实习报告

**题目：霍夫曼编码**

**班级：电信1808 姓名：文一晴 学号：U201813452 完成日期：2019.4.25**

1. 需求分析
2. 利用赫夫曼编码进行通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据预先编码，在接收端讲传来的数据进行译码（复原）。对于双工信道（即可以双向传输信息的信道），每端都需要一个完整的编/译码系统。
3. 测试数据

输入空格和A~Z字母的频度：

186 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

请输入需要编码的字符串：

THIS PROGRAM IS MY FAVORITE

该字符串编码为：

1101000101100011111100010001010011000010010101011001011101100011111110010100011111110011101011000001001001001101101010

1. 概要设计

为实现上述功能，应以树来表示。

1. 本程序包含两个模块：
2. 主函数模块：

Int main（）{

设置字符数组和权重数组的值

设置测试串

调用函数}

1. 霍夫曼编码函数——实现树的抽象数据类型
2. 详细设计
3. Main函数
4. int main()
5. {
6. int n=27;
7. int m=2\*n-1;
8. htnode \*hfmTree=(htnode \*)malloc((m+1)\*sizeof(htnode));
9. char l[27];
10. int i=0;
11. for(i=1,l[0]=' ';i<27;l[i]='a'+i-1,i++);
12. int \_weight[30]={186,64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1,48,51,80,23,8,18,1,16,1};//Ȩ������
13. char test[50]={"this program is my favorite"};
14. huffmancoding(hfmTree,n,\_weight,l,test);
15. free(hfmTree);
17. }

2.操作函数

void huffmancoding(htnode \*ht, int n ,int \*\_weight,char \*letter,char \*test)

{

int m=2\*n-1;

int i=0;

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

{

ht[i].weight=\_weight[i-1];

ht[i].l=letter[i-1];

ht[i].parent=0;

ht[i].lchild=0;

ht[i].rchild=0;

}

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

{

ht[i].weight=0;

ht[i].l='0';

ht[i].parent=0;

ht[i].lchild=0;

ht[i].rchild=0;

}

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

{

int s1\_weight=999, s2\_weight=999;

int s1=-1, s2=-1;

//select(ht,i-1,&s1,&s2);

for(int d=1;d<=i-1;d++)

{

if(ht[d].weight<s1\_weight && ht[d].parent==0)

{

s1 = d;

s1\_weight = ht[d].weight;

}

}

for(int d=1;d<=i-1;d++)

{

if(ht[d].weight<s2\_weight && d!=s1 && ht[d].parent==0)

{

s2 = d;

s2\_weight = ht[d].weight;

}

}

ht[s1].parent=i;

ht[s2].parent=i;

ht[i].lchild=s1;

ht[i].rchild=s2;

ht[i].weight=ht[s1].weight+ht[s2].weight;

}

printf("end of building tree\n");

huffmancode code=(huffmancode)malloc((n+1)\*sizeof(char \*));

char \_code[n];

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

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

{

int start=n-1;

int c;

int f;

for(c=i,f=ht[i].parent;f;c=f,f=ht[f].parent)

{

if(ht[f].lchild==c)

{

--start;

\_code[start]='0';

}

else

{

--start;

\_code[start]='1';

}

}

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

printf("i = %d, \_code+start: %s\n", i, \_code+start);

strcpy(code[i],&\_code[start]);

printf("code[i]: %s\n", code[i]);

}

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

{

printf("--%s",code[i]);

}

printf("end of char encoding\n");

printf("\nresult:");

for(int k=0; k<strlen(test); ++k)

{

for(int i=0;i<27;i++)

{

if(letter[i]==test[k])

{

printf("%c -> %s\n", test[k], code[i+1]);

}

}

}

free(code);

}

1. 测试结果

输入this program is my favorite

输出结果

1101000101100011111100010001010011000010010101011001011101100011111110010100011111110011101011000001001001001101101010

1. 附录

源程序文件名清单

huffmancoding //实现单元

htnode //树结构

char node //储存编码单元