

数据结构和算法应用 大作业

姓名：章星宇

学号：19200300029

基于哈夫曼编码的压缩和解压实现

一、程序功能

根据文件中各字符出现的频率情况创建哈夫曼树，再将各字符对应的哈夫曼编码写入文件中，实现文件压缩，并通过逆向方式实现文件解压。

二、涉及知识点

- 哈夫曼树
- 哈夫曼编码
- 队列
- 文件操作

三、设备与环境

微型计算机、Windows10 操作系统、Dev-C++5.11 软件

四、概要设计

4.1 数据结构类型定义

1、哈夫曼树

```
typedef struct  
  
{  
  
    datatype data;      //存字符  
  
    WeightType weight; // 用来存放各个结点的权值 //  
  
    int parent, LChild, RChild; //指向双亲、孩子结点的指针 //  
  
} hufmtree;
```

2、哈夫曼编码表

```
typedef struct
```

```

{
    char bits[N];    //存储编码位串

    int start;      //指示位串在 bits 中的起始位置

    datatype data;  //存储字符

}codetype;

```

3、队列

```

typedef struct
{
    int tag;

    int front;

    int rear;

    datatype length;

    char elem[Maxsize];

} SeqQueue;

```

4.2 程序构成

本程序的构成，共有 14 个函数，1 个主函数。

1、队列初始化

```
int InitQueue(SeqQueue * Q)
```

2、入队操作

```
int In_seqQueue(SeqQueue * Q, char x)
```

3、出队操作

```
int Out_Queue(SeqQueue * Q, char *x)
```

4、创建哈夫曼树

```
hufmtree *CreatHFM(FILE * fp, short *n, WeightType * FileLength)
```

5、选择两个 parent 为 0 且 weight 最小的结点

```
void SelectMinTree(hufmtree * ht, int n, int *k)
```

6、对哈夫曼树排序

```
int SortTree(hufmtree * ht)
```

7、求哈夫曼 0-1 字符编码表

```
char **CrtHuffmanCode(hufmtree * ht, short LeafNum)
```

8、从队列里取 8 个字符 (0、1)， 转换成一个字节

```
datatype GetBits(SeqQueue * Q)
```

9、求最长（最短） 编码长度

```
void MaxMinLength(FILE * File, hufmtree * ht, char **hc, short NLeaf, datatype *  
Max,datatype * Min)
```

10、把出现过的字符编码表经过压缩写进文件

```
short CodeToFile(FILE * fp, char **hc, short n, SeqQueue * Q, datatype * length)
```

11、把读出的字符， 转换成 8 个 0、1 字符并入队

```
void ToQueue(SeqQueue * Q, datatype ch)
```

12、压缩

```
void Compress()
```

13、解压缩

```
void UnCompress()
```

14、主函数

```
int main()
```

五、详细设计

5.1 文件操作

文件打开：fopen(desFile, "rb");//以二进制形式打开文件

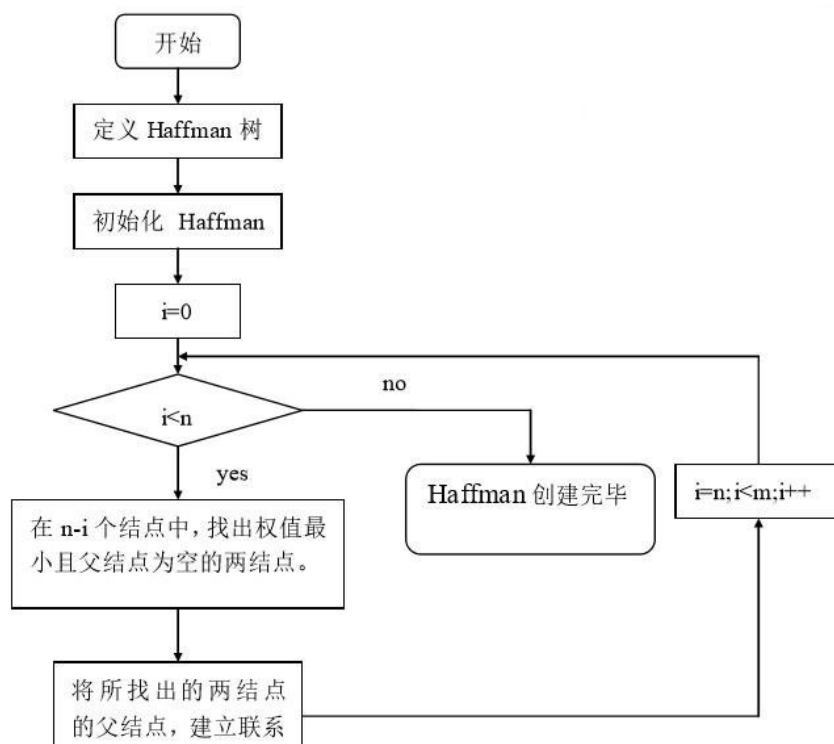
文件写入：fopen(rename, "wb");//以二进制形式写入文件

文件内容读取：fseek(FILE *stream, long offset, int fromwhere);//重定位文件内部的指针

文件内容写入：fseek(FILE *stream, long offset, int fromwhere);//写入若干数据块

5.2 构造哈夫曼树步骤

- 1、在森林中选取两棵根结点权值最小的树作左右子树，构造一棵新的二叉树，置新二叉树根结点权值为其左右子树根结点权值之和
- 2、在森林中删除这两棵树，同时将新得到的二叉树加入森林中
- 3、重复上述两步，直到只含一棵树为止，这棵树即哈夫曼树。



5.3 哈夫曼编码的构建

思想：根据字符出现频率编码，使电文总长最短

编码：根据字符出现频率构造 Huffman 树，然后将树中结点引向其左孩子的分支标“0”，引向其右孩子的分支标“1”；每个字符的编码即为从根到每个叶子的路径上得到的 0、1 序列

5.4 文件压缩的基本步骤：

- 1、打开需压缩文件
 - 2、遍历文件，根据文件字符出现的次数赋予权值，以此创建哈夫曼树，并统计原文件串长
 - 3、根据哈夫曼树创建哈夫曼编码表
 - 4、创建并初始化队列
 - 5、根据哈夫曼编码表将压缩后的数据写入文件
 - 6、压缩文件中的每一位（bit）入队，每 8 位出队，转换成一个字节，统计压缩文件串长
 - 7、计算压缩比（压缩后文件串长/原文件串长*100%）
- 最后，通过解压缩来验证是否失真现象。

5.5 程序运行界面的优化

改变一般程序黑底白字的界面，采用 `system("color f5");`，将程序界面改成淡紫色字体和浅灰色背景，优化使用体验

六、测试结果及分析

压缩文件运行过程截图：

欢迎使用哈夫曼压缩器

1. 压缩
 2. 解压缩
 3. 退出
-

请选择 (1 to 3):1

```
filename to be compressed:input.txt
filename after compressed:output.txt
103号    g 码长: 4;编码:0101
 97号    a 码长: 5;编码:01100
 98号    b 码长: 5;编码:01101
 99号    c 码长: 5;编码:01110
100号    d 码长: 5;编码:01111
101号    e 码长: 5;编码:10000
102号    f 码长: 5;编码:10001
104号    h 码长: 5;编码:10010
105号    i 码长: 5;编码:10011
107号    k 码长: 5;编码:10100
108号    l 码长: 5;编码:10101
109号    m 码长: 5;编码:10110
110号    n 码长: 5;编码:10111
111号    o 码长: 5;编码:11000
112号    n 码长: 5;编码:11001
```

解压缩文件运行过程截图:













欢迎使用哈夫曼压缩器

1. 压缩
 2. 解压缩
 3. 退出
-

请选择 (1 to 3):2

```
filename to be uncompressed:output.txt
filename after uncompressed:output2.txt
Please wait a minute,uncompressing...
Press Enter to continue...
```

对各类格式文件的压缩效果：

 解压后.txt	174 KB
 压缩后.txt	130 KB
 压缩前.txt	174 KB
 解压后.doc	68 KB
 压缩后.doc	32 KB
 压缩前.doc	68 KB
 压缩前.bmp	1,688 KB
 压缩后.bmp	229 KB
 解压后.bmp	1,688 KB
 解压后.mp4	6,028 KB
 压缩后.mp4	5,993 KB
 压缩前.mp4	6,028 KB

分析：文件解压后和压缩前一致，可见压缩文件并没有失真，但压缩效果时好时坏。实质上，哈夫曼编码的压缩效率取决于文件中相同字符出现的频度，在图片中，也就是相同区域的“冗余”，当冗余越多，压缩效果就越好。但这个程序对视频压缩的支持并不理想，还有改进空间。