**Huffman编码问题实验报告**

1. **实验目的**

通过实践，加深对贪心问题求解方法和霍夫曼编码问题的理解，提高编程能力。

1. **实验内容**

1.对字符串进行01编码，输出编码后的01序列，并比较其相对于定长编码的压缩率。

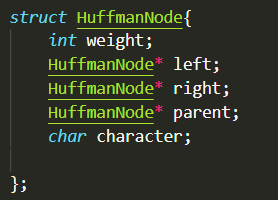
2.读取字符串文件data.txt,对文件data.txt的字符串按照Huffman编码方式编码为01序列，并输出到encode.txt文件，控制台打印压缩率。

1. **实验步骤**

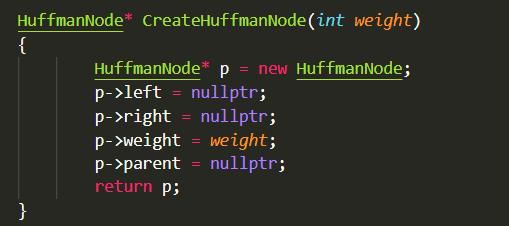
霍夫曼编码问题实现过程中最重要的部分就是根据给定字符集建立霍夫曼树，然后再根据建立的霍夫曼树生成霍夫曼编码。

1. **定义霍夫曼树节点数据结构**

如下图所示，霍夫曼树节点包括一个权值域、一个字符域和三个指针域。三个指针left、right和parent分别指向了节点的左右孩子以及父节点。叶节点的字符域中存储了该节点所代表的字符，叶节点的权值域代表了该节点所代表的字符在字符集中出现的次数。非叶节点的权值域为其左右孩子节点的权值之和， 而字符域中存储着字符’0’,指示此节点为非叶节点。



并建立CreaHuffmanNode函数用于生成一个新的节点，在生成叶节点时，统计每种字符在字符集中出现的次数，用它来初始化叶节点的权值域。生成非叶节点时，其权值为左右孩子权值之和。指针域设为空指针，字符域另外做初始化。



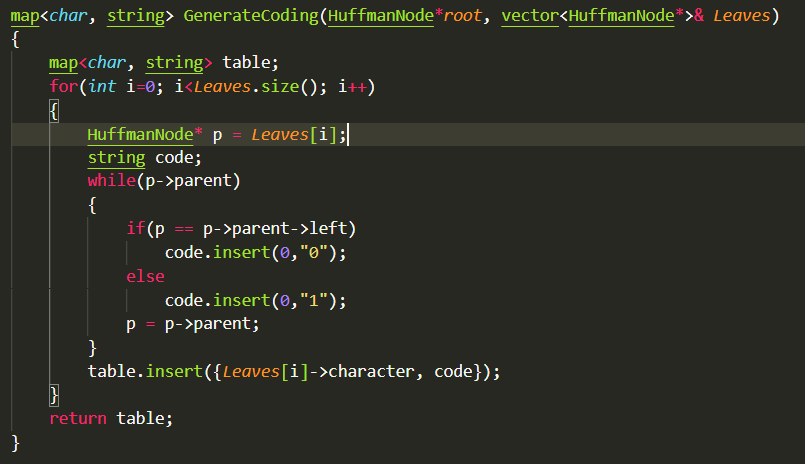
1. **建立霍夫曼树**

使用贪心算法建立霍夫曼树，使用最小优先队列存储节点指针，每次取剩余节点集中权值最小的两个节点，然后生成一个新的霍夫曼树节点，并令其权值域为刚取出的两个节点权值之和，令新生成的节点为刚取出的两个节点的父节点，同时更新两个孩子节点以及父节点的响应指针，最后再将新生成的节点的指针插入到优先队列中。

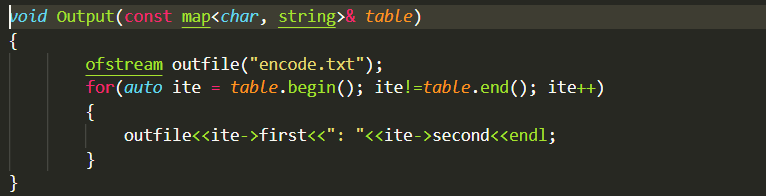


1. **生成霍夫曼编码**

根据第(2)步中生成的霍夫曼树生成霍夫曼编码，由于在生成叶节点的过程中就已经存储了所有指向叶节点的指针，因此，对每个叶节点从它自身上溯到霍夫曼树的根节点就可以生成该节点对应的霍夫曼编码，在上溯过程中，若当前结点是其父节点的左孩子，就在已生成的编码串前面加上’0’,若是右孩子则加上’1’,这样遍历到根节点即可生成编码串，并将其存储在哈希表中。



生成所有字符的霍夫曼编码串之后，再建立一个Output函数，可将字符及其对应的编码串保存到一个.txt文件中。

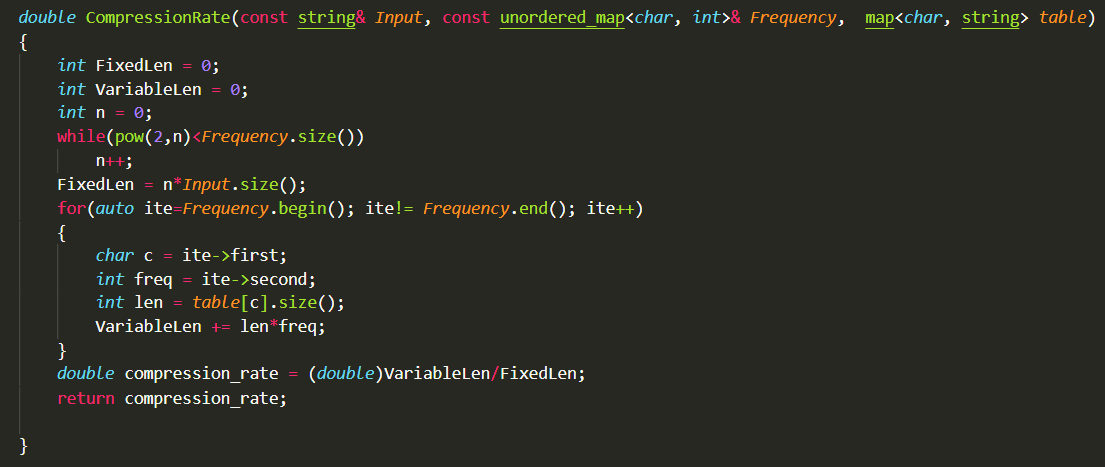


**(4)计算压缩率**

压缩率计算公式为：采用变长编码的总码串长度采用定长编码的总码串长度。

采用定长编码的总码串长度的计算方法为：先统计字符集中的字符种类N，再求出一个n满足，n即为每个字符的码长，然后用每个字符的码长n乘以N即为定长编码的总码串长度。

采用变长编码的总码串长度的计算方法为：先求出每个字符的霍夫曼编码，再用每个字符出现的次数乘以每个字符的霍夫曼编码长度，最后求和即可得到字符集的变长编码总码长。



1. **实验结果**

当输入字符集为AASMABBAAARRAABCAACCRRSNEEFF时，对应的霍夫曼编码以及压缩率为：





**五、实验心得**

在采用贪心算法建立霍夫曼树时，我用到了最小优先队列，然而我的最小优先队列中存储的并不是霍夫曼树节点，而是指向霍夫曼树节点的指针，然而我刚开始并没有意识到存储节点和存储指针的区别：区别就在于存储节点时我要重载的operator<函数是节点的operator<函数，而存储指针时我要重载的operator<函数是指针的operator<函数，两者区别很大。实际上，存储霍夫曼节点而不是指针是更常见、更好想的做法。

