# 哈夫曼树实验报告（因为文件没学周一未完成验收）

电信1809蔡俊哲 学号U201813485 2019/4/25

## 需求分析

在发射端通过编码系统对代传数据预先编码，在接收端将传来的数据进行译码，利用哈夫曼编码可以大大提高效率缩短时间降低成本。

## 概要设计

typedef struct HTff

{

char c;

int lchild;

int rchild;

int parent;

int weight;

}Node;

主程序流程：

1、初始化读入字符集大小，字符数和权值建立哈夫曼树。

2、编码。利用哈夫曼树对正文进行编码。

3、译码。利用哈夫曼树对代码进行译码

4、显示。将次字符型式的哈夫曼树显示在终端上

## 详细设计

#include<stdio.h>

#include<stdlib.h>

#define N\_LEAVE 26

#define N\_NODE (26\*2-1)

typedef struct HTff

{

char c;

int lchild;

int rchild;

int parent;

int weight;

}Node;

typedef struct HTffc

{

int HufCode[N\_LEAVE];

int Start;

char Char;

}Code;

void Huffman(Node Ht[],int k[])

{

int i,j,x1,x2;

int min1,min2;

for(i=0;i<N\_NODE;i++)

{

Ht[i].parent = -1;

Ht[i].lchild = -1;

Ht[i].rchild = -1;

if(i<N\_LEAVE)

{

Ht[i].weight = k[i];

Ht[i].c = i+65;

}

else

{

Ht[i].weight = 0;

Ht[i].c = '?';

}

}

for(i=1;i<=N\_LEAVE-1;i++)

{

min1 = min2 = 1000;

x1 = x2 = 0;

for(j=0;j<N\_LEAVE-1+i;j++)

{

if(Ht[j].parent == -1 && Ht[j].weight<min1 )

{

min2 = min1;

x2 = x1;

min1 = Ht[j].weight;

x1 = j;

}

else

{

if( Ht[j].parent == -1 && Ht[j].weight<min2 )

{

min2 = Ht[j].weight;

x2 = j;

}

}

}

Ht[x1].parent = N\_LEAVE-1+i;

Ht[x2].parent = N\_LEAVE-1+i;

Ht[N\_LEAVE-1+i].weight = Ht[x1].weight + Ht[x2].weight;

Ht[N\_LEAVE-1+i].lchild = x1;

Ht[N\_LEAVE-1+i].rchild = x2;

}

}

void Code\_Ht(Node Ht[],Code Hc[])

{

int i,d,p,j;

Code x;

for(i=0;i<N\_LEAVE;i++)

{

x.Char = Ht[i].c;

x.Start = N\_LEAVE-1;

d = i;

p = Ht[i].parent;

while(1)

{

if(Ht[p].lchild == d)

x.HufCode[x.Start] = 0;

else if(Ht[p].rchild == d)

x.HufCode[x.Start] = 1;

else

printf("ERROR!");

d = p;

p = Ht[d].parent;

if(p == -1) break;

x.Start--;

}

for(j=x.Start;j<=N\_LEAVE-1;j++)

{

Hc[i].HufCode[j] = x.HufCode[j];

}

Hc[i].Start = x.Start;

Hc[i].Char = x.Char;

}

}

void translate(Node Ht[],char a[])

{

int i,j;

j=0;

Node root;

Node temp;

for(i=0;i<N\_NODE;i++){

if(Ht[i].parent==-1) root=Ht[i];

}

temp=root;

while(a[j]!='\0'){

if(a[j]=='0') {temp=Ht[temp.lchild];}

if(a[j]=='1') {temp=Ht[temp.rchild];}

if(temp.lchild==-1) {printf("%c",temp.c);temp=root;}

j++;

}

}

void PrintCode(Code Hc[])

{

int i,j;

for(i=0;i<N\_LEAVE;i++)

{

for(j=Hc[i].Start;j<N\_LEAVE;j++)

{

printf("%d",Hc[i].HufCode[j]);

}

printf("%5c\n",Hc[i].Char);

}

}

void FC(Code Hc[],char s[])

{

int i,j,k=0;

printf("\n");

printf("The code is :");

printf("\n");

while(s[k]!='\0'){

for(i=0;i<N\_LEAVE;i++)

{

if( s[k] == Hc[i].Char )

{

for(j=Hc[i].Start;j<N\_LEAVE;j++)

{

printf("%d",Hc[i].HufCode[j]);

}

}

}

k++;

}

printf("\n");

}

int main()

{

Node HufTree[N\_NODE];

Code HCode[N\_LEAVE];

int Wt[N\_LEAVE] = {64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1,48,51,80,23,8,18,1,16,1};

char s[]={"THISPROGRAMISMYFAVORITE"};

char a[]={"100"};

Huffman(HufTree,Wt);

Code\_Ht(HufTree,HCode);

PrintCode(HCode);

FC(HCode,s);

translate(HufTree,a);

}

## 调试分析

11110 D

100 E

00011 F

110001 G

0101 H

1010 I

000001000 J

0000011 K

11111 L

00010 M

1011 N

1101 O

110010 P

000001001 Q

0110 R

0111 S

001 T

01001 U

000000 V

00001 W

000001010 X

110011 Y

000001011 Z

The code is :

0010101101001111100100110110111000101101110000101010011100010110011000111110000000110101101010001100

问题：对文件不了解，并且建立哈夫曼树不熟

时间复杂度：O（n\*2）