《数据结构》实验报告T007

班 级: DL062138 姓 名: 张苏宇 学 号: 2021302853

E-mail:905159071@qq.com 日 期: 2022/5/7

◎实验题目: T007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**

**输入：**

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。

**输出：**

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。

样例输入：5 a b c d e 12 40 15 8 25

Bbbaddeccbbb

样例输出：00011111110111010110110000

Bbbaddeccbbb

**拟解决关键问题：**

关键问题就是主要的两个操作，如何建立哈夫曼树，以及如何在哈夫曼树上完成对每一个字符的编码。

**二 概要设计**

**哈夫曼编码：**

首先我们考虑到哈夫曼树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。因此我们需要做的是先将输入的权值序列与字符对应后存入结构体，然后每次挑选节点合并压入队列合并。

**解码：**

对于解码，我们可以使用map映射字符串与字符，也可以使用哈希来进行。

**三 详细设计**

**存储结构：**

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;

我选择使用一个结构体来存储，里面包含了左右孩子指针，以及这个节点存储的字符data，以及他对应的权值val，这样既可以保证我们可以分别他是否是一个叶子节点，也可以分辨他的权值。

建树函数：Tree GetHuffman()

建树我是通过一个 do-while循环来每次选择两个权值最小的节点，然后将其合并到一个新的节点上，新节点的左右孩子为选出的两个节点，data域为空，说明他不是叶子节点，权值域为两个孩子节点权值的加和。

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;

}

选择函数：int Getmin(int length)

这个函数是用于选择在length序列中权值最小的那个节点，返回值为节点在序列中对应的下标。其中实现的过程为扫描 0 ~length-1 ，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。选出后通过一个bool数组来标记该元素是否已经使用过，初始值为false，使用后标记为true，这样可以避免重复选择。

int Getmin(int length) {

int k = -1, minn = 1000005;

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

if ( vis[i] == false && huff[i]->val < minn ) {

minn = huff[i]->val;

k = i;

}

}

if (k != -1)

vis[k] = true;

return k;

}

编码函数：void Huffmancoding(Tree root,string str,int len)

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。

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);

}

四 使用说明、测试分析及结果

1. 说明如何使用你编写的程序；

命令行/IDE中运行后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。

1. 测试结果与分析；

正确的输入：5 a b c d e 12 40 15 8 25

bbbaddeccbbb

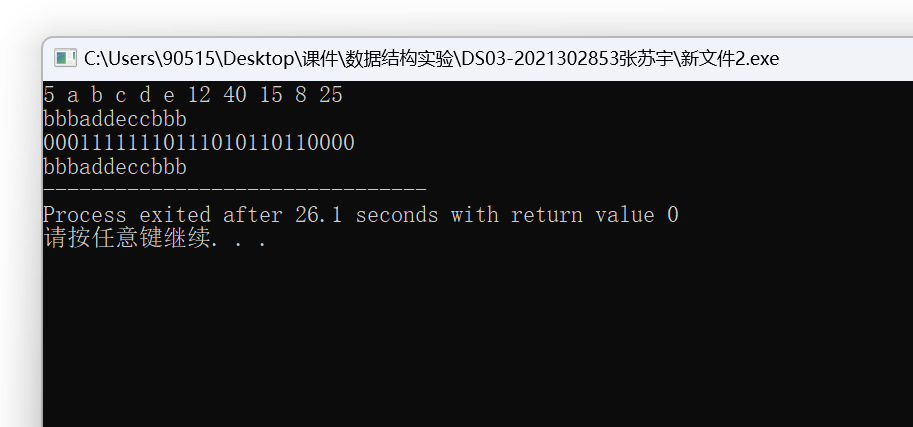
正确的输出：00011111110111010110110000

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

调试过程中出现了PE了，后来分析后发现是由于编码过程中没有回溯，加入回溯后得以解决。

1. 运行界面



1. 实验总结(实验心得)

哈夫曼树的定义：给定N个权值作为N个叶子结点，构造一棵二叉树，若该树的带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树(Huffman Tree)，哈夫曼树是带权路径长度最短的树。权值较大的结点离根较近。

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。哈夫曼树还是比较容易理解，主要构造利用贪心算法的思想去从下往上构建，学了之后有所收获。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。