**《数据结构》**

**实验报告**

专 业 软件工程

年 级 2018级

姓 名 谢梓聪

学 号 201824100445

实 验 室 信工北414

使用日期 2019年11月21日

郑州大学信息工程学院

# 实验六 Huffman树及Huffman编码的算法实现

**一、实验目的及要求：**

熟悉掌握Huffman树的构造方法及Huffman编码的应用，了解Huffman树在通信、编码领域的应用过程。

**二、实验方法：**

1．使用windows平台。

2．使用DEV-C或Visual C软件。

**三、实验内容：**

（1）输入一段100—200字的英文短文，存入一文件a中。

（2）写函数统计短文出现的字母个数n及每个字母的出现次数

（3）写函数以字母出现次数作权值，建Haffman树（n个叶子），给出每个字母的Haffman编码。

（4）用每个字母编码对原短文进行编码，码文存入文件b中。

（5）用Haffman树对文件b中码文进行译码，结果存入文件c中，比较a,c是否一致，以检验编码、译码的正确性。

**四、实验结果：**

/\* HuffmanTree.cpp -- 英文短文哈夫曼编码以及解码 \*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

typedef char ElemType;

typedef struct{

ElemType data;

unsigned int weight;

unsigned int parent,lChild,rChild;

}HTNode,\*HuffmanTree; //动态分配数组存储赫夫曼树，因不知叶子结点个数

typedef char\*\* HuffmanCode; //动态分配数组存储赫夫曼编码，因不知叶子节结个数和每个结点的字符长度

void HuffmanCoding(HuffmanTree &HT,HuffmanCode &HC,int \*weigh,char \*map2char,int n); //weigh数组各元素要赋初值为0

void select(HuffmanTree &HT,int n,int &min1,int &min2);

void count(int \*weigh,int \*map2num,char \*map2char,int &num);

void PassageRead(void);

void PassageEncoding(HuffmanCode &HC,int \*map2num);

void CodeOutput(HuffmanTree HT,int num,HuffmanCode HC);

void PassageDecoding(HuffmanTree &HT,int m);

int main(void)

{

int weigh[260]={0},num=0,map2num[260]={0};

char map2char[260]={'\0'};

//map2num数组表示当前字符所对应的结点序号映射，用来通过直到当前字符找哈夫曼编码

//map2char数组表示当前序号对应的字符映射，用来为哈夫曼树叶子结点的data域赋值以便解码输出

HuffmanTree HT;

HuffmanCode HC;

PassageRead();

count(weigh,map2num,map2char,num);

HuffmanCoding(HT,HC,weigh,map2char,num); //产生哈夫曼树，进行哈夫曼编码

CodeOutput(HT,num,HC); //输出各字符对应的哈夫曼编码，空格和换行符不参加编码，其余字符均可参与编码

PassageEncoding(HC,map2num); //为英文短文进行编码

PassageDecoding(HT,2\*num-1);

//为产生的编码进行解码，解码思路为一个一个读字符，然后从根节点开始，

//读到0向左孩子走，读到1往右孩子走，直到叶子结点然后输出，

//然后重新从根结点开始执行相同步骤直至字符读完

return 0;

}

void PassageRead(void)

{

char ch;

FILE \*fpt=NULL;

printf("请输入英语短文，输入结束后按ENTER键，然后CTRL+Z和ENTER键结束文章输入\n");

printf("开始读取英语短文……\n");

fpt=fopen("Passage.txt","w");

while((ch=getchar())!=EOF)

{

fputc(ch,fpt);

}

fclose(fpt);

printf("已存入文件Passage.txt文件中\n");

}

void CodeOutput(HuffmanTree HT,int num,HuffmanCode HC)

{

int i=0;

printf("各字符哈夫曼编码为：\n");

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

{

printf("%c:\t%s\n",HT[i].data,HC[i]);

}

}

void PassageDecoding(HuffmanTree &HT,int m)

{

char ch;

int i,key,root;

FILE \*fptR,\*fptW;

printf("开始解码……\n");

fptR=fopen("PassageEnCoding.txt","r");

fptW=fopen("PassageDecoding.txt","w");

for(i=1;i<=m;i++) //找到根结点

{

if(HT[i].parent==0)

root=i;

}

key=root;

ch=fgetc(fptR);

while(!feof(fptR))

{

key=root; //重新从根结点开始

while((HT[key].lChild!=0||HT[key].rChild!=0)&&!feof(fptR))

{

if(ch=='0') //左孩子

{

key=HT[key].lChild;

}

else if(ch=='1') //右孩子

{

key=HT[key].rChild;

}

else //为空格和换行符时直接输出

{

fputc(ch,fptW);

}

ch=fgetc(fptR);

}

fputc(HT[key].data,fptW);

}

fclose(fptR);

fclose(fptW);

printf("解码已存入PassageDecoding.txt文件中\n");

printf("解码结束……\n");

}

void PassageEncoding(HuffmanCode &HC,int \*map2num)

{

FILE \*fptR,\*fptW;

char ch;

printf("开始编码……\n");

fptR=fopen("Passage.txt","r");

fptW=fopen("PassageEncoding.txt","w");

ch=fgetc(fptR);

while(!feof(fptR))

{

if(ch==' '||ch=='\n') //为空格和换行符直接输出

fputc(ch,fptW);

else

fprintf(fptW,"%s",HC[map2num[ch]]); //输出各字符对应的哈夫曼编码

ch=fgetc(fptR);

}

fclose(fptR);

fclose(fptW);

printf("编码已存入PassageEncoding.txt文件中\n");

printf("编码结束……\n");

}

void count(int \*weigh,int \*map2num,char \*map2char,int &num)

{

int i=0;

char ch;

num=0;

FILE \*fpt=fopen("Passage.txt","r");

ch=fgetc(fpt);

while(!feof(fpt))

{

if(ch!=' '&&ch!='\n') //不为空格和换行符编码

weigh[ch]++;

ch=fgetc(fpt);

}

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

{

if(weigh[i])

{

weigh[num]=weigh[i];

weigh[i]=0;

map2num[i]=num+1; //存字母-序号映射

map2char[num+1]=i; //存序号-字母映射

num++;

}

}

fclose(fpt);

}

void select(HuffmanTree &HT,int n,int &min1,int &min2)

{

int i,temp;

min1=min2=0;

//select函数基本思想：

//先从1-n中找到parent为0的两个结点，

//并且保证min1为当前遍历过的结点中最小的，min2为遍历过的结点中次小的

//则一直比较min1和之后每个结点的值，若比min1还小，则min1赋值为该结点的序号，

//min2赋值为min1的序号，最后结果即为min1最小，min2次小

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

{

if(HT[i].parent==0)

{

if(min1)

{

min2=i;

break;

}

min1=i;

}

}

if(HT[min1].weight>HT[min2].weight)

{

temp=min1;min1=min2;min2=temp;

}

for(i=(min1>min2?min1:min2)+1;i<=n;i++)

{

if(HT[i].parent==0)

{

if(HT[i].weight<=HT[min1].weight)

{

temp=min1;min1=i;min2=temp;

}

}

}

}

void HuffmanCoding(HuffmanTree &HT,HuffmanCode &HC,int \*weigh,char \*map2char,int n)

{

int m,i,min1,min2,pre,now,start;

HTNode \* p;

char \*tempCh; //临时的存储一个叶子节结的字符串的字符数组

if(n<=1) return;

m=2\*n-1; //总结点个数

HT=(HuffmanTree)malloc(sizeof(HTNode)\*(m+1)); //0号单元未用

for(p=HT+1,i=1;i<=n;i++,p++) //两部分结点分开初始化

{

p->data=map2char[i]; //存叶子结点的字母值

p->weight=weigh[i-1];

p->parent=p->lChild=p->rChild=0;

}

for(;i<=m;i++,p++)

{

p->weight=p->parent=p->lChild=p->rChild=0;

p->data='\0';

}

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

{

select(HT,i-1,min1,min2); //在树中从1到i-1个结点中找到parent为0的最小的两个结点

HT[min1].parent=HT[min2].parent=i;

HT[i].lChild=min1;

HT[i].rChild=min2;

HT[i].weight=HT[min1].weight+HT[min2].weight;

}

//开始根据哈夫曼树从叶子到根逆向求每个字符的哈夫曼编码

HC=(HuffmanCode)malloc(sizeof(char\*)\*(n+1));

tempCh=(char \*)malloc(sizeof(char)\*(n));

tempCh[n-1]='\0';

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

{

start=n-1;

for(pre=i,now=HT[i].parent;now!=0;pre=now,now=HT[now].parent)

{

if(HT[now].lChild==pre)

tempCh[--start]='0';

else

tempCh[--start]='1';

}

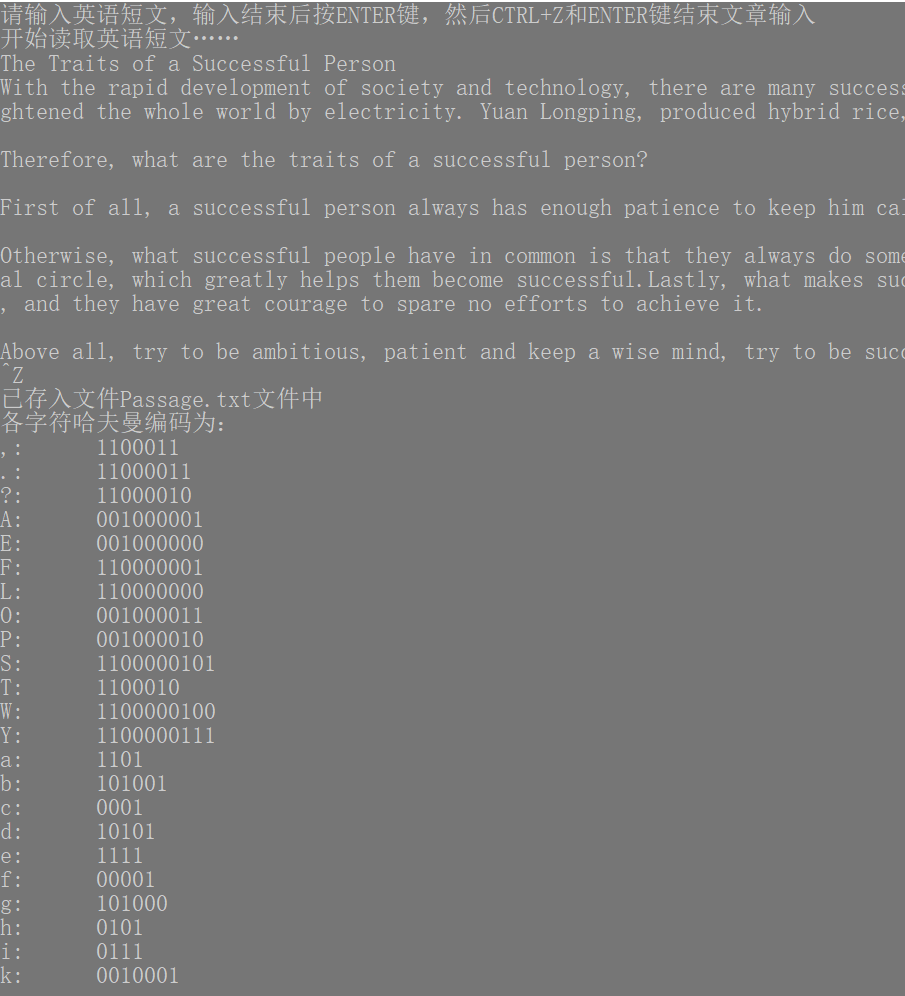
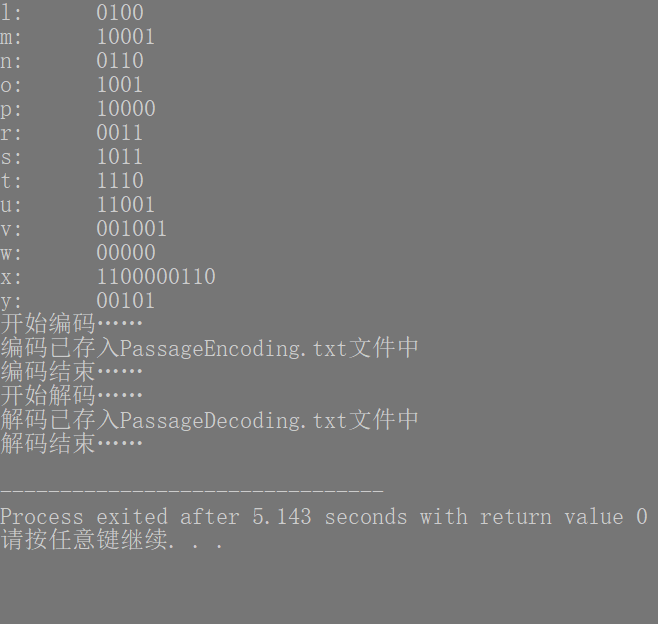
HC[i]=(char \*)malloc(sizeof(char)\*(n-start));

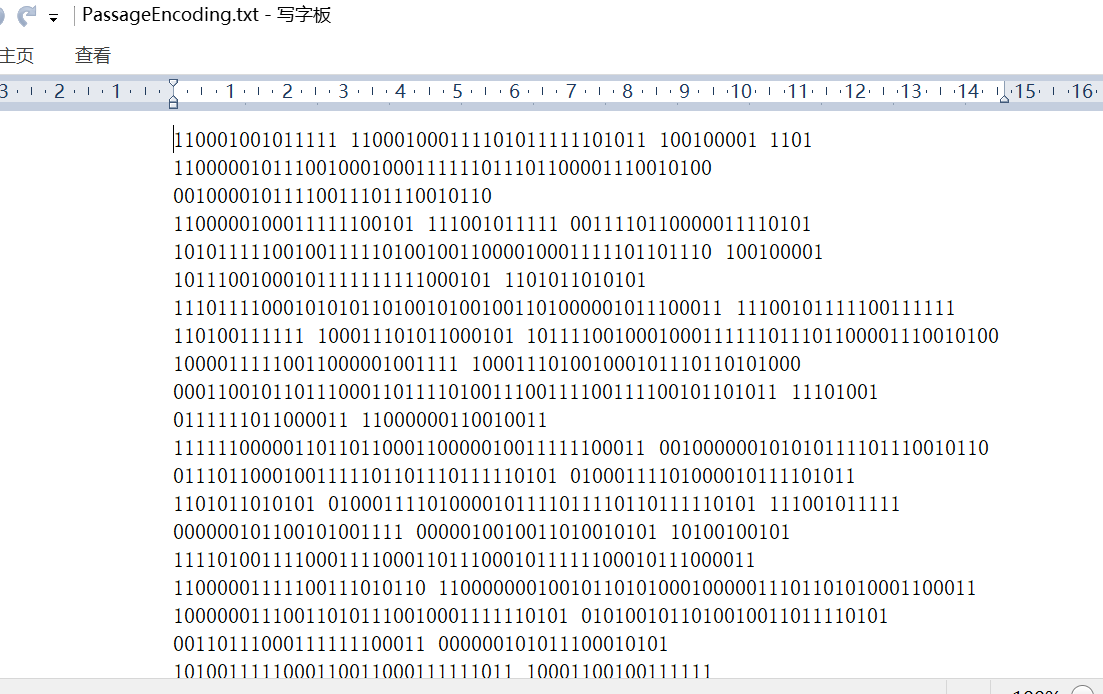
strcpy(HC[i],&tempCh[start]); //从第start个字符起进行赋值

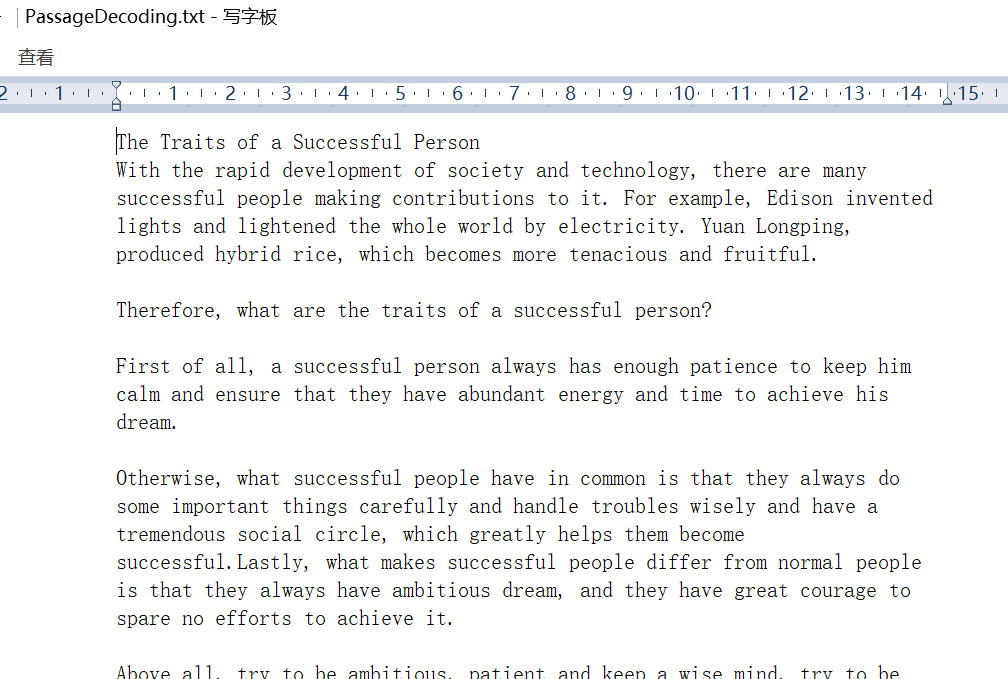
}

free(tempCh);

}







**五、实验体会：**

哈夫曼树为一种特殊的二叉树，也称最优二叉树，是一种带权路径长度最短的树，常用于判定二叉树的生成比如百分制转换为五级分制，和数据压缩存储，而哈夫曼编码即是根据哈夫曼树而对字符进行编码生成哈夫曼编码，原理是根据各字符出现的次数作为权重，从而对字符多的01编码位数少，字符多的01编码位数多，并且生成的是前缀编码便于解码，哈夫曼编码不像ASCII码那样固定8位二进制，当所有字符都出现一次时则会对空间浪费，因此通过哈夫曼编码能对数据进行压缩存储。本实验涉及到了文件上的操作，以英语短文的字符个数作为叶子结点个数，然后字符出现次数作为权重生成哈夫曼树和哈夫曼编码，然后再对短文遍历编码存到文件中，然后读取编码文件通过遍历哈夫曼树的方法进行解码，即从根结点开始，读到0则往左子树遍历，读到1则往右子树遍历，直到叶子结点输出，然后重复执行以上操作直至文件结束，从而得到解码文件。