

《数据结构》上机实验报告

第 4 次上机

学号： 202011140104

姓名： 李馨

学院： 物理学系

专业： 物理学

教师： 郑新

日期： 2022.10.24

1. 实验要求

1.上机之前应做好充分准备，认真思考所需的上机题目，提高上机效率。

2.独立上机输入和调试自己所编的程序，切忌抄袭、拷贝他人程序。

3.上机结束后，整理出实验报告。书写报告时，重点放在实验的方法、思路以及总结反思上，以达到巩固课堂学习、提高动手能力的目的。

1. 实验过程
   1. 问题描述：

利用最小堆编程实现给定权值集合下构造相应霍夫曼树的算法，并解决以下问题：

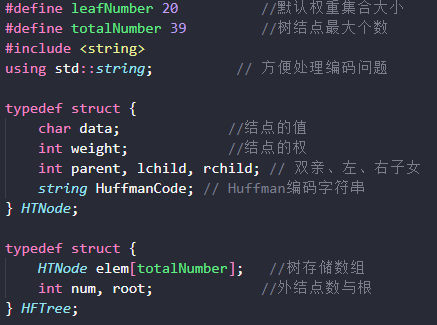
有一电文共使用五种字符a, b, c, d, e, 其出现频率依次为4, 7, 5, 2, 9。

(1) 构造对应的编码哈夫曼树(要求左子树根结点的权小于等于右子树根结点的权)。

(2) 给出每个字符的哈夫曼编码。

(3) 译出编码系列11000111000101011的相应电文。

* 1. 问题一：利用最小堆构建编码哈夫曼树（Huffman.h, Huffman.cpp）
     1. **哈夫曼树定义（**Huffman.h**）**



* + - 1. 树的储存结构定义为静态三叉链表。
      2. 树中外结点有Huffman编码，为了方便后续字符串处理，使用C++标准库中的string类类型。
    1. **利用最小堆构造哈夫曼树（**void createHFTreeWithHeap()**）**

构造哈夫曼树的算法描述如下：

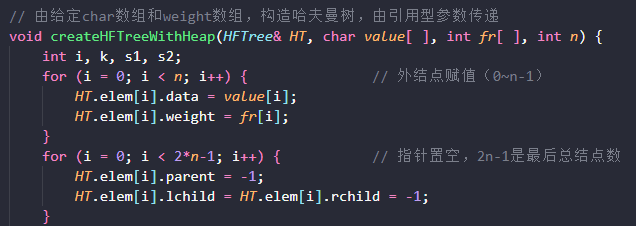
由给定n个权值{w0, w1, w2, …, wn-1}，构造具有n棵二叉树的森林F = { T0, T1, T2, …, Tn-1 }，其中每棵二叉树Ti只有一个带权值wi的根结点, 其左、右子树均为空。重复以下步骤, 直到F中仅剩一棵树为止：

(1)在F中选取两棵根结点权值最小的二叉树, 做为左、右子树构造一棵新的二叉树。置新的二叉树的根结点的权值为其左、右子树上根结点的权值之和。

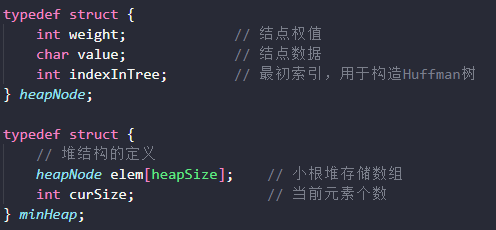
(2) 在F中删去这两棵二叉树。

(3) 把新的二叉树加入F。

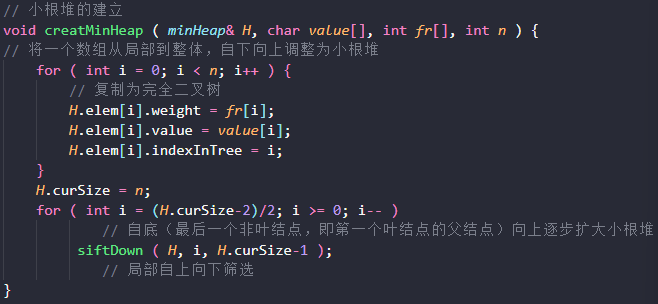
* + - 1. 首先先由给定的结点数据和权值对最初的森林进行赋值，并进行指针置空：



* + - 1. 选取最小权值和次小权值根结点，可以使用最小堆。
         1. 最小堆结构定义如下（minHeap.h），其结点数据是结构体，储存了权值、数据和在Huffman树中的地址（构造最小堆和构造哈夫曼树时，所用原始value数组的索引数）。



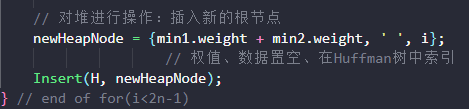
* + - * 1. 建立最小堆的函数（minHeap.cpp）：



* + - 1. 不断选取最小权值和次小权值根结点的二叉树构造新二叉树，被构造的根节点是内结点，在数组中的地址是n到2n-2：



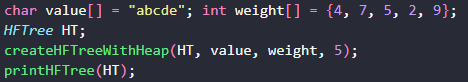
* + - 1. 对堆进行操作，插入新二叉树的根节点：



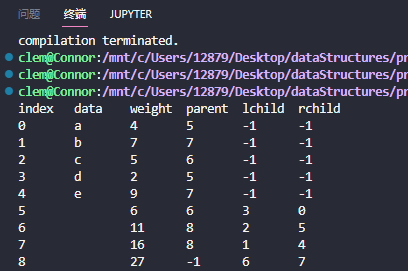
* + - 1. 循环结束后构造完毕，再设置外结点数与根地址：



* 1. 问题一：运行结果（main.cpp）
     1. 给定数据weight和权值weight，进行构造，测试代码（所用打印函数在Huffman.cpp中）：



输出结果为：



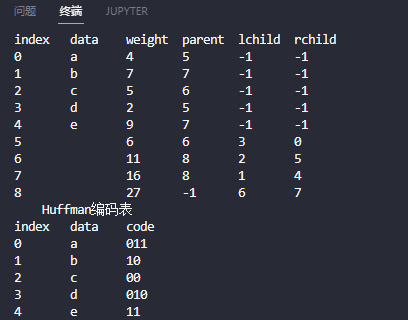
* 1. 问题二：外结点哈夫曼编码（Huffman.cpp: void encodeHuffman()）
     1. **算法描述**
        1. 考察每一个外结点（索引i=0~n-1，for循环），向上追溯其双亲，判断（if）是双亲的左孩子还是右孩子。若是左孩子，在初始为空的temp字符串的首位插入0，右孩子则插入1。
        2. 插入后再追溯其双亲的双亲，进行同样的判断和对字符串的操作，直至追溯至根节点（while循环）。
     2. **代码部分**



* + - 1. 编码通过引用型参数储存到被引用的哈夫曼树的每个外结点中，即HT.elem[i].HuffmanCode。
      2. 方便起见，用到了C++的string类的很多操作，比如“在字符串首位插入”这一操作用了C++特有的加号的运算符。
  1. 问题二：运行结果（main.cpp）
     1. 沿用上一问的哈夫曼树，对之进行编码，测试代码为：



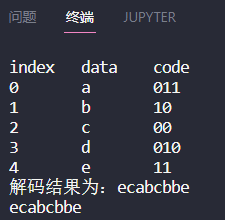
* + 1. 运行结果为：



* 1. 问题三：翻译编码（Huffman.cpp: void decodeHuffman(HFTree, string)）
     1. **算法描述**
        1. 首先对给定哈夫曼树进行编码，即调用上一问encodeHuffman函数，将编码储存进该树中。
        2. 然后对字符串进行从前到后的扫描，与树中哈夫曼编码进行匹配，匹配成功后删除被匹配的字符串区间，再继续扫描，直到字符串为空，结束扫描，返回结果字符串。
     2. **代码部分**



* + - 1. “匹配”和“删除”操作使用了string类的find和erase函数。
  1. 问题三：运行结果（main.cpp）
     1. 沿用上一问的哈夫曼树，对之进行解码，测试代码为：
     2. 运行结果为：



三、总结（实验中遇到的问题、取得的经验、感想等）

* 由于哈夫曼树结点和最小堆的结点是两种不同的结构体，在运用时互通会有一些麻烦。比如，找到了最小权值点和次小权值点后突然发现，我们并不知道它们在树中的位置。
  + 这里选择要求在建立哈夫曼树和最小堆时使用相同的数组，根据这一原始数组的索引实现互通。
    - 哈夫曼树的结点中外结点位置索引相较原数组不改变；但在最小堆的结点里会反复重新筛选，位置索引发生改变。
    - 故而在最小堆结点的结构体中储存原始数组索引信息，也即是在哈夫曼树中的索引。
* 实验后两问中为了方便字符串的处理大量运用了C++的string类的方法。
* 在实验中加强了对哈夫曼树构造和编码过程、最小堆的应用的理解和把握。