**电子科技大学**

**计算机科学与工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 软件开发综合实验**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

# 实 验 报 告

**学生姓名： 学 号： 指导教师：**

**实验地点： 实验时间：**

**一、实验室名称：**

**二、实验项目名称：哈夫曼编码**

**三、实验学时：8学时**

**四、实验原理：**

1.运用哈夫曼编码算法，将原8bit一个字符的编码转化为根据文章内容不同而不同的编码，主要原理是使出现频率高的字符用较短的编码，出现频率低的字符用较长编码，同时保证编码属于前缀码，这样就可以减小空间，达到压缩的目的。

2.运用文件操作读取和写入文件，以实现对文件的压缩和解压操作。

3.运用字符串和二进制的相关知识，做到二进制数和字符串的转化功能。

**五、实验目的：**

1.熟悉基本数据结构（数组等）的使用。

2.掌握字符串函数、内存分配、文件操作等函数功能。

3.掌握建立哈夫曼树和哈夫曼编码的算法和原理。

4.学会利用数据结构和算法编写简单程序。

**六、实验内容：**

打开一个txt文档，读取文档中内容，统计出现的字符及其频率，创建huffman树，并对文本内容进行huffman编码，对文件压缩。保存压缩文件和huffman编码，对压缩后的文件解压缩。要求如下：

1.获取文件内容，统计字符及对应出现词频。

2.构建哈夫曼树，对字符进行编码

3.根据字符编码进行文件压缩，写入二进制文件

4.打开二进制文件，根据表头信息，将二进制编码转换为对应字符，写入文本文件

1. **实验步骤：**
2. 先定义两个结构体，一个是哈弗曼树，用于建立和生成哈弗曼树以及哈夫曼编码，另一个是信息表，用于记录每一个字符的频率（权重）、哈夫曼编码等信息。部分代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct FreqCountTable //定义频数存储表结构用于存储统计的频数

{

char asciicode; //储存对应ascii码

int freq; //储存其对应的频数

char huffmancode[15]; //储存编码后的哈夫曼码

int flag; //1表示有元素 0表示无元素

}FreqCountTable;

typedef struct HuffmanTree

{

unsigned int weight; //定义哈夫曼树结构体，用数组存，包含权值，左右孩子和父亲的下标

unsigned int parent; //按照此方法定义的哈弗曼树结构，不需要复杂的构成，可以任意查找与当前节点相关的节点

unsigned int LChild

unsigned int RChild;

}HuffmanTree;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 接下来定义函数操作，重要的操作包括扫描文件并统计信息操作，建立哈弗曼树和哈弗曼编码操作，字符转化操作，压缩操作和解码操作等，部分函数头代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void select(HuffmanTree \*ht,int n, int \*s1, int \*s2) //哈夫曼生成算法操作

void InitialHuffmanTree(HuffmanTree \*ht , FreqCountTable \*w, int n, int m) //初始化哈弗曼树

void CreateHuffmanCode(HuffmanTree \*ht, FreqCountTable \*hc, int n) //建立哈夫曼编码表 传入一个树 一个原编码表 一个叶子数n

void OutputHuffman(HuffmanTree \*HT, int m) //输出哈夫曼树中的元素（打印出来） 中序打印 刚好是完全二叉树的存储顺序

void InitialFreq(FreqCountTable \*pF) //信息表初始化操作

int SearchFreq(char a, FreqCountTable \* pF) //Freq表搜索操作，如果查到有该字符，则返回下标，如果没有，则返回-1

void FreqCount(FILE \* pFILE, FreqCountTable \*pF) //读取原文件并统计词的种类和词频操作

int LeavesNum(FreqCountTable \* pf) //统计叶子节点数

int CodeLength(FreqCountTable \* pc, int i) //计算并返回当前编码长度（即路径长度）

int CodeWPL(FreqCountTable \* pc,int \* WPL) //计算并返回当前哈弗曼树的带权路径长度 且自动加满八位

int RealCode(FreqCountTable \* pc) //计算并返回码表的长度

void GenerateString(FILE \* pold, HuffmanTree \* pt, FreqCountTable \* pc, char \* a, char \* b, int delta, int \*count) //自动把密文补零

void OutputFILE(FILE \* pold, FILE \* pnew, HuffmanTree \* pt, FreqCountTable \* pc) //码表文件

void Output(int MAXB, char \*b, FILE \* pnew) //每八位返回给C并输出fputc到新文件 //输出密文文件

void ReadHead(FreqCountTable \* pFF, FILE \*phead) //解压时读取码表操作

void Convert(unsigned char a, char b[]) //把读到的二进制数 转化为8位字符串

void Decode(FILE \*pcode, FILE \*pdecode, FreqCountTable \* pFF, int \*count) //解码操作

void OutputFreq(FreqCountTable \* pF) //打印信息表操作

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 设置原始文件：Data.txt，大小为8.94KB，通过压缩和解码操作得到压缩文件Code.txt，和解码文件Decode.txt，大小分别为4.88KB和8.94KB，成功达到了压缩的目的，且压缩率为183%左右。如下是一些文件和程序的截图：

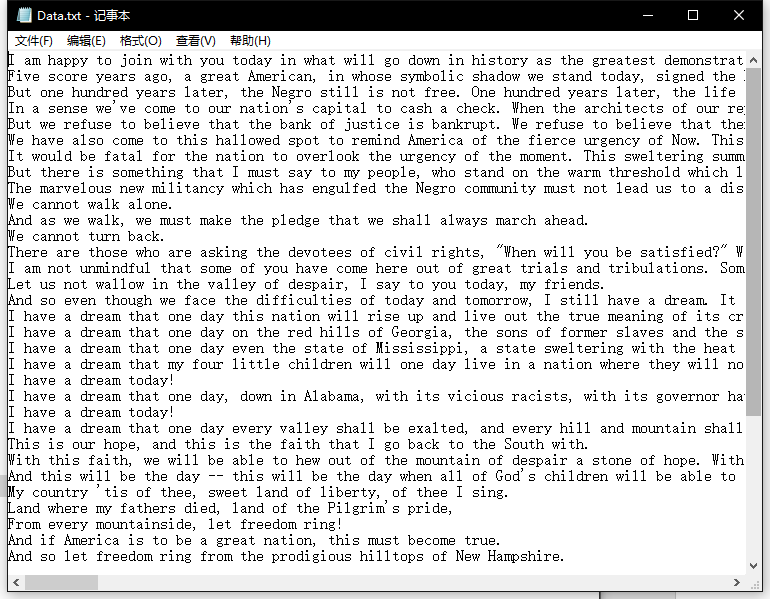


图1.Data.txt文件

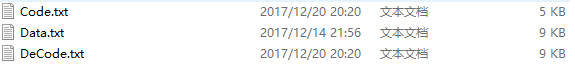


图2.三个文件及其大小



图3.建立哈夫曼树后的部分编码表

**八、总结及心得体会：**

通过本实验的练习，学会了哈夫曼树以及哈夫曼编码算法的原理，深入理解了算法在实际程序中的具体应用与实现。并且学会了一些基本的文件操作，比如文件打开与关闭，读写，文件指针移位，系统删除文件等操作。

**九、对本实验过程及方法、手段的改进建议：**

可以类比对文本文件压缩的操作，扩展到对个各种类型文件的压缩过程，做成真正实用的压缩软件。

**报告评分：**

**指导教师签字：**