离散数学课程设计

项目说明文档

最优二元树的应用

|  |  |
| --- | --- |
| 作者姓名： | 杨烜赫 |
| 学 号： | 2252709 |
| 指导教师： | 李 冰 |
| 学院专业： | 软件学院 软件工程 |



同济大学

Tongji University

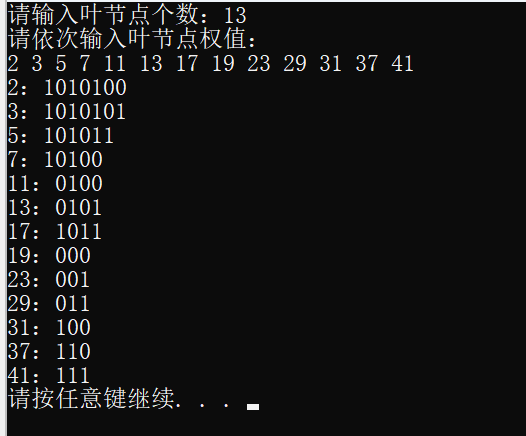
1. **题目简介**
2. **实验目的**

输入一组通信符号的使用频率，求各通信符号对应的前缀码。

**2.实验内容**

根据用户给出的通信编码使用频率，建立哈夫曼树，并求每个节点的对应二元哈夫曼编码。

**3.项目示例**

****

1. **项目实现**

## 实验原理

哈夫曼树相关概念如下：

在一棵树中，从一个结点到另一个结点所经过的所有结点，被我们称为两个

结点之间的路径。在一棵树中，从一个结点到另一个结点所经过的“边”的数量，

被我们称为两个结点之间的路径长度。树的每一个结点，都可以拥有自己的“权

重”（Weight），权重在不同的算法当中可以起到不同的作用。结点的带权路径

长度，是指树的根结点到该结点的路径长度，和该结点权重的乘积。在一棵树中，

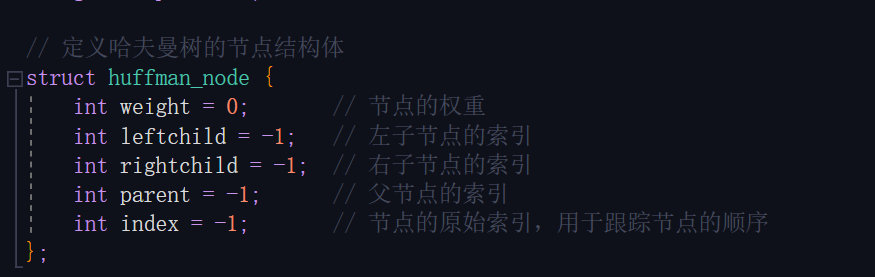
所有叶子结点的带权路径长度之和，被称为树的带权路径长度，也被简称为 WPL。而哈夫曼树（Huffman Tree）是在叶子结点和权重确定的情况下，带权路径长度

最小的二叉树，也被称为最优二叉树。

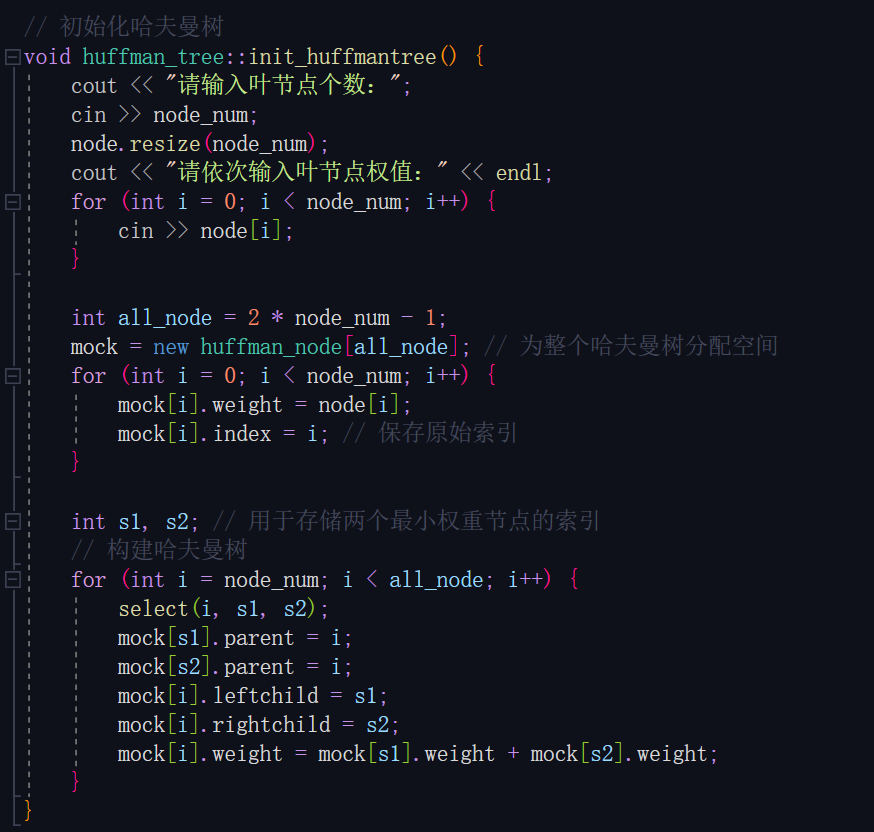
## 2.实现过程

哈夫曼树类的构建：





**具体实现如下：**

****

**初始化哈夫曼树**

**输入叶节点个数**：首先从用户那里获取叶节点的数量 node\_num。

**输入叶节点权值**：根据叶节点数量，循环读取每个叶节点的权值，并存储在 node 向量中。

**创建哈夫曼树节点数组**：创建一个大小为 all\_node 的哈夫曼节点数组 mock。这里 all\_node 为 2 \* node\_num - 1，即哈夫曼树中所有节点的数量（包括叶节点和非叶节点）。

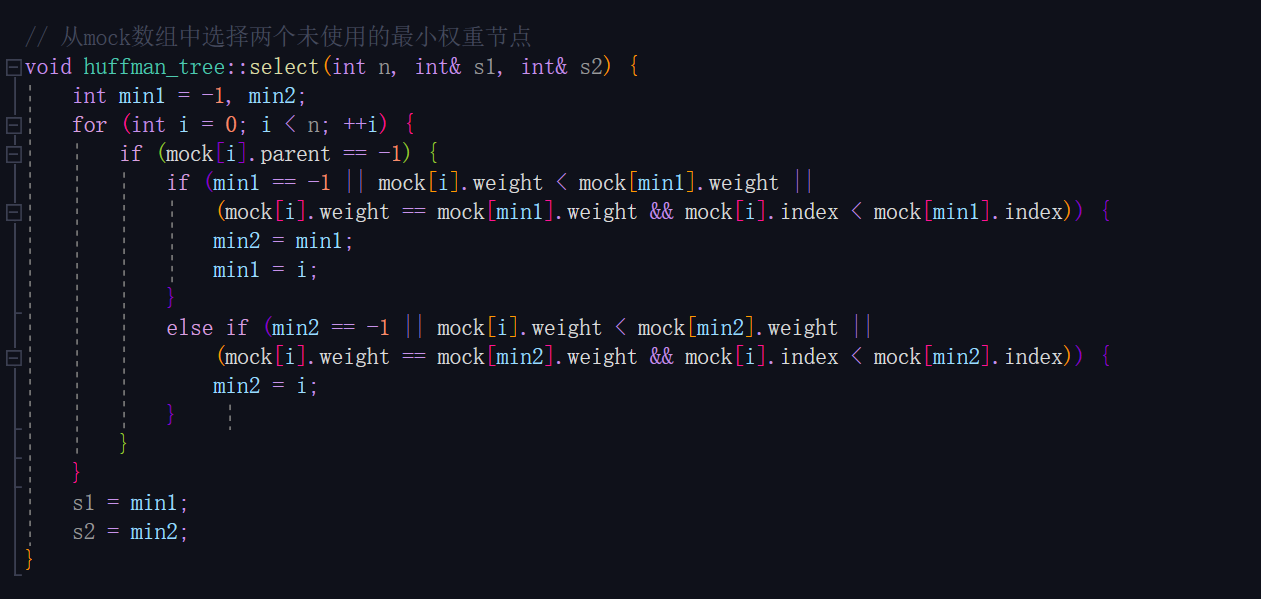
**初始化叶节点**：遍历每个叶节点，初始化它们在 mock 数组中的权重（weight）和索引（index）。

**构建哈夫曼树**

**选择两个最小权重节点**：在每次循环中，选择两个未被选取且权重最小的节点，这部分逻辑由 select 函数实现（该函数代码未给出）。

更新父节点、左右子节点和权重：更新这两个节点的父节点为新的节点 i（即合并后的节点），并设置新节点的左右子节点分别为这两个最小权重节点。新节点的权重为这两个子节点权重之和。

**循环直至构建完整树**

****循环继续进行，直到所有节点都被加入到哈夫曼树中，即 i 遍历到 all\_node。

**select 方法功能**

参数 n 表示要在前 n 个节点中进行选择。

引用参数 s1 和 s2 用于返回两个选定的最小权重节点的索引。

**方法逻辑**

**初始化最小值索引**：初始化两个变量 min1 和 min2，分别用于存储找到的两个最小权重节点的索引。初始值设为 -1，表示还未找到任何节点。

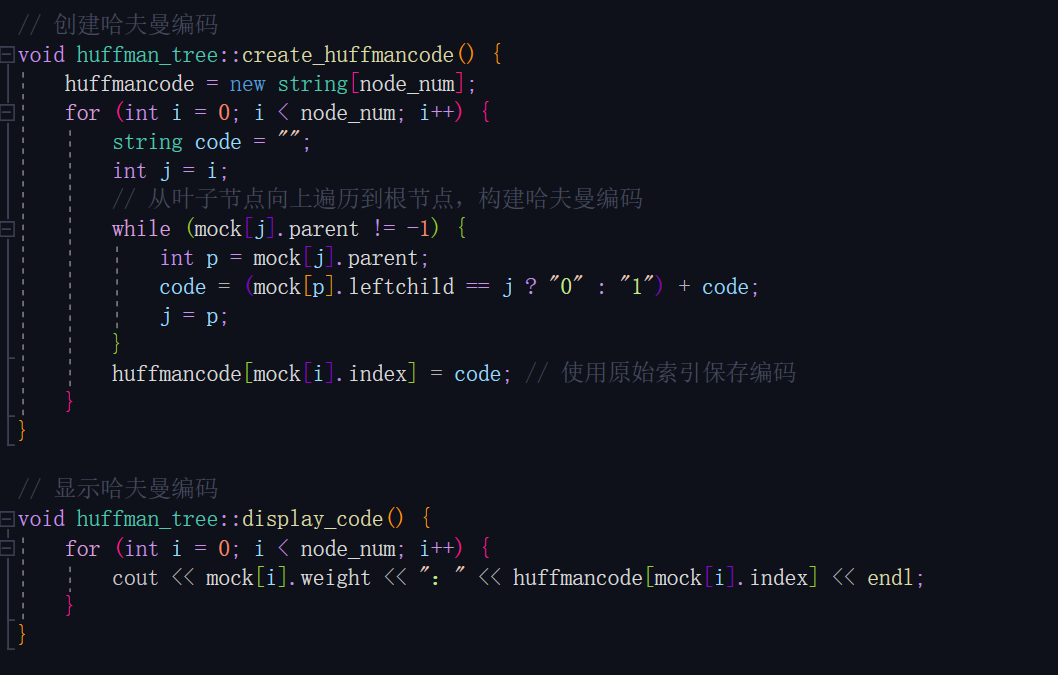
**遍历节点**：遍历 mock 数组中的前 n 个节点，寻找两个最小权重的节点。

只考虑那些还没有成为其他节点子节点的节点（即 mock[i].parent == -1）。

**更新最小权重节点索引**：如果找到一个节点权重小于当前 min1，或者权重相等但索引较小（即优先选择索引较小的节点），则更新 min1 和 min2。min1 更新为当前节点，而 min2 更新为之前的 min1。

如果当前节点权重不小于 min1 但小于 min2，或者权重相等但索引较小，则更新 min2 为当前节点。

**设置返回值**：最后将 min1 和 min2 的值赋给 s1 和 s2，返回这两个最小权重节点的索引。

****

**create\_huffmancode 方法**

功能：此方法用于为每个叶子节点生成对应的哈夫曼编码。

**实现逻辑**：

**初始化哈夫曼编码数组：**创建一个字符串数组 huffmancode，大小与叶子节点数 node\_num 相同，用于存储每个节点的哈夫曼编码。

**为每个叶子节点生成编码**：遍历所有叶子节点，从每个叶子节点开始向上遍历至根节点，沿途构建哈夫曼编码。如果当前节点是其父节点的左子节点，则在编码字符串前添加 "0"；如果是右子节点，则添加 "1"。编码过程是从叶子节点向根节点反向进行的，因此编码字符串是逆序拼接的。

**保存编码：**使用节点在 mock 数组中的原始索引将编码保存在 huffmancode 数组中，确保编码的顺序与输入顺序一致。

**display\_code 方法**

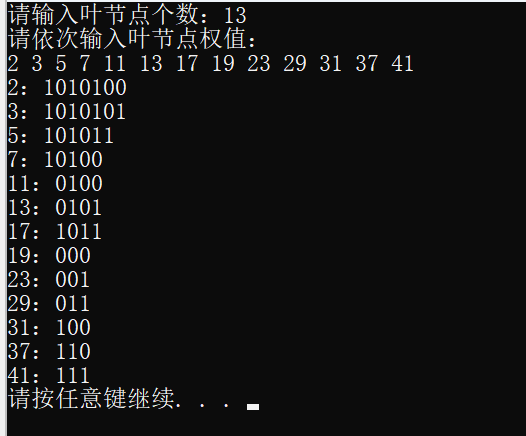
功能：此方法用于显示每个叶子节点及其对应的哈夫曼编码。

**实现逻辑**：

遍历所有叶子节点，并打印每个节点的权重及其哈夫曼编码

**三、项目测试**

结果测试：

****

**实际值与理论值相同**

**四、心得与总结**

在实现哈夫曼编码的实验中，我深入了解了数据压缩和编码理论，并在实践中应用了这些知识。通过构建哈夫曼树并为每个叶子节点生成编码，我深入理解了哈夫曼编码的原理。哈夫曼编码通过给频率高的符号分配较短的编码，频率低的符号分配较长的编码，实现了有效的数据压缩。实现哈夫曼编码过程中，我提高了对树结构、递归方法和字符串操作的掌握。同时，这个过程还锻炼了我的逻辑思维和问题解决能力，特别是在处理编码的生成和逆向构建过程中。总之，这次实验是对我的编程技能、离散数学理论知识和算法理解的全面提升。通过实际编写和测试哈夫曼编码算法，我不仅加深了对相关理论的理解，还提高了解决复杂编程问题的能力。这些经验对我未来在更高级算法和离散数学理论领域的学习和应用将是宝贵的财富。