《数据结构》实验报告3

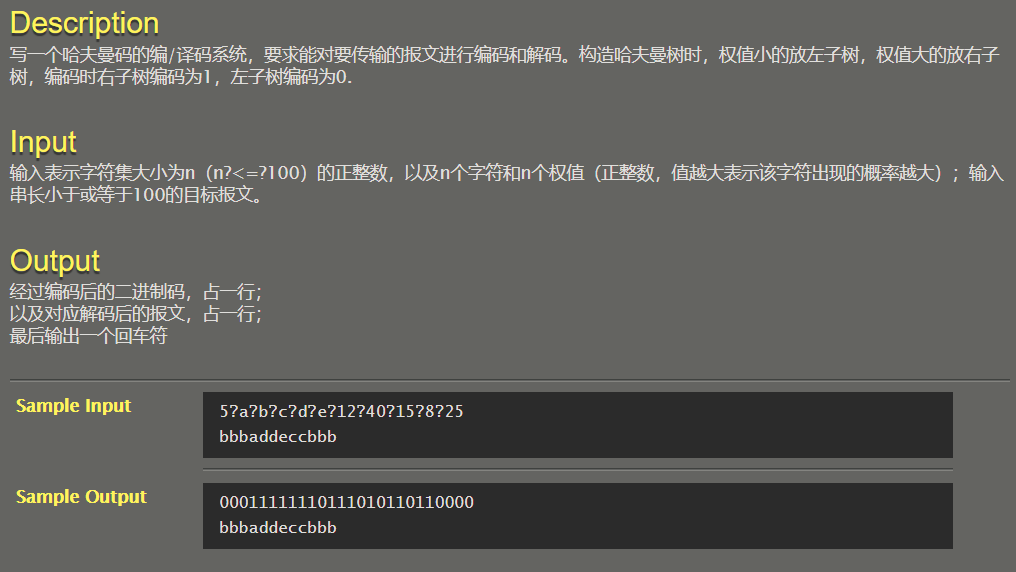
班级:DL062048 姓 名:白顺文 学 号:2020303092 E-mail:2662269845@qq.com日期:2021/5/9

**◎实验题目: 哈夫曼编/译码**

**◎实验目的：写一个哈夫曼码的编/译码系统，要求能对要传输的报文进行编码和解码**

**◎实验内容：写一个哈夫曼码的编/译码系统，要求能对要传输的报文进行编码和解码。构 造哈夫曼树时，权值小的放左子树，权值大的放右子树，编码时右子树编码 为 1，左子树编码为 0**

1. **需求分析**

说明程序设计的任务，强调的是程序要做什么，明确规定：

1. 输入形式：输入表示字符集大小为n(n<=100)的正整数，以及n个字符和n个权值(正整 数，值越大表示该字符出现的概率越大)；输入串长小于或等于100的目标报 文。

2、输出形式： 经过编码后的二进制码，占一行；

以及对应解码后的报文，占一行；

最后输出一个回车符；

3、实现功能：哈夫曼编码，设计出与报文相匹配的哈夫曼码

4、测试数据： 样例输入： 5 a b c d e 12 40 15 8 25

bbbaddeccbbb

样例输出： 00011111110111010110110000

bbbaddeccbbb

1. 概要设计

首先我们要根据字符集以及它的权重创建一个哈夫曼树，它的算法是每次从节点序列中选权值最小的两个节点将其合并后，再压入节点序列中，重复上述过程直到合并成只有一个根节点为止。若有n个节点，则最后会出现2\*n-1个节点。

所以我们要将他给的字符集以及权重输入进结构体当中，然后不断挑选节点进行合并操作，直到生成哈夫曼树。

其次，我们要根据所给的报文进行编码，编码函数使用递归的思想，不断遍历报文，并进行编码。

1. 详细设计

**数据结构：**

typedef struct node

{

int val; // 存储对应的权值

char data; // 存储对应的字符

struct node \*lchild;

struct node \*rchild;

}Node,\*Tree;

val 存储对应的权值

data 存储对应的字符

然后指向左右两个子节点

其他数据结构元素

int flag[maxn]={0};//标记节点是否被用过

Tree huff[maxn];//树节点

int n;//叶节点个数

char str[maxn];

string code[maxn];

int Getmin(int length);

Tree GetHuffman();

void Huffmancoding(Tree root,string str,int len);

GetHuffman函数用来创建哈夫曼树，其中通过一个do-while循环来选择flag为0未被用过的节点中权值最小的两个节点，通过调用Getmin函数获得，并将节点数放在l和r变量中，让左右子树节点指向。新建的节点data为空，说明是合并节点，权值是两个孩子节点权值的加和。通过这样直到l和r相等说明合并完毕。

源码：

Tree GetHuffman()

{

Tree root;

int len = n;

int l,r;

do

{

l = Getmin(len);

r = Getmin(len);

if(l==-1 || r==-1) break;

huff[len]->data = '\0';

huff[len]->val = huff[l]->val + huff[r]->val;

huff[len]->lchild = huff[l];

huff[len]->rchild = huff[r];

len++;

}while(l!=r);

root = huff[l];

return root;

}

Getmin函数用来寻找所有未被使用的节点当中的权值最小的节点，他通过遍历节点，通过一个中间变量来标记权值最小元素的下标，以及最小权值。如果这个节点最后被选中，那么对应的flag元素为1，防止下次被使用。

源码：

int Getmin(int length)

{

int k=-1,minn = 1000005;

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

{

if( flag[i] == 0 && huff[i]->val < minn )

{

minn = huff[i]->val;

k = i;

}

}

if(k!=-1) flag[k] = 1;

return k;

}

Huffmancoding函数用来对报文编码，通过递归的过程，从root往下递归，string记录当前已经存储的编码序列，len为序列的长度，这样不断地为右子树序列后面+1，回溯后在左子树后面+0。如果当前节点data域不为空，就用一个code数组将字符映射到当前的string上，建立两者的对应关系，编码的时候直接查询即可。

源码：

void Huffmancoding(Tree root,string str,int len)

{

if(root == NULL) return ;

if(root->data != '\0')

{

code[(int)(root->data)] = str;

}

str += "1";

Huffmancoding(root->rchild,str,len+1);

str[len] = '0';

Huffmancoding(root->lchild,str,len+1);

}

主函数：

int main()

{

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

{

huff[i] =(Tree)malloc(sizeof(Node));

huff[i]->lchild = NULL;

huff[i]->rchild = NULL;

huff[i]->data = '\0';

huff[i]->val = 0;

}

scanf("%d",&n);

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

{

scanf("%c",&huff[i]->data);

}

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

{

scanf("%d",&huff[i]->val);

}

scanf("%s",str);

Tree root = GetHuffman();

Huffmancoding(root,"\0",0);

int len = strlen(str);

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

{

printf("%d",code[(int)str[i]]);

}

printf("\n%s",str);

return 0;

}

功能是输入数据，并调用函数创建哈夫曼树以及对报文进行编码并且输出。

**(四) 程序使用说明及测试结果**

1．程序使用说明

（1）本程序的运行环境为VC6.0。

（2）输入数据： 输入表示字符集大小为n(n<=100)的正整数，以及n个字符和n个权值(正 整数，值越大表示该字符出现的概率越大)；输入串长小于或等于100的目 标报文。

2．测试结果： 样例输入： 5 a b c d e 12 40 15 8 25

bbbaddeccbbb

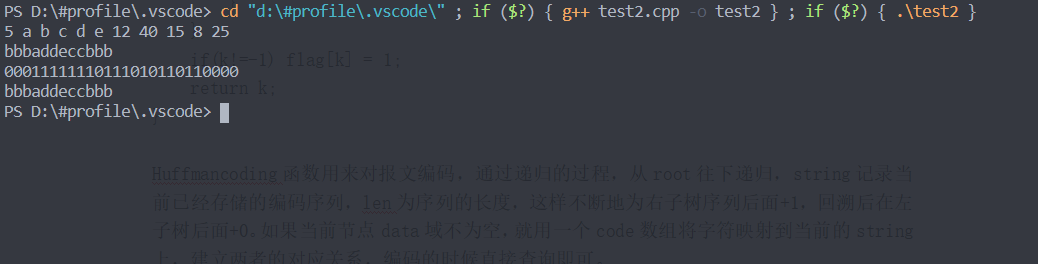
样例输出： 00011111110111010110110000

Bbbaddeccbbb

1. 调试过程中遇到的问题是如何解决提以及对设计与实现的回顾讨论和分析：

报文编码函数出问题，debug好几遍发现问题在遍历顺序出错。

4. 运行界面：



1. 、实验总结(实验心得)

通过此次实验，第一次使用哈夫曼树进行编码，对哈夫曼树的创建也有了了解，能用代码去进行操作，并且了解了哈夫曼编码的规则，以及如何遍历哈夫曼树创建编码。并且不断提高自己的debug调试能力，深入变量内部查看函数进行过程。

**教师评语：**

**实验成绩：**

指导教师签名：

批阅日期：