数据结构上机实验题报告

题目：基于哈夫曼树的数据压缩算法

姓名：王小龙　　班级：2020211310　　学号：2020211502　　提交日期：2021.11.28

一　题目

编写一个程序，将一段拟传输的数据中的内容，进行哈夫曼编码；同时将所收到的一段二进制数据进行解码，得到一段字符数据。

输入为一串字符串，当输入字符串为“0”时，输入结束。

输出为 2n+3 行（n 为输入串中字符类别的个数）。第 1 行为统计出来的字符出现频率（只输出存在的字符，格式为：字符：频度），每两组字符之间用一个空格分隔，字符按照 ASCII 码从小到大的顺序排列。第 2 行至第 2n 行为哈夫曼树的存储结构的终态。第 2n+1 行为每个字符的哈夫曼编码(格式为: 字符：编码)，每两组字符之间用一个空格分隔，字符按照 ASCII 码从小到大的顺序排列。第 2n+2 行为编码后的字符串，第 2n+3 行为解码后的字符串。

二　程序设计

算法分析：

void CreateHT() 哈夫曼树创建

假设有 n 个权值，则构造出的哈夫曼树有 n 个叶子结点。 n 个权值分别设为 w1、w2、…、wn，则哈夫曼树的构造规则为：

(1) 将 w1、w2、…，wn 看成是有 n 棵树的森林(每棵树仅有一个结点)；

(2) 在森林中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和；

(3)从森林中删除选取的两棵树，并将新树加入森林；

(4)重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求得的哈夫曼树。

void Code() 哈夫曼树编码

从叶子结点出发，向根前进，进一步进行判断，左孩子为 0,右孩子为 1。最后逆序保存即可。

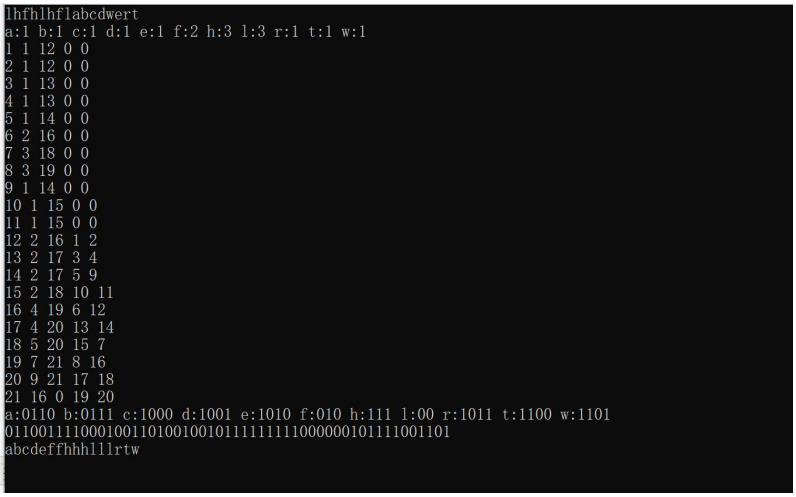
void Encode() 哈夫曼树解码

从根开始，左 0 右 1,到叶节点输出。再回到根节点重复即可。

三　程序测试运行

编译和运行环境为 vc++6.0；

运行测试：



四　编程工作总结

本次实验使我巩固了以前的知识并在此基础上还对数据结构的特点和算法有了更深的理解，使我在这门课程的实际应用上也有了一个提高。

五　程序源代码

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<string>
#include<set>
#include<list>
#include<vector>
#include<map>
#include<iterator>
#include<algorithm>
#include<iostream>
#define MAX 1000
#define MAXW 1000
using namespace std;
typedef struct HNode{
    char data;
    int weight;
    int parent;
    int lchild;
    int rchild;
}Hnode;
typedef struct HCNode{
    char data;
    string code;
}HCnode;
typedef struct minnodes{
    int m1;
```

```
    int m2;
    bool flag;
}minnodes;
bool flag[MAX]={false};
Hnode HT[MAX];
HCnode HC[MAX];
string s1="";
minnodes select(int max){
    double min = MAXW;
    minnodes mins;
    mins.m2 = 0;

    for (int i = 1;i < max;i++)
    {
        if (!flag[i] && HT[i].weight < min)
        {
            min = HT[i].weight;
            mins.m1 = i;
        }
    }
    flag[mins.m1] = true;
    min = MAXW;

    for (int i = 1;i < max;i++)
    {
        if (!flag[i] && HT[i].weight < min)
        {
            min = HT[i].weight;
            mins.m2 = i;
        }
    }
    flag[mins.m2] = true;
    if (0 == mins.m2)
    {
        mins.flag = false;
    }
    else
    {
        mins.flag = true;
    }
    return mins;
}//选择两棵最小权值的树
void PrintHT(int max){
    for (int i = 1;i<=max;i++)
```

```
    {
        cout <<i<<" "<< HT[i].weight << " " << HT[i].parent << " " << HT[i].lchild << " " <<
HT[i].rchild << endl;
    }
}//打印哈夫曼树
void PrintHC(int n){
    for(int i=1;i<n;i++){
        cout << HC[i].data << ":" << HC[i].code << " ";
    }
    cout << endl;
    for(int i=1;i<n;i++){
        for(int j=0;j<HT[i].weight;j++){
            cout << HC[i].code;
            s1+=HC[i].code;
        }
    }
    cout << endl;
}//打印编码
void Code(){
    int i=1;
    for(;;i++){
        int j=i;
        string str="";
        HC[i].data=HT[i].data;
        while(HT[j].parent!=0){
            if(HT[HT[j].parent].lchild==j){
                str+='0';
            }
            else{
                str+='1';
            }
            j=HT[j].parent;
        }
        reverse(str.begin(),str.end());
        HC[i].code = str;
        if (HT[i].lchild == 0&& HT[i].rchild == 0)continue;
        else break;
    }
    PrintHC(i);
}//哈夫曼编码
void Encode(){
    int root=1;
    while (HT[root].parent != 0) root++;
    int j = root;
```

```
    for (int i=0;i<s1.length();i++)
    {
        if ('0' == s1[i])
        {
            j = HT[j].lchild;
        }
        else
        {
            j = HT[j].rchild;
        }
        if (HT[j].lchild == 0 && HT[j].rchild == 0)
        {
            cout << HT[j].data;
            j = root;
        }
    }
    cout << endl << endl;
    s1.assign("");
}//解码
void CreatHT(){
    string str1;
    int a[30];
    cin >> str1;
    int n;
    while(str1[0]!='0'){
        int cnt=0;
        for(int j=0;j<MAX;j++){
            flag[j]=false;
        }
        for(int i=1;i<=26;i++){
            a[i]=count(str1.begin(),str1.end(),'a'+i-1);
            if(a[i]!=0){
                cout << char('a'+i-1)<< ":" << a[i] << " ";
                HT[++cnt].data='a'+i-1;
                HT[cnt].weight=a[i];
                HT[cnt].lchild=0;
                HT[cnt].rchild=0;
                HT[cnt].parent=0;
            }
        }
        cout << endl;
    n=cnt;
    minnodes mins;
    int i=n+1;
```

```
	for(;;i++){
		mins=select(i);
		if(mins.flag==false){
			HT[mins.m1].parent=0;
			break;
		}
		HT[i].weight=HT[mins.m1].weight+HT[mins.m2].weight;
		HT[mins.m1].parent=i;
		HT[mins.m2].parent=i;
		HT[i].data=' ';
		HT[i].lchild=mins.m1;
		HT[i].rchild=mins.m2;
	}
	PrintHT(i-1);
	Code();
	Encode();
	cin >> str1;
}
}//建立哈夫曼树并编码输出
int main(){
	CreatHT();
}
```