EX6说明文档及学习心得

构造哈夫曼树的逻辑自己想起来不难，但是用代码实现还是有点麻烦。用到了扩充二叉树及其相关算法、最小堆及其相关算法。每次调试的时候就会看到最小堆（完全二叉树）的结点是扩充二叉树，扩充二叉树的结点的data才是真正的weight，层层嵌套阅读起来特别有障碍。除了这个两个比较重要需要具体实现，另外比较麻烦的是要考虑new和delete的调用位置。然后没想到最后解决不了的是另个玄学一样的问题（在后面提及）。

1. 首先构建n个只有根结点的扩充二叉树。扩充二叉树没有或只有两个孩子，正好适合将两棵树合成一棵新树，因此重载了构造函数，用于将两棵已用的树创建新的树。在这个构造函数里面要new一个根结点root。
2. 然后先定义一个最小堆（以扩充二叉树为结点），将上述n个扩充二叉树加入最小堆中，用creatHeap()先初始化一遍最小堆。之后每次都重复过程：要取出两个最小的结点，将其在最小堆中删除，再用他们构造新的树，将其加入最小堆中，直到最小堆中剩下一棵树—根结点。这里面涉及到最重要的是adjustDown(int r,int j)函数。r代表调整的结点下标，j代表最后一个结点的下标。这里是假设第r+1个结点到第j个结点已经满足最小堆序，实际上只需要调整第r个结点。因此在creatHeap()中是要用一个循环调用adjustDown (size-1)/2次的就是从最后一个非叶子结点的结点一直调整到根结点。adjustDown具体是先将需要调整的结点赋值给一个临时的变量，因为是要和孩子结点中较小的一个比较，因此先让左右孩子（需要判断有没有右孩子）比较，然后再比较，交换。
3. 关于delete的问题。此处就要考虑什么时候delete这个root，一开始想在把新树放入最小堆后就delete，但是发现由于所以的赋值都是直接浅拷贝，复制地址，因此实际上所有new出来的结点都只有一份，所有的结点都是最后的哈夫曼树的结点，所有的扩充二叉树都是最小推的结点。因此最后用非递归的后序遍历封装成breakTree()函数，将哈夫曼树（扩充二叉树）结点全部delete之后，再将最小堆结点（扩充二叉树）全部delete就可以了。
4. 然后就是编码过程。用hcodes[n][\*]记录weights[n]的编码。由于要按weights数组的顺序依次找到对应的叶子结点，依次从下往上编码。先遍历哈夫曼树找到对应的叶子结点，再依次往上找到它的父亲结点直到根结点，根据是父亲结点的左孩子还是右孩子来给hcodes[n][i]赋值’0’或’1’。后来考虑到可能有权值相同的结点，因此给树的结点多加了一个成员变量char c来作为判断依据（这里的结点直接就在BinTreeNode里写了，实际上应该多写一个HuffmanNode继承于它加这个成员变量，否则对于别的结点来说char c完全用不到就是空间浪费，但是由于时间仓促就直接写了。）
5. 打印过程需要把hcodes[n][\*]倒过来打印，用’\0’记录结束。
6. 最后收尾delete所有new过的东西。
7. 关于玄学一般的问题：weights[],c[] 在对最小堆的操作过程中会莫名的被修改最后一个元素，debug了很久也找不到问题，后来无奈只能新加两个局部变量，而且必须写在first、second的声明下面，否则又会影响first、second（不知道为什么）；除此之外就是cout<<endl；会有运行时异常，也找不到原因，最后只能改成printf()；可能是我学艺不精不知道哪里出问题了。如果助教或者老师知道的话，能否为我解答一下？