

**实验报告**



**题目： 实验三：树-哈夫曼编/译码器**

**班 级： 2022211313**

**学 号： 2022211363**

**姓 名： 谢牧航**

**学 院： 计算机学院**

**2023年11月 28 日**

1. 实验目的
2. 熟悉哈夫曼树的构造实现及应用；
3. 培养根据实际问题合理选择数据结构的能力；
4. 学习自己查找相关资料以解决实际问题的能力。

二、实验环境

简述使用的工具（操作系统+c环境+编译器）

操作系统：Windows 11

C语言环境+编译器：Vscode + gcc version 8.1.0

三、实验内容（简单行编辑程序）

**问题描述：**

哈夫曼编码是一种基于最优二叉树的无损编码方案。需要根据字符集和频度的实际统计构建哈夫曼树，然后进行编码和译码。

**基本要求：**

1. 初始化，从终端读入字符集大小n，以及n个字符和n个权值，建立哈夫曼树，并将它存于文件hfmTree中。

2. 编码：利用已经建好的哈夫曼树，对文件ToBeTran中的报文进行编码，然后将结果存入文件CodeFile中。（为简化处理，可以在CodeFile中用一个字节来存储码字中的一个0/1比特位）

3. 译码：利用已建好的哈夫曼树，对CodeFile中的代码进行译码，结果存入TextFile中。

**选做内容**

对一个512\*512的lena.bmp灰度图片进行哈夫曼编码。BMP文件由：BMP文件头+像素数据组成，灰度图1个像素占用1个字节。lena.bmp文件大小是263222字节，包括1078字节的头部+512\*512个像素值。了ena.bmp文件见实验作业附件

四、实验步骤（基础80分+附加10分）

操作步骤+运行截图

## 阶段一

初始化，从终端读入字符集大小n，以及n个数值和n个权值，建立哈夫曼树，并将它存于文件hfmTree中。

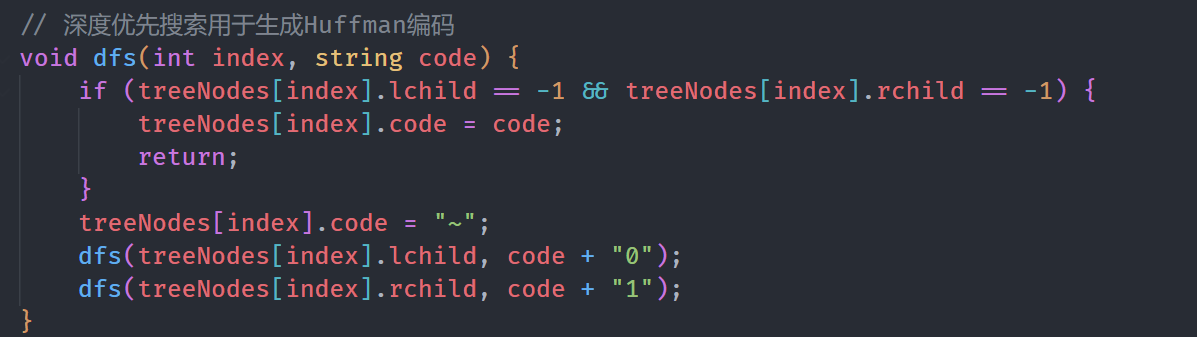
（因为为了方便后面对bmp文件的编码，我编写的文件用整数值代替了字符，实际算法思想和数据结构设计不影响）

首先定义数据结构如下：

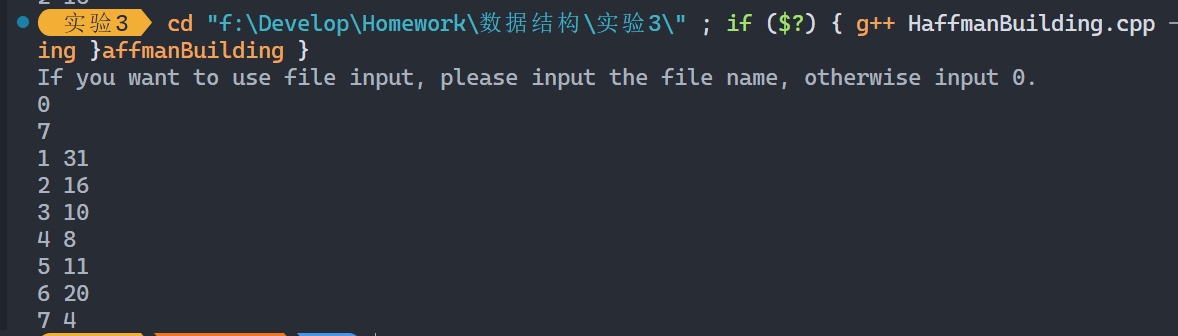


建立哈夫曼树，这里我们使用了优先队列也就是二叉堆来寻找每次权值最小的两个结点，这样比朴素的方法速度更快，时间复杂度为O(nlogn)。

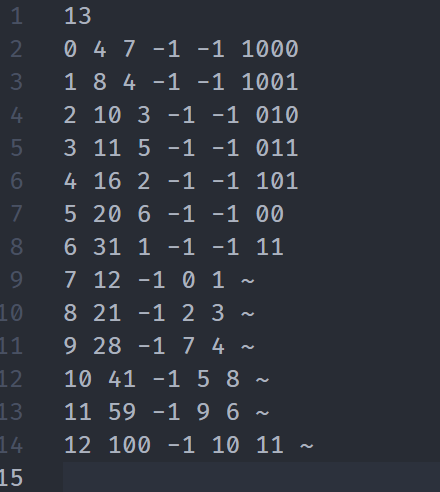


首先进行哈夫曼编码。再按照0向左，1向右的原则dfs遍历树，保存每个结点的编码值。

输入测试数据：



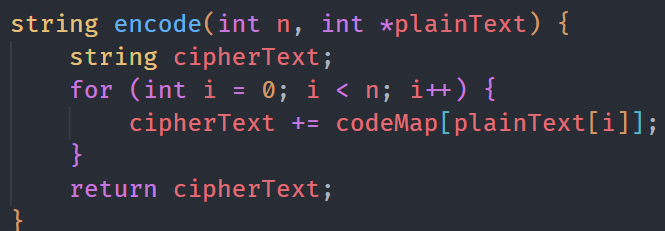
输出哈夫曼树如下：



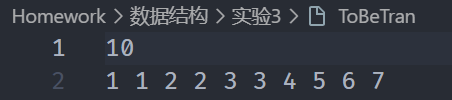
## 阶段二

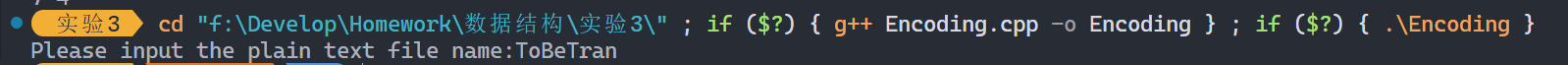
编码：利用已经建好的哈夫曼树，对文件ToBeTran中的报文进行编码，然后将结果存入文件CodeFile中。（为简化处理，可以在CodeFile中用一个字节来存储码字中的一个0/1比特位）

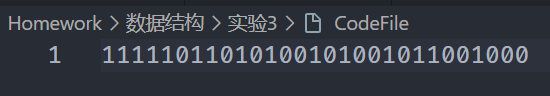
这个阶段比较简单，读入保存的哈夫曼树后，我们只需要根据编码值，把输入数据拼接成一个编码字符串即可。为了方便，这里我们使用的C++的STL数据结构unordered\_map，也就是哈希表，这样可以将结点内容和编码一一对应，方便查找。



输入测试数据如下





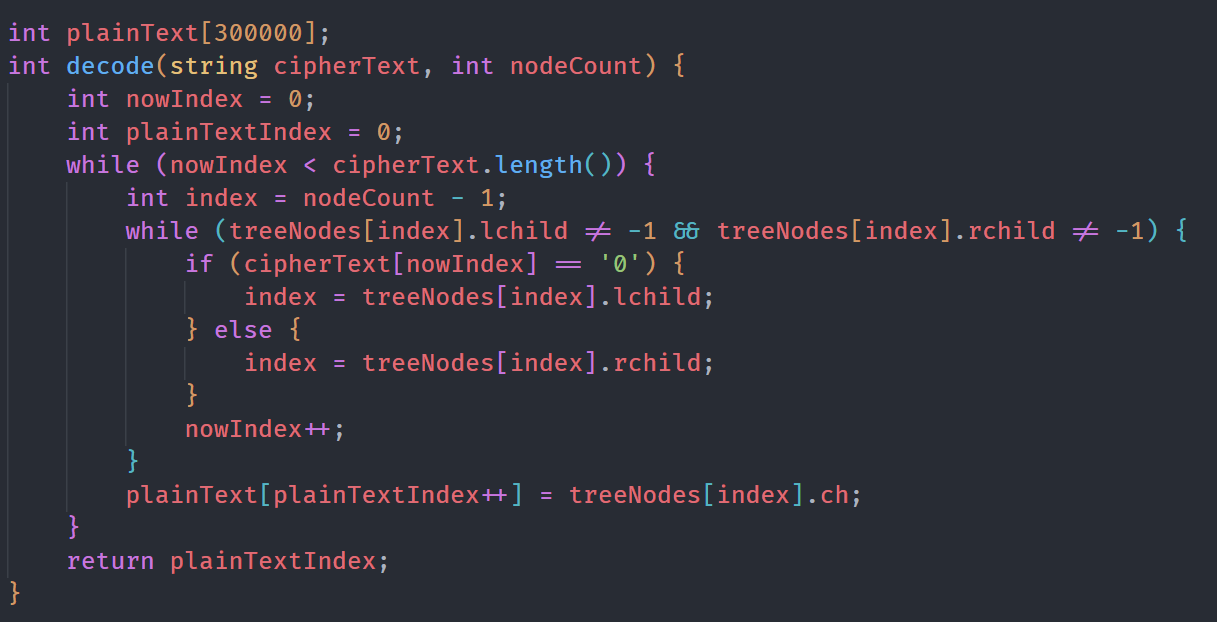


经计算这棵树的WSL为2.61，比正常二进制编码的3精简了13%。

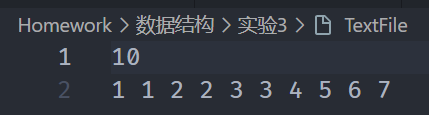
## 阶段三

译码：利用已建好的哈夫曼树，对CodeFile中的代码进行译码，结果存入TextFile中。

由于哈夫曼树是无前缀相同的编码，所以我们只需要按顺序读入编码字符串，按照0向左，1向右的原则，只要走到叶子结点就能得到一个原文数值。



测试结果如下：



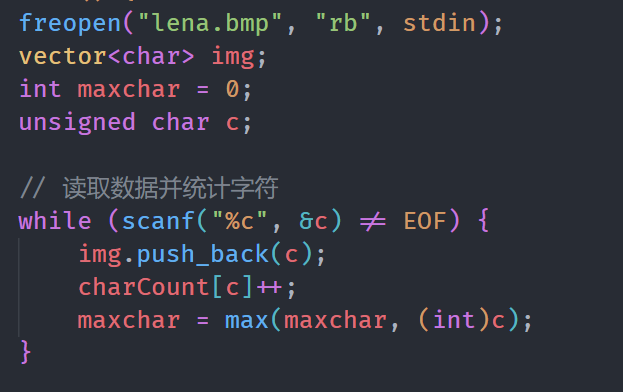
发现TextFile还原了我们的原本的测试数据，解码成功

## 阶段四

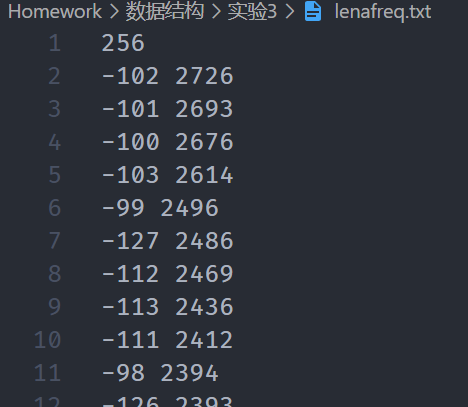
**选做内容**

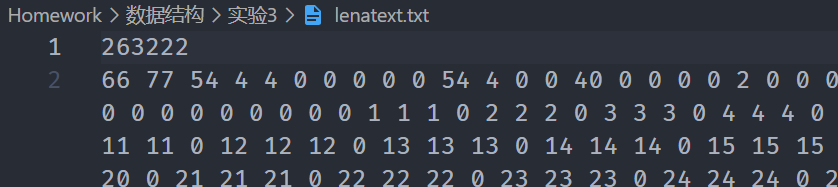
对一个512\*512的lena.bmp灰度图片进行哈夫曼编码。BMP文件由：BMP文件头+像素数据组成，灰度图1个像素占用1个字节。lena.bmp文件大小是263222字节，包括1078字节的头部+512\*512个像素值。了ena.bmp文件见实验作业附件

Bmp文件实际上是由二进制比特位来描述的图片文件，我们可以直接以unsigned char数组格式读入（每1字节8bit存入数组），然后对这个数组进行哈夫曼编码即可。

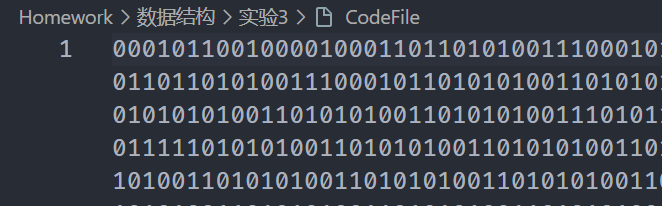


然后我们将统计的频度排序后输出到lenafreq.txt，bmp数据数组格式输出到lenatext.txt。

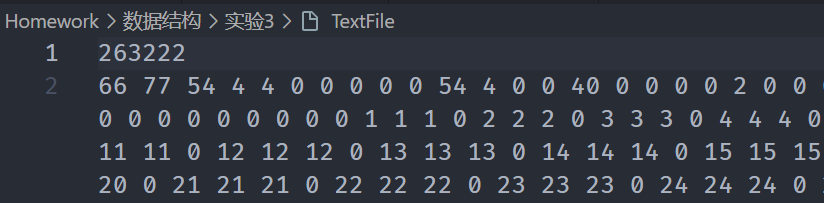




对lenafreq.txt进行哈夫曼建树，对lenatext.txt进行编码如下：



对CodeFile解码



还原了原bmp文件数组格式。

五、实验分析和总结（10分）

实验时的工作思路、设想、效果等综合分析（包括针对自己设计的数据结构，简要分析说明各种操作的时间复杂度。）

哈夫曼编码是一种广泛用于数据压缩的算法，基于字符出现的频率构造最优前缀码。较频繁的字符使用较短的编码，较少出现的字符使用较长的编码。

设计数据结构：

优先队列：用于构建哈夫曼树，通常使用最小堆实现。

哈夫曼树：二叉树结构，存储字符及其频率。

哈希表：存储字符与其对应的哈夫曼编码，便于快速编码和解码。

实验设想：

输入一段文本，计算每个字符的频率。构建哈夫曼树并生成对应的哈夫曼编码。使用生成的编码对文本进行编码和解码，验证正确性。比较原始数据与编码后数据的大小，计算压缩率。

编码过程由于使用了有限队列，所以建树操作复杂度为O(nlogn)，编码操作为O(n)，译码过程复杂度为O(lenlogm)，其中m是不同的字符数。最后效果符合预期。

总结心得（包括遇到的困难，自己一些不成功的设计和设想）

读入Bmp文件时，一开始freopen标签使用的为“r”，而不是“rb”，导致读入错误。实际上应该以二进制文件读入。

六、程序源代码（10分）

见压缩包中HaffmanBuilding.cpp，Encoding.cpp，Decoding.cpp，ReadBmp.cpp。