**《数据结构》实验报告X**

**学号：\_09022107\_ 姓名 ：\_梁耀欣\_\_\_\_\_\_\_**

**实验题号：5 实验日期：\_**2023.12.6**\_\_ 实验类型：** 必做

**1．问题描述：**描述实验内容和要求以及需要解决的问题。

**提示：**结合教师课堂讲授内容，仔细分析实验要求，对实验内容进行需求分析。

构建一个哈夫曼树ADT，用相应的数据结构存储这个树，完成读取文件、文字中字符频率统计、树的构造、结点的编码、整段文字的编码以及解码，并输出在屏幕上。

**2．算法思想：**详细描述解决相应问题所需要的算法设计思想。

首先通过getcount()函数统计输入文件中每个字符的频率，将统计结果保存在charcount的map容器中。构造HT类对象ht，传入字符频率的个数和频率统计结果charcount。在HT类的构造函数中，根据传入的字符频率，初始化哈夫曼树的节点数组ht。其中，叶子节点的数据为字符，频率为对应字符的出现频率。构建哈夫曼树，依次选择频率最小和次小的两个节点，创建一个新的父节点，并将其频率设置为两个子节点的频率之和。重复此过程，直到所有节点都被加入树中。通过递归的方式，生成每个字符的哈夫曼编码。从根节点开始，左子树路径记为0，右子树路径记为1，沿着树的路径依次生成编码。提供编码和解码功能： encode()函数读取输入文件，将每个字符替换为其对应的哈夫曼编码，并将编码结果输出。 decode()函数读取编码后的文件，根据哈夫曼编码逐个匹配字符，将编码还原为原始文本。最后，在main()函数中，先将字符频率统计结果输出到文件"result.txt"中，然后创建HT对象，显示哈夫曼树的结构，进行编码和解码操作。

**3．功能函数：**描述所设计的功能函数。如果有多个函数，需要描述它们之间的关系。

1. map<char, int> getcount(): 该函数从文件中读取字符，并统计每个字符出现的频率，返回一个存储字符及其频率的map。

2. HT::HT(int n, map<char, int> mp): HT类的构造函数，用于构建哈夫曼树。它接受字符频率的映射以及字符数量作为参数，并根据哈夫曼算法构建树结构。

3. HT::~HT(): HT类的析构函数，释放动态分配的内存。

4. void HT::select(int n, int& s1, int& s0): 该函数用于选择前n个结点中频率最小和次小的两个结点。

5. void HT::decode(int n): 解码函数，根据已经构建好的哈夫曼树，将编码后的文本解码成原始文本。

6. void HT::generateCodes(HN\* node, char\* code): 给每个字符创建哈夫曼编码。

7. void HT::encode(int n): 编码函数，根据已经构建好的哈夫曼树，将原始文本编码成二进制字符串。

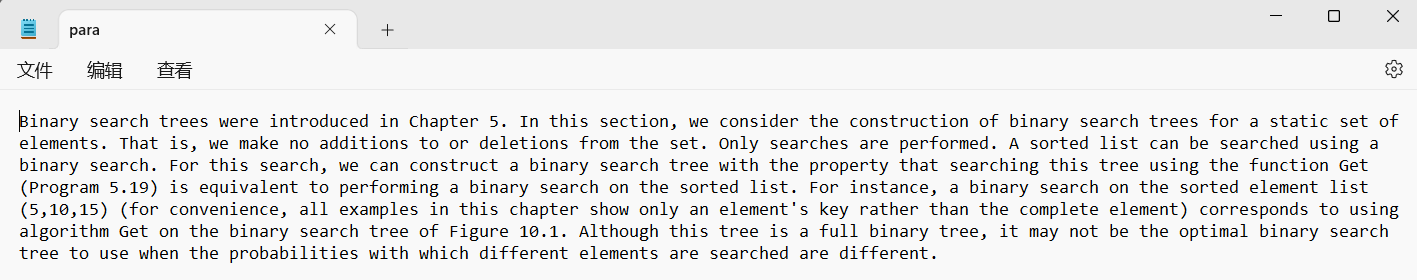
8. void HT::show(int n): 打印哈夫曼树的结点信息。

在HT类的构造函数中，根据字符频率的映射构建了哈夫曼树，并为每个叶子结点生成了对应的哈夫曼编码。然后在编码和解码函数中，根据已经构建好的哈夫曼树，进行编码和解码操作。show()函数用于打印哈夫曼树的结点信息，方便查看树的构造情况。

**4．测试数据：**设计测试数据，或具体给出测试数据。

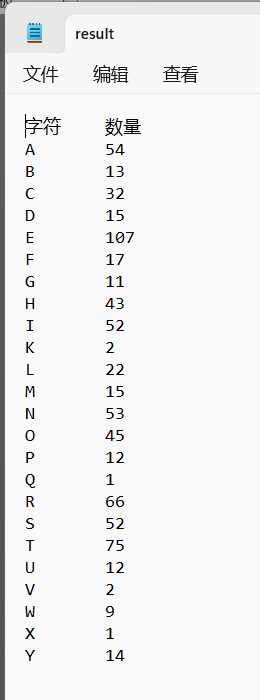
提示：要求测试数据能全面地测试所设计程序的功能。

**在文件中输入两个英文段落，要求根据它进行哈夫曼编码并把这段文字译码和解码：**

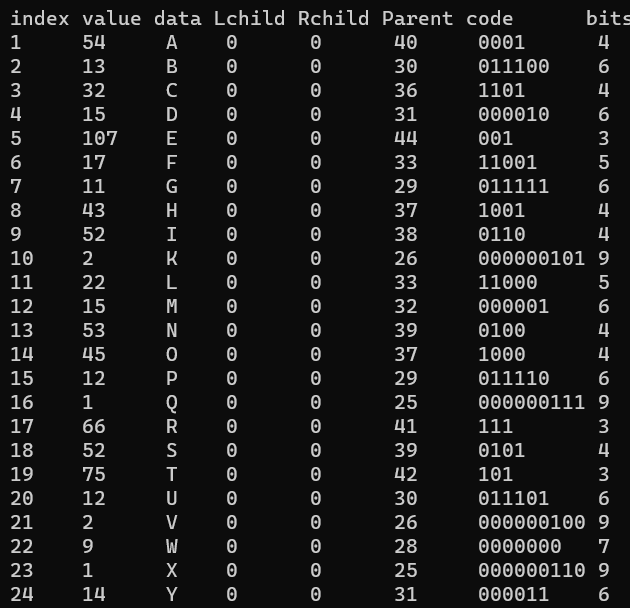


**5．测试情况：**给出程序的测试情况，分析运行结果，显示实验结果截图。

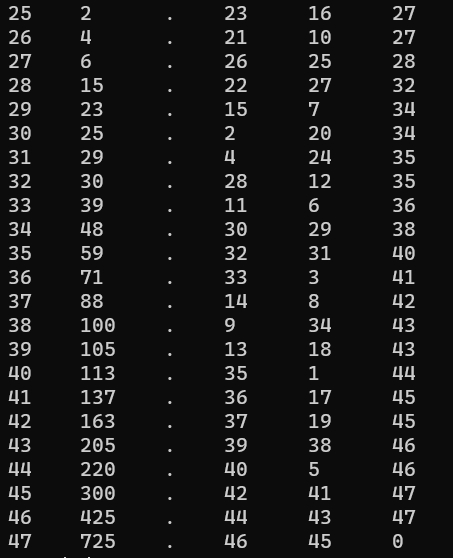
**首先，对每个字符的频率进行统计，结果保存到map中，把字符和数量输出到文件”result.txt”内：**



**然后将字符的左右子结点下标、父结点下标、编码、位数显示到屏幕上：**



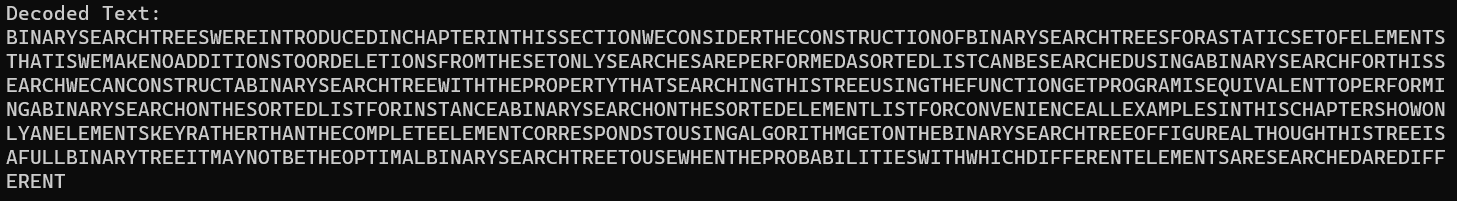
**再把根据这些有值结点构造出来的其他节点输出：**



**把这段话的编码输出：**



**把编码复制到txt文件中，读取txt文件并进行解码：**



**6．实验总结：**写出实验过程中遇到的问题，以及问题的解决过程。分析算法的时间复杂度和空间复杂度，总结实验心得体会。

**时间复杂度：O(nlogn)，其中 n 为字符的种类数；编码和解码的时间复杂度与待编码文本长度成正比。**

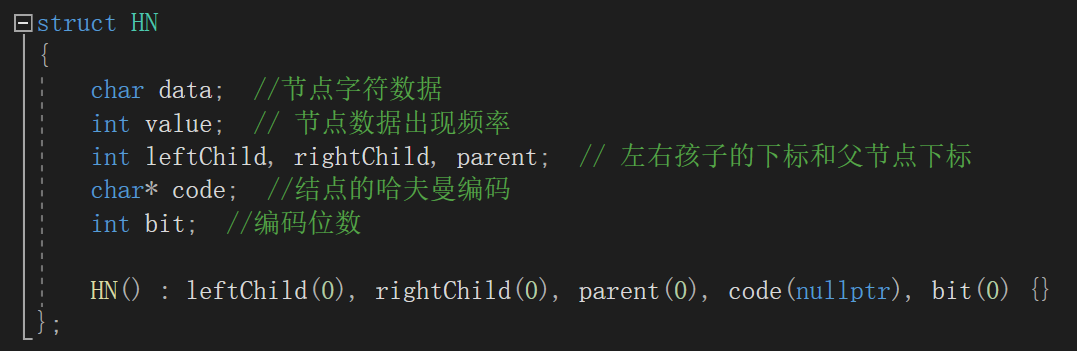
**空间复杂度： O(n)，其中 n 为叶子节点个数；编码和解码的空间复杂度与待编码文本长度成正比。**

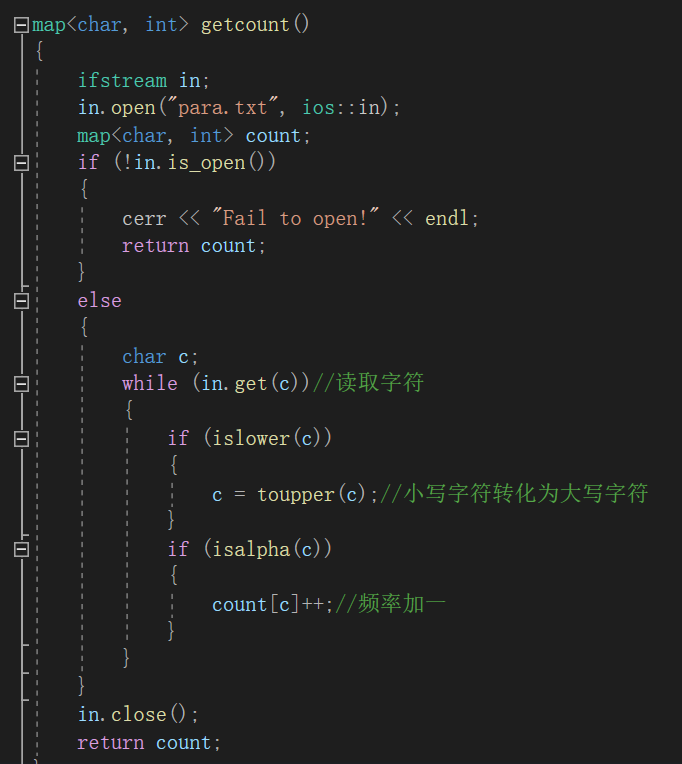
**实验中遇到的问题：解码的时候一开始没有把比较函数设置为前n位比较，忽略了后面的位数为0；得到每个字符的code的时候被怎样倒序输出困扰，后来想到了用栈的方式。**

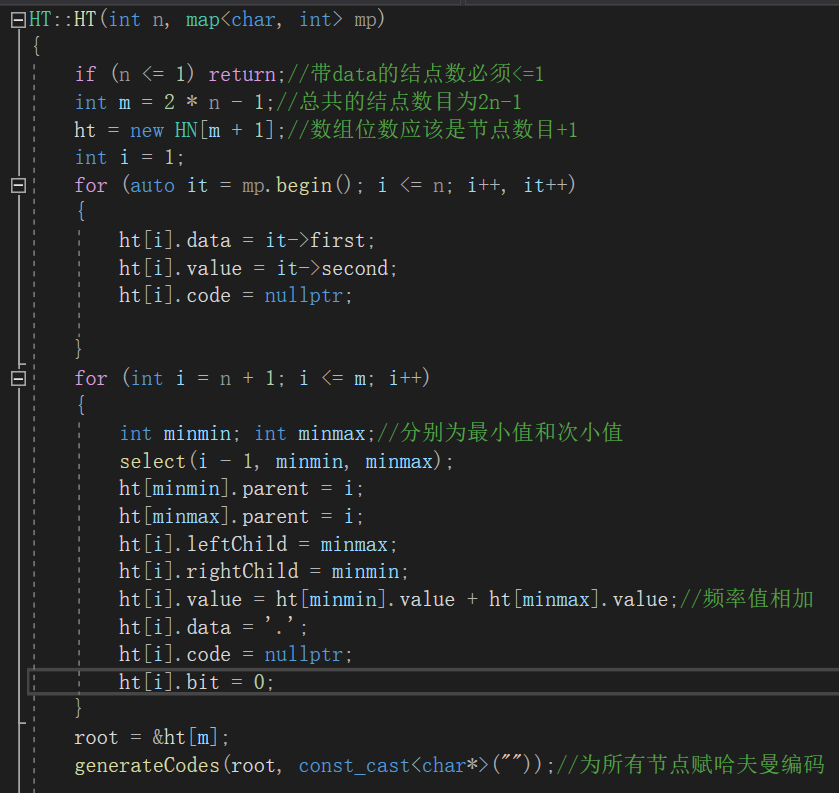
**心得体会：通过这个实验，我深入理解了哈夫曼编码的原理和实现方法，掌握了编码树的构建、编码和解码过程。在实现过程中，加深了对数据结构和算法的理解，同时也学会了如何处理文件操作和内存管理方面的一些常见问题。**

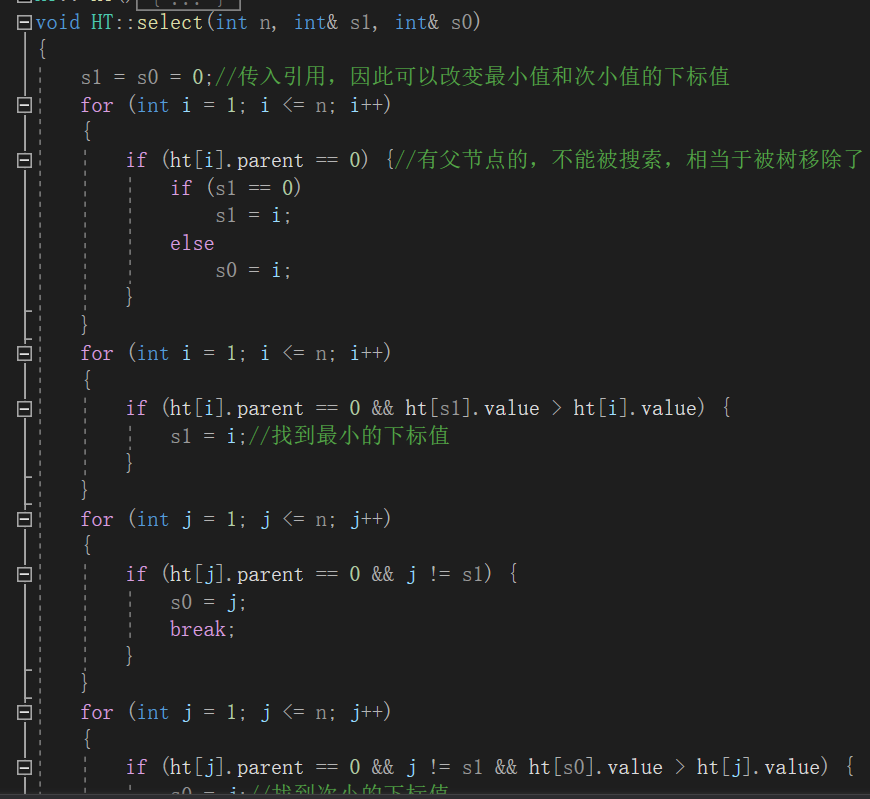
**7. 主要源代码：**

建议：源程序中应有充分的注释，例如注释每个函数参数的含义、函数返回值的含义、函数的功能、主要语句段的功能，等等。









**8. 实验源程序：**附件清单

(1) HUFFMAN.cpp

(2) para.txt

(3) codeinput.txt

…