

题目：哈弗曼编码/译码

学生姓名：郑嘉涛

学号： 20152100015

专业班级：计算机类5班

指导教师：陈红英

哈弗曼编码/译码

摘 要

如今社会发展越来越快，对数据的传送处理需求越来越大。在远古时代，人们用结绳来记数，在石壁上面画画来保留信息等，远古的信息传送处理方式适合于远古，而对于现在的信息社会，再使用这样的技术已经不再可能，因为这样的方式实在是太慢，太落后了。再看看现在这个时代，我们有多少次尝试过下载东西下载的很慢？或者说在下载到99%时停住甚至于下载的文件是损坏的。在这种时候，有没有什么方法可以帮助我们的下载速度提升，或者说减小下载的文件的大小，让我们可以在下载下来以后在本地对文件进行恢复操作呢，第一个问题我们不讨论，但是第二个问题就是这里要研究的问题。哈弗曼编码是一种有效的二进制编码，可以将不同频率的输入值进行编码，使之能够得到有效的压缩，并且也可以没有损失的将编码恢复成原来的输入值，因此能够使原来的文件得到压缩而在译码后不损坏原来文件的内容。

**关键词**：C++；面向对象；哈弗曼树；编码/译码；压缩；

前言

**实验目的：**

建立哈夫曼树，使用该哈夫曼树建立哈夫曼编码，使用该编码将输入文件的内容进行编码压缩输出存在另一个文件中，然后再从该文件中读取出来进行译码操作，译码的结果保存在另外的文件中。

**实验意义：**

实验建立的哈夫曼编码，能够有效的对文件进行编码压缩，同时也能够译码从二进制文件得到原来的文件。

**实验内容概述：**

实验首先使用传入的权值建立了一个哈夫曼树，然后使用传入的数据建立了哈夫曼编码，实验还建立了一个专门的编码类，用来操作建立哈夫曼编码后的编码值。实验还将一个本地的文件的内容进行了译码，译码后的结果保存在本地文件中，只需将该文件保存，并且使用此实验建立的程序对该文件进行译码操作，就能的到原来文件的内容而没有损失。

实验设计

**采用的关键技术：**

程序采用的关键技术如下：

程序建立了一个编码类，该类的对象可以直接对编码进行操作，可以用来保存编码。

这是程序建立哈夫曼树的过程。

bool hfmTree::create(BinTreeNode \*&root , char \*key , int \*value , int n){

if(value == NULL)return false;

int m = 2 \* n - 1;

BinTreeNode \*\*t = new BinTreeNode\*[m];

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

if(i < n){

t[i] = new BinTreeNode(value[i] , key[i]);

}

else t[i] = new BinTreeNode();

}

for(int i = 0 , j = 0 ; i < m-1 ; i += 2 , j += 1){

chooseSortTree(t , i , n + j);

t[i]->parent = t[n + j];

t[i+1]->parent = t[n + j];

t[n + j]->value = t[i]->value + t[i+1]->value;

t[n + j]->leftChild = t[i];

t[n + j]->rightChild = t[i+1];

}

root = t[m-1];

return true;

}

在这个建立的过程中，没有使用通常使用的最小堆来进行哈夫曼树的建立操作，而是使用了数组来建立哈夫曼树。对于一个有n个叶子节点的树来说，总共的节点数是2n-1，所以可以建立一个2n-1的数组，前面n个值保存的都是建立了对应权值的单节点树，并且已按照权值进行了排序。使用前面两个节点建立了局部哈夫曼树，并将新的节点放到n+1的位置上，于此同时，前面两个节点看做舍去，不断重复这个过程，最终得到的最后一个节点的就是所建立的哈夫曼树的根结点。

其他的关键技术还包括编码译码。编码的过程就是对建立好的二叉树进行前序遍历，这里使用到前面建立的编码类，二叉树的左孩子代表0，右孩子代表1，当二叉树遍历完毕时，也就建立好编码，并且存在了一个编码类的对象的数组中，后面的操作都可以直接使用。译码操作则是将读入的文件中的内容与这里建立的编码值进行比较以此得到原来的数据。

**主要设计思想与设计流程：**

设计思想：

建立哈夫曼树，建立哈夫曼编码，将哈夫曼树建立的哈夫曼编码保存在一个编码数组中，与其的关键码对应，在后面的文件的操作中直接进行使用，以此来完成哈夫曼编码译码的实验。

设计流程：

二叉树结点类

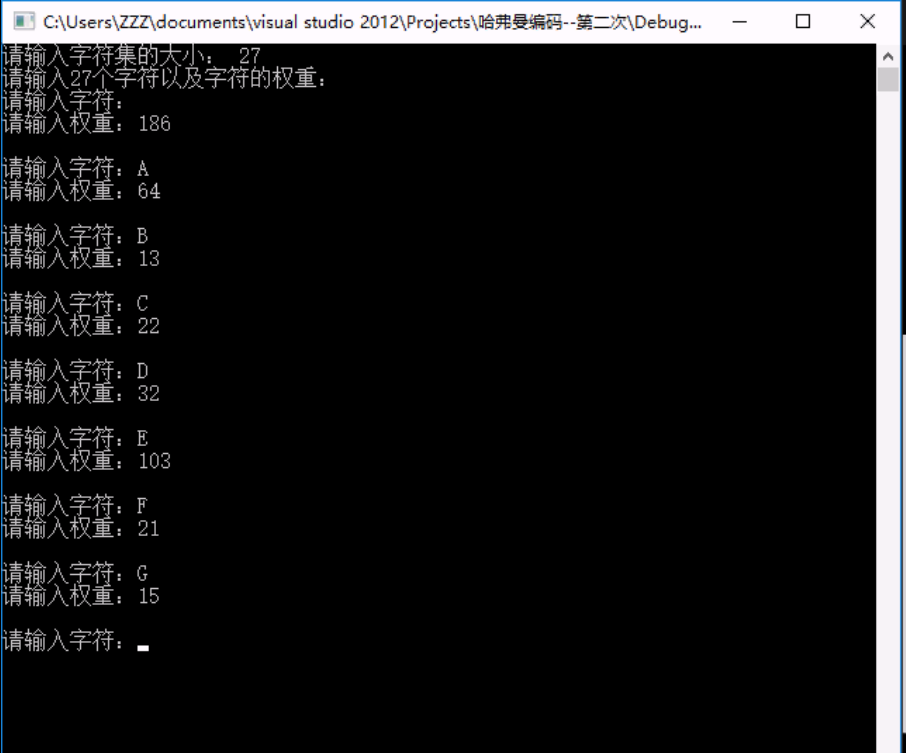
编码类

建立哈夫曼树

文件编码译码

实验实现

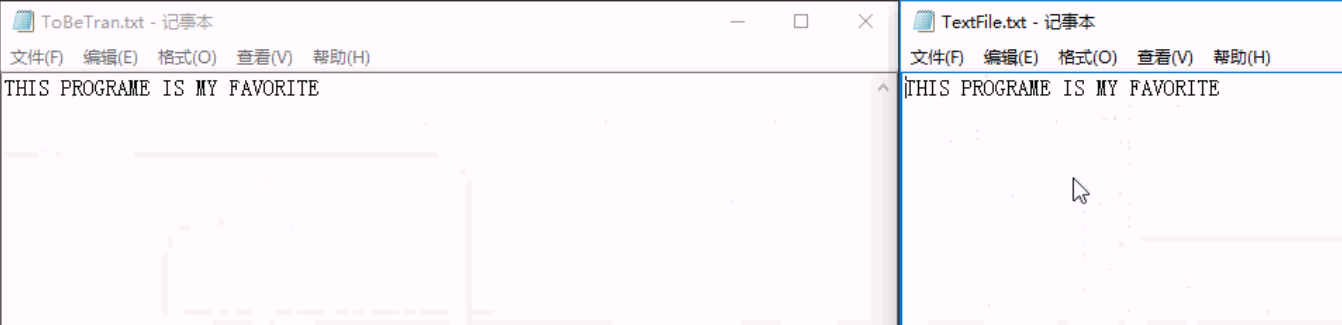
**主要功能模块分析:**

****

建立哈夫曼树，用户根据提示输入字符集的大小，然后输入不同的字符和权重，这些字符会被单独保存下来，但是权重不会被保存，权重值会保存在建立的哈夫曼树中。这里没有使用最小堆建立哈夫曼树，而是使用了一个数组来建立哈夫曼树，对于具有n个叶子节点的二叉树，树中的节点总共有2n-1个，所以需要建立一个2n-1的数组来保存。先建立好哈夫曼树的所有叶子节点，保存在数组的前n个位置，然后进行排序，保证最小的两个权值保存在最前面。然后取最小的两个权值建立一个树并保存在数组的n+1个位置，然后舍去最小的两个权值。不断重复这个过程，最后建立的二叉树的最后一个节点就是哈夫曼树的根结点。

****

这是建立的哈夫曼编码后输出的结果。



这是译码后的结果，可以看到，文件译码后与原文件没有区别。

**采用的软硬件平台介绍:**

软件平台：

Windows10：

美国[微软](http://baike.baidu.com/view/2353.htm" \t "http://baike.baidu.com/_blank)公司所研发的新一代跨平台及设备应用的[操作系统](http://baike.baidu.com/subview/880/4940471.htm" \t "http://baike.baidu.com/_blank)。

Visual studio 2012：

[Visual Studio](http://baike.baidu.com/view/28727.htm" \t "http://baike.baidu.com/item/_blank)是微软公司推出的开发环境。是最流行的[Windows](http://baike.baidu.com/view/4821.htm" \t "http://baike.baidu.com/item/_blank)平台应用程序开发环境。

硬件平台：

处理器：Intel Core i7-4710MQ

内存：8GB

显卡：Intel HD Graphics 4600

结果分析和讨论

在这个实验的过程中，完成了哈夫曼树的建立，并且通过哈夫曼树的使用，建立了一个哈夫曼编码，使用该编码能够有效的将文件进行编码成二进制文件，同时也能够将二进制文件进行译码，较好的还原成为原来的字符文件，供用户使用，所以从这个意义上而言，这个实验是成功的。但是从另一个方面而言，这个实验还有几个不过好的地方，例如使用数组建立哈夫曼树，空间复杂度较高，如果使用最小堆来建立则效果更好，而且使用最小堆来建立，时间复杂度也更让人满意。文件编码后写入文件中使用了换行符，因为哈夫曼编码是一种无前缀编码，所以如果不使用换行符来编码写入文件的话，效果会更加好。基于这些原因，这个实验还有较多的改进空间。

参考文献

《C++程序设计（第2版）》谭浩强

《数据结构（用面向对象方法和C++语言描述）》

《算法导论》

《数据结构（C语言版）》