# 哈夫曼编译器实验报告

班级：电信1809班 姓名：刘宏政

学号：U201813476 完成日期：2019/4/25

1. 需求分析
2. 利用输入的原始数据，即字符以及其对应的权值，建立哈夫曼树。
3. 利用已经建立好的哈夫曼树，对给出的字符串（即正文）进行编码。
4. 对接收的0，1代码串也利用哈夫曼树进行译码。
5. 对编码，译码进行打印处理。
6. 概要设计

此程序最主要即是利用字符以及其对于的权值建立哈夫曼树，并得到每个字符对于的编码。

设计思想：

1. 根据给定的n个字符以及对应权值构成一个由n棵二叉树构成的二叉树的集合。每一棵二叉树只有一个带权的根结点，其左右子树都为空。
2. 在上述所说集合中选取两棵根节点权值最小的树作为左右子树构造一棵新的二叉树，且使得新的二叉树的根节点的权值为其左右子树上根节点的权值之和。
3. 在原二叉树集合之中删除这两棵树，将新得到的二叉树加入二叉树集合中。
4. 重复（2），（3）过程，直到二叉树集合中只含一棵树为止，这棵树即是哈夫曼树。
5. 从哈夫曼根结点出发到叶子结点，以0表示左分支，以1表示右分支，以这个路径上分支字符组成的字符串作为该叶子结点字符的编码。

程序主要包括四大模块：

1. 主函数模块；
2. 建立哈夫曼树；
3. 编码模块；
4. 译码模块；
5. 详细设计

#include<stdio.h>  
#include<stdlib.h>  
#define N\_LEAVE 26  
#define N\_NODE  (26\*2-1)  
typedef struct \_Node  
{  
char character;  
int weight;  
int lchild;  
int rchild;  
int parent;  
}Node;  
typedef struct \_Code  
{  
int HufCode[N\_LEAVE];  
int  Start;  
char Char;  
}Code;  
void Huffman(Node Ht[],int Wt[])  
{  
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 = Wt[i];  
Ht[i].character = i+65;  
}  
else  
{  
Ht[i].weight = 0;  
Ht[i].character = '?';  
}  
}  
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].character;  
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.character);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 FindCode(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);  
FindCode(HCode,s);  
    translate(HufTree,a);  
}

1. 用户说明
2. 用户在主函数中提前定义字符所对应的权值，以及想要进行译码的0，1组成的编码序列和想要编码的由已定义的字符的字符串。
3. 回车，即可显示每个字符所对应的编码，以及字符串对应的编码，以及编码所对应的字符串。
4. 调试测试

程序运行成功并能成功得出结果。

1. 测试结果

在预定义情况下，将编码序列所对应的字符串设为THISPROGRAMISMYFAVORITE，再利用此结果进行编码，得出结果一致，即证明程序正常。