编码
                break;
            }
        }
        i++;
    }
}

```

```

    }
    printf("输入字符串的编码为:    ");
    puts(code); //输出编码
}

```

### 3.2.9 用户输入解码字串

获取用户输入的解码子串，将其译码为哈夫曼编码并输出。

```

//用户输入解码字串
void decode() {
    char str[500]; //存储字符
    char code[5000]; //存储编码
    printf("\n 请输入要译码的字串(用 0 和 1 表示)\n");
    scanf("%s", code);
    //若解码成功
    if(releaseHuffCode(str, code)) {
        printf("输入的编码的译码为: ");
        puts(str);
    } else
        printf("你输入的字串错误! \n");
}

```

### 3.2.10 计算压缩比

压缩后文件所占大小为每个字符的哈夫曼编码长度之和，其与原文件大小之差记为 up。

原文件大小为每个字符等长编码的长度乘以字符数，记为 down。

压缩比即为 up/down。

```

//计算压缩比
float compressRate() {
    double up=0.0, down=0.0; //up 为压缩后文件大小与原文件之差, down 为压缩前原文件
    大小
    for(int i=0; i<n; i++)
        up += ht[i].weight*(n-hcd[i].start + 1);
    int x = ceil(log2(n)); //未压缩前每个字符的大小
    down = n*x;
    if(down!=0) return up/down;
}

```

## 四. 调试分析

### 4.1 图书管理系统

#### 4.1.1 欢迎界面

程序运行显示欢迎界面，有全组成员的信息，输入 1-7 的各种功能。

```
*****
                        图书管理系统
*
*
*
*
*      版权所有      -----瞿晟
*                        -----王鑫
*                        -----夏洋
*                        -----向景荣
*                        -----刘雅婷
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
```

#### 4.1.2 采编入库

在欢迎界面中输入 1，则可添加书籍。

```
*****
                        图书管理系统
*
*
*
*
*      版权所有      -----瞿晟
*                        -----王鑫
*                        -----夏洋
*                        -----向景荣
*                        -----刘雅婷
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
1
*****输入你要写入的图书信息*****
依次输入:
      书号, 书名, 作者, 存量, 总量
```

依次输入书号，书名，作者，存量，总量即可添加图书入库。  
如依次输入：100 数据结构 张乃孝 31 132。以回车键结束输入。  
打开文本文件 book.txt，则里面存储了这本书的信息。

```
*****输入你要写入的图书信息*****
依次输入：
      书号，书名，作者，存量，总量
100 数据结构 张乃孝 31 132
添加成功
还需要继续添加书籍吗？若是，输入非0，若不是，输入0
```

book.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

100 数据结构 张乃孝 31 132

我们选择再次添加图书，若依然加入书号为 100 的《数据结构》，则系统会在这本书的原有基础上增加其存量和总量。

如输入：100 数据结构 张乃孝 10 20。以回车键结束输入。  
打开文本文件 book.txt，则此书的存量和总量增加。

```
*****输入你要写入的图书信息*****
依次输入：
      书号，书名，作者，存量，总量
100 数据结构 张乃孝 10 20
添加成功
还需要继续添加书籍吗？若是，输入非0，若不是，输入0
```

book.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

100 数据结构 张乃孝 41 152

#### 4.1.3 添加学生信息

在欢迎界面输入 2，则可添加学生信息。



```

输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
2
*****输入你要写入的学生信息*****
依次输入:
    学号, 姓名, 性别, 院系

```

依次输入学号, 姓名, 性别, 院系即可添加学生信息。

如依次输入: 1710210119, 瞿晟, 男, 计院。以回车键结束输入。

打开文本文件 student.txt, 则里面存储了学生信息, 院系信息后的五个 0 表示未借书 (最多借书数量为 5)。

```

*****
                        图书管理系统
*                               *
*                               *
*                               *
*                               *
*   版权所有   _____ 瞿晟   *
*                               _____ 王鑫   *
*                               _____ 夏洋   *
*                               _____ 向景荣 *
*                               _____ 刘雅婷 *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
2
*****输入你要写入的学生信息*****
依次输入:
    学号, 姓名, 性别, 院系
1710210119 瞿晟 男 计院

```

```

student.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1710210119 瞿晟 男 计院 0 0 0 0 0

```

#### 4.1.4 查询图书信息

在欢迎界面输入 3, 则可查询图书信息。

有三种查询方式: 以书号查询, 以书名查询, 以作者查询。

```

*****
图书管理系统
*
*
*
*
*      版权所有      -----瞿晟      *
*      -----王鑫      *
*      -----夏洋      *
*      -----向景荣      *
*      -----刘雅婷      *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询

```

输入 1，选择以书号查询。

```

输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询
1
输入查询的图书书号:

```

输入图书书号，如：100。以回车结束输入。

则以书号，书名，作者，存量，总量的顺序依次显示图书信息。

```

输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询
1
输入查询的图书书号: 100
依次输出: 书号, 书名, 作者, 存量, 总量
100 数据结构 张乃孝 72 284
还需要继续查询书籍吗? 若是, 输入非0, 若不是, 输入0

```

若在查询方式界面输入 2，则选择以书名查询。

```
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询
2
输入你要查询的图书的书名: _
```

输入图书书名，如：数据结构。则可显示该图书信息。

```
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询
2
输入你要查询的图书的书名: 数据结构
依次输出: 书号, 书名, 作者, 存量, 总量
100 数据结构 张乃孝 72 284
还需要继续查询书籍吗? 若是, 输入非0, 若不是, 输入0
```

若在查询界面输入 3，则选择以作者查询。

```
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询
3
输入你要查询的图书的作者:
```

输入图书作者，如：张乃孝。则显示该作者的图书信息。

```
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
3
请输入查询方式:
输入1: 以书号查询, 输入2: 以书名查询, 输入3: 以作者查询
3
输入你要查询的图书的作者: 张乃孝
依次输出: 书号, 书名, 作者, 存量, 总量
100 数据结构 张乃孝 72 284
还需要继续查询书籍吗? 若是, 输入非0, 若不是, 输入0
```

#### 4.1.5 显示学生信息

在欢迎界面输入 4，则可显示学生信息。

```
*****
图书管理系统
*
*
*
*
*      版权所有      -----瞿晟      *
*      -----王鑫      *
*      -----夏洋      *
*      -----向景荣      *
*      -----刘雅婷      *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
4
输入学生借书证号:
```

输入学生的学号（借书证号），如：1710210119。以回车结束输入。

则以学号，姓名，性别，院系，借阅书籍顺序显示学生信息，若未借书，则不会显示借阅书籍信息。

```
*****
图书管理系统
*
*
*
*
*      版权所有      -----瞿晟      *
*      -----王鑫      *
*      -----夏洋      *
*      -----向景荣      *
*      -----刘雅婷      *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
4
输入学生借书证号: 1710210119
依次输出: 学号, 姓名, 性别, 院系, 借阅书籍:
1710210119 瞿晟 男 计院
还需要继续显示学生信息吗, 若是, 输入非0, 若不是, 输入0
```

若输入 100（一个不存在的学生学号），则显示“未找到此人”。

```

输入学生借书证号：100
未找到此人
还需要继续显示学生信息吗，若是，输入非0，若不是，输入0

```

#### 4.1.6 借阅书籍

在欢迎界面输入 5，则可借阅书籍。

```

*****
          图书管理系统
*
*
*
*
*      版权所有      _____瞿晟      *
*                      _____王鑫      *
*                      _____夏洋      *
*                      _____向景荣      *
*                      _____刘雅婷      *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
5
输入要借阅的书籍书号: _

```

输入书籍书号，如：100。以回车结束输入。

```

*      版权所有      _____瞿晟      *
*                      _____王鑫      *
*                      _____夏洋      *
*                      _____向景荣      *
*                      _____刘雅婷      *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
5
输入要借阅的书籍书号: 100
输入你的学号:

```

输入学号，如：1710210119。以回车结束输入。则借书成功。



打开 book.txt 文件，书号为 100 的《数据结构》存量减少了一本。

打开 student.txt 文件，学号为 1710210119 的瞿晟的借阅书籍处增加了 100（借阅书籍的书号）。

```

*          版权所有          -----瞿晟      *
*          -----王鑫      *
*          -----夏洋      *
*          -----向景荣    *
*          -----刘雅婷    *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
5
输入要借阅的书籍书号: 100
输入你的学号: 1710210119
借书成功
还需要继续借书吗? 若是, 输入非0, 若不是, 输入0

```

 book.txt - 记事本	 student.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)	文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
100 数据结构 张乃孝 71 284	1710210119 瞿晟 男 计院 100 0 0 0 0

若输入的图书书号不存在，或学生学号不存在，或学生已借书籍数量已为 5 本，都会显示错误。

4. 1. 7 归还书籍

在欢迎界面输入 6，则可归还书籍。

```

*****
          图书管理系统
*
*
*
*
*          版权所有          -----瞿晟      *
*          -----王鑫      *
*          -----夏洋      *
*          -----向景荣    *
*          -----刘雅婷    *
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
6
输入要归还的书籍书号: 

```

输入图书书号，如：100。以回车结束输入。

```
*****
图书管理系统
*
*
*
*
*      版权所有      -----瞿晟
*      -----王鑫
*      -----夏洋
*      -----向景荣
*      -----刘雅婷
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
6
输入要归还的书籍书号: 100
输入你的学号: 
```

输入学号，如：1710210119。以回车结束输入。则还书成功。

打开 book.txt 文件，书号为 100 的《时间简史》存量增加了一本。

打开 student.txt 文件，学号为 1710210119 的瞿晟的借阅书籍处书号为 100 的书籍被删除。

选择C:\Users\qs486\Desktop\vscode\C\curriculumDesign\libraryMana

```
*****
图书管理系统
*
*
*
*
*      版权所有      -----瞿晟
*      -----王鑫
*      -----夏洋
*      -----向景荣
*      -----刘雅婷
*****
输入1: 添加书籍 输入2: 添加学生信息 输入3: 查询书籍
输入4: 显示学生信息 输入5: 借阅书籍 输入6: 归还书籍
输入7来退出
6
输入要归还的书籍书号: 100
输入你的学号: 1710210119
还书成功
还需要继续还书吗? 若是, 输入非0, 若不是, 输入0
```

book.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
100 数据结构 张乃孝 72 284

student.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
1710210119 瞿晟 男 计院 0 0 0 0

若输入的图书书号不存在，或学生学号不存在，或学生未借此书，都会显示错误。

#### 4.1.8 退出

在欢迎界面输入 7，则可退出系统。

#### 4.1.9 程序分析

采用两个文件（book.txt 和 student.txt 文件来分别存储图书信息和学生信息），因 C 语言无法修改文件内容，因此遇到修改操作时，先创建一个伪文件来存储修改后的内容，再将原文件删除，将伪文件命名为原文件，这使得空间占用率较高，开辟了不少新的空间，在一开始着手时应避免使用文本文件来存储信息。

程序中数据结构都为简单的线性表结构，因内容皆为无序排列，故遍历时采用顺序遍历，时间复杂度为  $O(n)$ 。

开发过程中遇到的问题：

1. 采用何种方式存储数据？

解决方法：虽然以文本文件存储方式有上述诸多不利，但因其显示方便，简单易操作，故选择文本文件方式存储。

2. C 语言无法修改文件内容

解决方法：采用新建另一个文本文件存储。

算法的改进思想：可换一种方式存储，如单链表或二叉树，减少空间占用，存储时以书号或学号有序排序，以方便使用二分法查找，减小时间复杂度。

### 4.2 哈夫曼编码

#### 4.2.1 欢迎界面

显示：

已将 file.txt 的字符及权值写入 character.txt

已将 character.txt 字符权值写入哈夫曼结点中

已创建哈夫曼树

已创建哈夫曼编码，保存在 code.txt 中

显示“哈夫曼编码与解码”和小组成员名字。

显示：

在 character.txt 文件中存放着各个字符的权值

程序从中读出各个字符的权值构造哈夫曼树并进行编码


各个字符的编码存在 code.txt 文件中



显示操作的三种选择：编码、译码、退出。

```
已将file.txt的字符及权值写入character.txt  
已将character.txt字符权值写入哈夫曼结点中  
已创建哈夫曼树  
已创建哈夫曼编码，保存在code.txt中  
/*****哈夫曼编码与解码*****/  
  
          *  
      *           *  
      *           *  
      *           *  
          *  
      *               版权所有   _____瞿晟        *  
      *               _____王鑫            *  
      *               _____夏洋            *  
      *               _____向景荣         *  
      *               _____刘雅婷         *  
          *****/  
在character.txt 文件中存放着各个字母的权值  
程序从中读出各个字母的权值构造哈夫曼树并进行编码  
各个字符的编码存在code.txt文件中  
哈夫曼编码的压缩比为：35.7%  
/*****/  
  
请输入你的选择：1 ---- 编码    2 ---- 译码    0 ---- 退出
```

打开 character.txt 文件，程序已将 file.txt 文件中字符权值保存在其中。

 character.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

---

28  
,3  
.3  
G1  
l1  
N1  
a10  
b3  
c6  
d4  
e31  
g2  
h6  
i7  
l4  
n9  
o4  
p4  
r8  
s14  
t6  
u5  
v7  
w3  
x1  
y5

#### 4. 2. 3 哈夫曼编码

打开 code. txt 文件，各个字符的哈夫曼编码保存在其中。

code.txt - 记事本  
文件(F) 编辑(E) 格式(O) 窗口(W) 帮助(H)

```
110
, 100001
. 101000
G 10000011
I 0100110
N 0100111
a 0110
b 101001
c 10001
d 111011
e 00
g 010010
h 10010
i 10101
l 111110
n 0101
o 111111
p 01000
r 11110
s 1011
t 10011
u 01110
v 11100
w 111010
x 1000000
y 01111
```

#### 4.2.4 用户输入字符

在欢迎界面输入 1，则可输入一串字符，并求其哈夫曼编码。

```
/******  
请输入你的选择: 1 ---- 编码 2 ---- 译码 0 ---- 退出  
1  
请输入要编码的字符串(length<50)
```

输入一串字符，如：because it is。以回车结束输入。

得到其编码为：10100100100010110011101011001101010110011110101011011

```

/*****/

请输入你的选择: 1 ---- 编码  2 ---- 译码  0 ---- 退出
1

请输入要编码的字符串 (length<50)
because it is
输入字符串的编码为:      10100100100010110011101011001101010110011110101011011
还要继续操作吗?

请输入你的选择: 1 ---- 编码  2 ---- 译码  0 ---- 退出

```

#### 4.2.5 用户输入解码字符

在欢迎界面输入 2，则可输入一串二进制数字，进行哈夫曼译码成字符串。

```

已将file.txt的字符及权值写入character.txt
已将character.txt字符权值写入哈夫曼结点中
已创建哈夫曼树
已创建哈夫曼编码，保存在code.txt中
/*****哈夫曼编码与解码*****/

      *
*      *
*      *
*      *
*      *      版权所有      瞿晟      *
*      *      王鑫      *
*      *      夏洋      *
*      *      向景荣      *
*      *      刘雅婷      *
      *
      *****/

在character.txt 文件中存放着各个字母的权值
程序从中读出各个字母的权值构造哈夫曼树并进行编码
各个字符的编码存在code.txt文件中
哈夫曼编码的压缩比为: 35.7%
/*****/

请输入你的选择: 1 ---- 编码  2 ---- 译码  0 ---- 退出
2

请输入要译码的字串 (用0和1表示)

```

输入一串二进制数字，如：10100100100010110011101011001101010110011110101011011。  
得到其哈夫曼译码为：because it is。

```

/*****
请输入你的选择: 1 ---- 编码  2 ---- 译码  0 ---- 退出
2

请输入要译码的字串(用0和1表示)
10100100100010110011101011001101010110011110101011011
输入的编码的译码为: because it is
还要继续操作吗?

请输入你的选择: 1 ---- 编码  2 ---- 译码  0 ---- 退出

```

#### 4.2.6 退出

在欢迎界面输入 0，则可退出。

#### 4.2.7 程序分析

以三个文件存储文本信息、权值信息和编码信息 (file.txt, character.txt, code.txt)，因此项目无需修改文本文件中内容，故采用文本文件方式存储。哈夫曼树存放在一个数组中，哈夫曼编码既存储于文本文件 code.txt 中方便查看，又存储于一个数组中方便读取。

因哈夫曼树为二叉树，故除第一次读入字符、初始化哈夫曼树、创建哈夫曼树的遍历时间复杂度为  $O(n)$ ，其余算法时间复杂度皆为  $O(\log_2^n)$ 。

开发过程中遇到的问题：

1. 如何计算每个字符权值？

解决方法：原本计划未每个字符都记录权值（包括文本文件中不存在的字符），但这样做空间占用过大，且会有很多生僻字符也被存储，故选择只记录出现过的字符的权值。

算法的改进思想：

可采用顺序表来存储哈夫曼结点的字符及权值，方便读取，编码无需放在文本文件中，减少了空间占用。

## 五. 课程设计总结

### 5.1 项目分工

图书管理系统：

瞿晟：设计所需的各种函数及算法，论文的书写，并主要负责 1. 存储结构、2. 采编入库、

## 8. 借阅书籍的算法及主函数

王鑫：负责进行项目的需求规划，不断优化各成员的算法，并主要负责 4. 按书号查询图书信息、5. 按书名查询图书信息、6. 按作者查询图书信息

夏洋：记录各阶段小组的进度及遇到的问题，解决并提出更优方案，并主要负责 3. 添加学生、7. 显示学生信息

刘雅婷：详细设计各种算法，并主要负责 2. 采编入库、8. 借阅书籍、9. 归还书籍

向景荣：项目的调试分析

## 哈夫曼编码：

瞿晟：设计所需的各种函数及算法，论文的书写，并主要负责 1. 存储结构、6. 哈夫曼译码、9. 计算压缩比的算法及主函数

王鑫：负责进行项目的需求规划，不断优化各成员的算法，并主要负责 4. 构造哈夫曼树、5. 哈夫曼编码

夏洋：记录各阶段小组的进度及遇到的问题，解决并提出更优方案，并主要负责 2. 计算字符的权值、7. 用户输入字符

刘雅婷：详细设计各种算法，并主要负责 3. 初始化哈夫曼树、8. 用户输入解码子串、9. 计算压缩比

向景荣：项目的调试分析

## 5.2 心得体会

数据结构是计算机程序设计的重要理论技术基础，它不仅是计算机科学的核心课程，而且已经成为其他理工专业的热门选修课。随着高级语言的发展，数据结构在计算机的研究和应用中已展现出强大的生命力，它兼顾了诸多高级语言的特点，是一种典型的结构化程序设计语言，它处理能力强，使用灵活方便，应用面广，具有良好的可移植性。

紧张的几周数据结构实践很快就过去了，通过这几周的实践学习，不仅使我们巩固了以前的知识并在此基础上还对数据结构的特点和算法有了更深的了解，使我们在这门课程的实际应用上也有了一个提高。

首先这几周的学习，使我们在巩固了原有的理论知识上，又培养了灵活运用和组合集成所学过知识及技能来分析、解决实际问题的能力，使我们体会到自身知识和能力在实际中的应用和发挥。其次，它激发了我们创新意识，开发创造的能力和培养沟通能力。另外，让我们进一步熟悉了数据结构的设计应用。每一处编码都是在反复的熟悉数据结构的结构特性，及其语法、函数和程序设计思想的过程，对我们数据结构的学习和提高很有益处，并且使我们明白了程序设计过程，如解决一些实际问题，从解决实际问题的角度，我们可以这样来看：第一要了解这个问题的基本要求，即输入、输出、完成从输入到输出的要求是什么；第二，从问题的要害入手，从前到后的解决问题的每个方面，即从输入开始入手，着重考虑如何从输入导出输出，在这个过程中，可确定所需的数据结构的基本类型——线性表、栈、队列、串、数组、广义表、树和二叉树以及图等，然后确定处理过程——算法，通过在编译环境中的编译与调试，可到最终的程序。最后，在这次的实践过程中，我们深刻的认识到了自己在学习方面的不足之处，我知道我还有太多的基本的思想没有真正的理解，当然我们不会灰心，我们会在以后的日子里努力弥补我们的不足。

在这几周的实训中，我们也体会到了团队合作的重要性，从最初的查阅资料到最后的程序的成功运行，我们组有过山穷水尽的困惑；有过柳暗花明的惊喜；有过唇枪舌剑的辩论；有过相互鼓励的安慰。两个礼拜的时间我们经历了很多，也收获了很多。与其说这次的实训是体力与脑力的作业，不如说它是合作精神和毅力的考验。经过这次课程设计，我们不仅学到了很多知识和技能，更重要的是我们学会了如何运用所学知识去解决实际问题。

在本课程设计中，我明白了理论与实际应用相结合的重要性，并提高了自己组织数据及编写大型程序的能力。培养了基本的、良好的程序设计技能以及合作能力。这次课程设计同样提高了我的综合运用所学知识的能力。并对 C 语言有了更深入的了解。《数据结构》是一门实践性很强的课程，上机实习是对学生全面综合素质进行训练的一种最基本的方法，是与课堂听讲、自学和练习相辅相成的、必不可少的一个教学环节。上机实习一方面能使书本上的知识变“活”，起到深化理解和灵活掌握教学内容的目的；另一方面，上机实习是对学生软件设计的综合能力的训练，包括问题分析，总体结构设计，程序设计基本技能和技巧的训练。此外，还有更重要的一点是：机器是比任何教师更严厉的检查者。因此，在“数据结构”的学习过程中，必须严格按照老师的要求，主动地、积极地、认真地做好每一个实验，以不断提高自己的编程能力与专业素质。

总之，这次课程设计让我们受益匪浅。我们深深认识到，要学好一门学科，没有刻苦钻研的精神是不行的，只有在不断的尝试中，经历失败，从失败中总结经验，然后再不断的尝试，才能获得成功。