**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：覃翊 学 号： 2016060107008 指导教师：王丽杰**

**实验地点： 主楼B412 实验时间：12.7-12.21**

**一、实验室名称： 计算机学院实验中心A2-412**

**二、实验项目名称：文本压缩软件**

**三、实验学时：12学时**

**四、实验原理：**

**任何文件都可以看作是由字节组成的字节块，将字节看作基本编码单元，一个文件就可以看作是由字节组成的信息串。对文件中各字节的出现频率进行统计，并以出现频率作为每个字节块的权值，就可以用字节为叶结点构造哈夫曼树，进而构造出各字节的对应哈夫曼编码。**

**对于源文件的字节编码是一种8位定长编码，而将各字节用哈夫曼编码进行重新编码，就有可能使得总的编码长度更短，这样将编码看做二进制位，每个8位作为一个字节写入压缩文件中，根据构建哈弗曼树的算法，在源文件中出现频率高的字符所对应的编码长度小，所以总的文件长度会小于原来的文件长度，从而达到文件压缩的效果。**

**五、实验目的：**

**了解哈夫曼树，增强动手能力，编程能力。学会用C语言实现一个文本压缩软件，了解树的存储与操作。理解二叉树的表示与存储；二叉树的基本操作。掌握哈夫曼算法；文件操作。**

**六、实验内容：**

**设计并实现一个基于哈夫曼树编码算法的英文文本编/解码器，其由编码器（compressor）对文件进行压缩编码，由解码器（decompressor）进行解码还原。**

1. **实验步骤：**

  创建一个256长度的数组，来记录每一个ASCII码值出现的频率。

将要压缩的文件遍历一遍，统计一下每一个字符出现的频率。

然后调用求Huffman树的函数，建立一棵对应此文件字符出现的频率的Huffman树。

建立哈夫曼树的步骤：

第1步：根据给定的n个权值{w1,w2,……wn}，构造n棵只有根结点的二叉树，令其权值为wj；

第2步：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

第3步：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

第4步：重复2-3两步，直到只含一棵树为止，这棵树即哈夫曼树

求出对应每一个ASCII码的Huffamn编码。

然后再遍历一遍，每次读出一个字节，根据其ASCII码值找出对应的Huffman编码，如果Huffman编码长度不足8位时，就再从文件中读出下一个字符，并且得到他的Huffman编码，与之前的没有写入文件的编码作为一个整体；

如果此时的编码长度大于或等于8了，就将它对应的ASCII码值写入压缩文件中。

就这样边读源文件，边进行翻译写入压缩文件，直到读完源文件。

这样就完成了文件的压缩功能。

还要解压文件，所以在进行压缩的时候还应该把Huffman树一块保存起来，所以压缩文件的前面一部分就是压缩文件是建立的Huffman树，而压缩文件的开头四个字节用来记录文件的扩展名。

解压的时候，根据压缩文件的格式，读出源文件的扩展名，然后读出Huffman树，根据Huffman树进行解压。

解压时，同样可以一边解压一边写入解压后的文件中，即每读出一个字节，就根据他的ASCII码值，查找Huffman树对应的叶子节点，叶子节点的下标就是源文件对应的ASCII码值，然后将得到的ASCII码值写入压缩后的文件中，就这样直到读完解压文件，就完成了文件的解压工作。

分块代码如下：

建立哈夫曼树的节点结构：

typedef int ElemType;

typedef struct BTreeNode

{

ElemType data;

ElemType data2;

struct BTreeNode\* left;

struct BTreeNode\* right;

}BTreeNode;

创建哈夫曼树：

BTreeNode\* CreateHuffman(int a[][2], int n)

{

int i, j;

BTreeNode \*\*b, \*q=NULL;

b = (BTreeNode \*\*)malloc(n\*sizeof(BTreeNode));

for (i = 0; i < n; i++)

{

b[i] = (BTreeNode \*)malloc(sizeof(BTreeNode));

b[i]->data = a[i][1];

b[i]->data2 = a[i][0];

b[i]->left = b[i]->right = NULL;

}

for (i = 1; i < n; i++)

{

//k1表示森林中具有最小权值的树根结点的下标，k2为次最小的下标

int k1 = -1, k2;

for (j = 0; j < n; j++) //让k1初始指向森林中第一棵树，k2指向第二棵

{

if (b[j] != NULL && k1 == -1)

{

k1 = j;

continue;

}

if (b[j] != NULL)

{

k2 = j;

break;

}

}

for (j = k2; j < n; j++) //构造最优解

{

if (b[j] != NULL)

{

if (b[j]->data < b[k1]->data)

{

k2 = k1;

k1 = j;

}

else if (b[j]->data < b[k2]->data)

k2 = j;

}

}

q = (BTreeNode \*)malloc(sizeof(BTreeNode));

q->data = b[k1]->data + b[k2]->data;

q->left = b[k1];

q->right = b[k2];

b[k1] = q;

b[k2] = NULL;

}

free(b);

return q;

}

解压函数：

BTreeNode\* unzip(BTreeNode \*node,char flag){

if(flag == 0)

return node->left;

if(flag == 1)

return node->right;

}

哈夫曼编码过程：

void HuffManCoding(BTreeNode\* FBT, int len,char HuffNode[][20])

{

static char a[20];

if (FBT != NULL)

{

if (FBT->left == NULL && FBT->right == NULL)

{

int i;

HuffNode[qj][0]=FBT->data2;

for(i=1;i<len+1;i++)

HuffNode[qj][i]= a[i-1];

HuffNode[qj][len+1]='|';

printf("ASCII码为%d的编码：",HuffNode[qj][0]);

for (i = 1; i < len+1; i++)

printf("%d", HuffNode[qj][i]);

printf("\n");

qj++;

}

else

{

a[len] = (00000000);

HuffManCoding(FBT->left, len + 1,HuffNode);

a[len] = (00000001);

HuffManCoding(FBT->right, len + 1,HuffNode);

}

}

}

Main函数：

int main(int argc, char \*argv[])

{

printf("-------------文本压缩软件-------------\n\t\t\t\t制作者：覃翊 2016060107008\n");

int data[256][2];

for(int i=0;i<256;i++)

for(int j=0;j<2;j++)

data[i][j]=0;

int n=0;int i=0;

char m;

freopen("data.txt","rb",stdin);

while(scanf("%c",&m)!=EOF){

int biaoji=0;

m=(int)m;

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

{

if(data[j][0]==m)

{

data[j][1]=data[j][1]+1;biaoji=1;break;

}

}

if(biaoji==0)

{

data[n][0]=m;data[n][1]=1;n++;

}

}

/\*

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

for(int j=0;j<2;j++)

printf("%d ",data[i][j]);

printf("\n");

\*/

fclose(stdin);

BTreeNode\* hfm;

hfm=CreateHuffman(data,n);

char HuffNode[256][20];

HuffManCoding(hfm,0,HuffNode);

/\*

for(int i=0;i<=25;i++)

for(int j=0;HuffNode[i][j]!='|';j++)

printf("%c ",HuffNode[i][j]);\*/

int length=0;

int length1=0;

FILE \*fpr = fopen("data.txt","rb");

FILE \*fpw = fopen("zip.txt","wb");

char zifu;

unsigned char savechar=0;

int num=0;

while((zifu=fgetc(fpr))!=EOF){

//printf("%c",zifu);

length++;

for(i=0;i<256;i++){ //变成int i=0后出现bug

if(zifu==HuffNode[i][0]) break;}

//printf("%d ",i);

for(int j=1;HuffNode[i][j]!='|';j++){

savechar=savechar+HuffNode[i][j];

//printf("%d ",savechar);

num++;

if(num==8){

fwrite(&savechar,sizeof(char),1,fpw);

savechar=0;

num=0;

length1++;

//printf("OK");

}

else{

savechar=savechar<<1;

}

}

}

if(num != 8){

savechar = savechar<<(7-num);

fwrite(&savechar,sizeof(char),1,fpw);

length1++;

}

fclose(fpr);

fclose(fpw);

int counttime = 20;

printf("\n");

printf("正在压缩，请稍等\n压缩进度：");

while(counttime > 0){

printf("■");

Sleep(counttime\*10);

counttime--;

}

printf("\n");

printf("\n压缩完成！压缩文件名为：zip.txt\n");

printf("\n下面开始解压\n");

int changdu=0;

FILE \*fpr1= fopen("zip.txt","rb");

// fseek(fpr1,0L,SEEK\_SET);

FILE \*fpw1 = fopen("unzip.txt","w");

unsigned char op=(10000000);unsigned char s;

BTreeNode \*z=hfm;

int x=0;

while(x!=length1){

zifu=fgetc(fpr1);

x++;

for(int k=0;k<8;k++){

s=op&zifu;

s=s>>7;

zifu=zifu<<1;

z=unzip(z,s);

if(z->left == NULL || z->right == NULL){

fprintf(fpw1,"%c",z->data2);

z=hfm;

changdu++;

}

if(changdu==length)break;

}

}

// printf("%d %d",length,changdu);

fclose(fpr1);

fclose(fpw1);

counttime = 20;

printf("\n解压中，请稍等\n解压进度：");

while(counttime > 0){

printf("■");

Sleep(counttime\*10);

counttime--;

}

printf("\n");

printf("\n解压完成！请到文件中查看解压后内容\n");

Sleep(8000);

return 0;

}

此外，我还添加了一些亮点功能，对用户友好的，就是压缩和解压时的进度条功能。

int counttime = 20;

printf("\n");

printf("正在压缩，请稍等\n压缩进度：");

while(counttime > 0){

printf("■");

Sleep(counttime\*10);

counttime--;

}

printf("\n");

printf("\n压缩完成！压缩文件名为：zip.txt\n");

counttime = 20;

printf("\n解压中，请稍等\n解压进度：");

while(counttime > 0){

printf("■");

Sleep(counttime\*10);

counttime--;

}

printf("\n");

printf("\n解压完成！请到文件中查看解压后内容\n");

进度条的加载时间都是不确定的，是逐渐加快的，看着很舒服。

完整的代码我不再贴了，放在了我的github上，从上面也能看出我做过的修改，文件的历史记录。（https://github.com/qinyi0411/Software-experiment-course）

**八、总结及心得体会：**

通过本实验的练习，学会了文件的读取和写入操作，深入了解了哈夫曼编码，哈夫曼树的创建，在编写程序的过程中积累了大量丰富的经验，加强了自身实力，对编程的理解。

曾压缩过一个300M的txt文件，压缩了很长时间，对计算机也是一个挑战，最后也能成功的解压了文件，程序跑了很久，成功时非常有成就感。

**九、对本实验过程及方法、手段的改进建议：**

1. 学习图片是怎么存储的，然后改动本程序，做一个压缩图片的软件。

2.将本程序分开，分成压缩软件与解压软件，并在压缩文件的前面加上校验码，即用本压缩程序所压缩出来的文件，必须用对应的解压软件才能解开，如果读入校验码不对，解压软件就不能进行，达到了保密的效果，比如，两个人在两个不同的地方，一方有压缩软件，一方有对应的解压软件，他们可以通过软件来传输压缩后的文件，即使文件中途被劫持了，也读不出来，因为只有通过唯一的解压软件才能将其解压。

**报告评分：**

**指导教师签字：**