7-1：

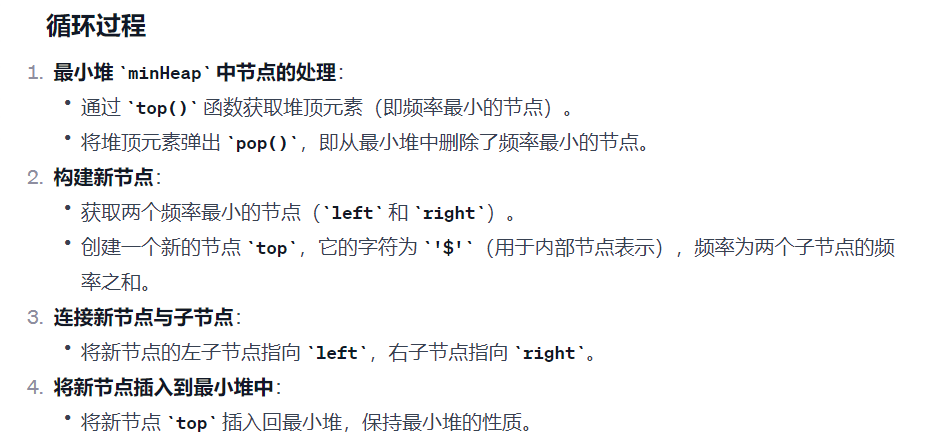


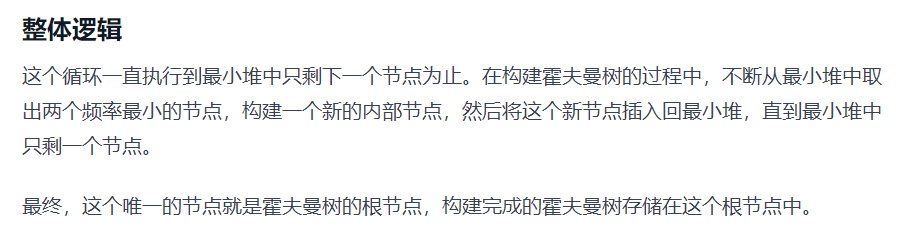


7-2：



构建霍夫曼树：





解码过程：

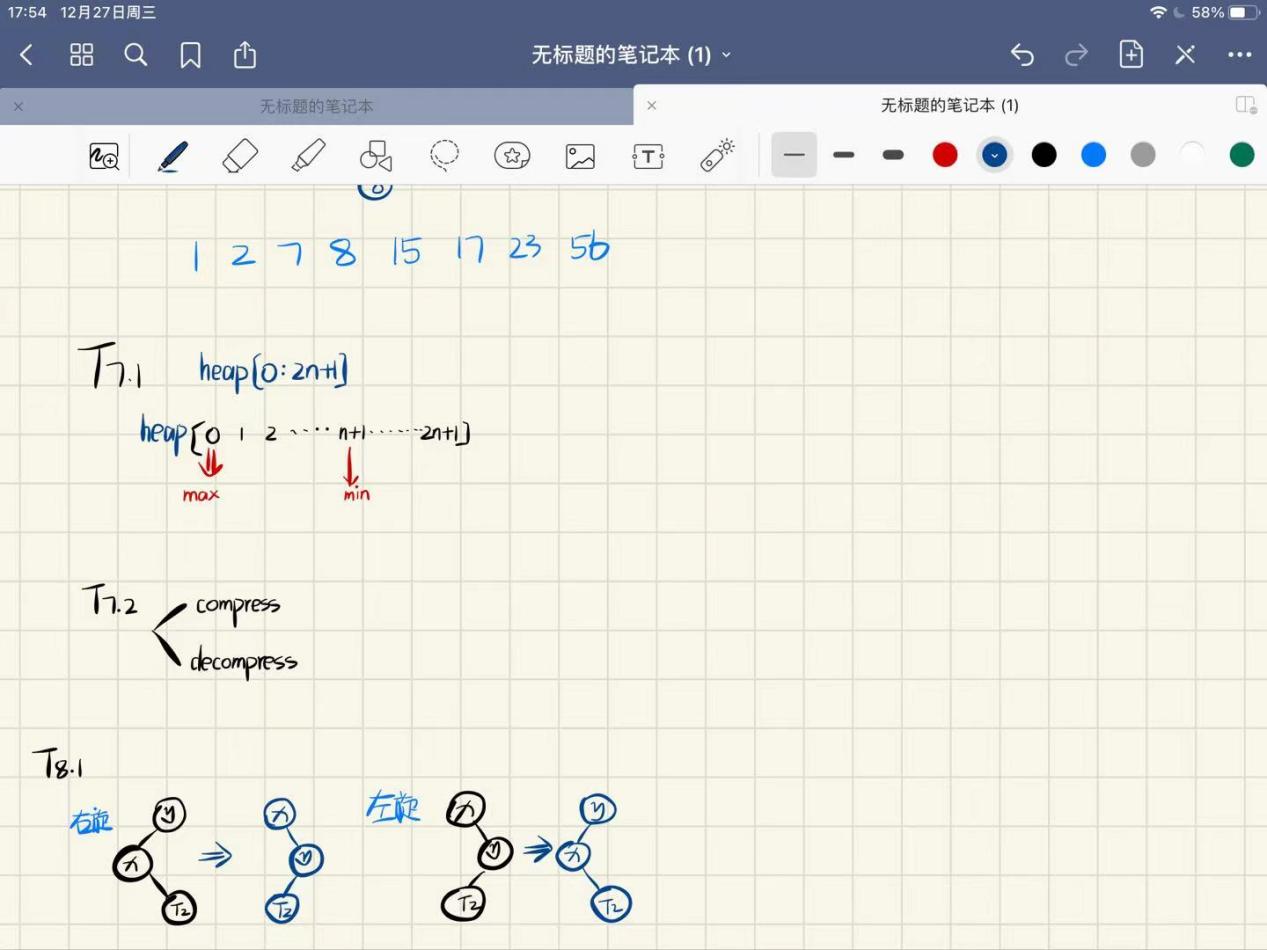




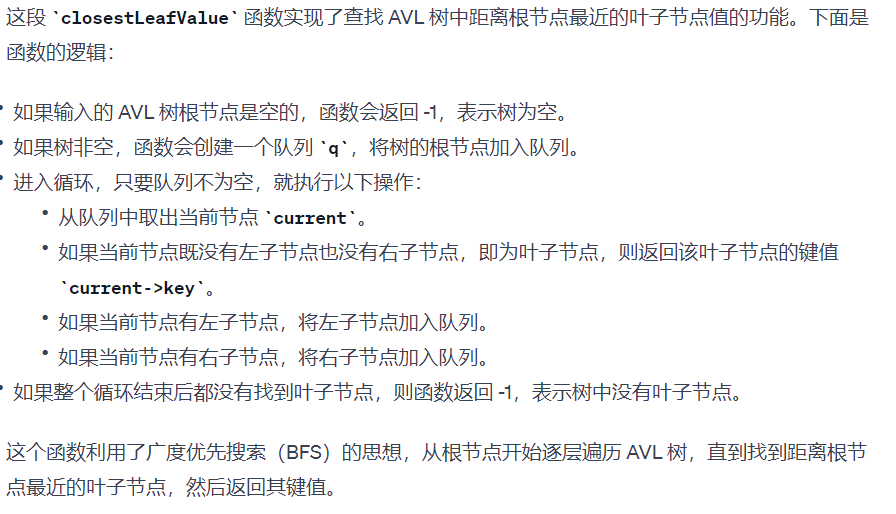
8-1：

在这种特定的实现中，**getHeight** 函数的时间复杂度为 O(1)，因为 AVL 树的每个节点都维护了一个高度属性。这个高度属性在节点的插入、删除操作时被更新。因此，获取树的高度只需要访问根节点的高度信息即可，而不需要遍历整棵树。这样的设计使得获取 AVL 树的高度的时间复杂度保持为常数时间。

首先，在 AVL 树中，每个节点的高度信息是由其子树的高度决定的。而在插入或删除节点时，AVL 树会通过旋转操作来保持平衡。在旋转操作中，节点的高度会被更新以反映其子树的新高度。因此，当需要获取 AVL 树的高度时，只需获取根节点的高度属性即可，这个操作的时间复杂度是 O(1)，不受树的大小影响。



8-2：

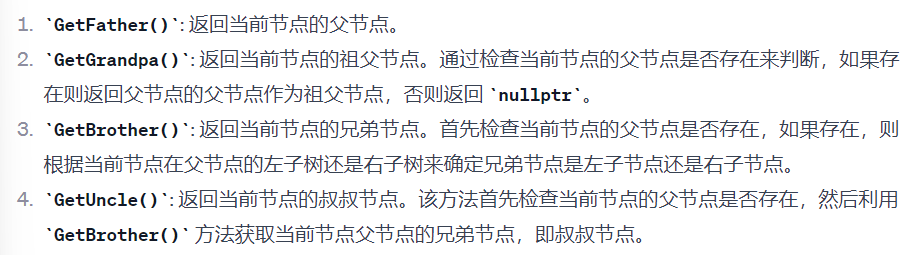


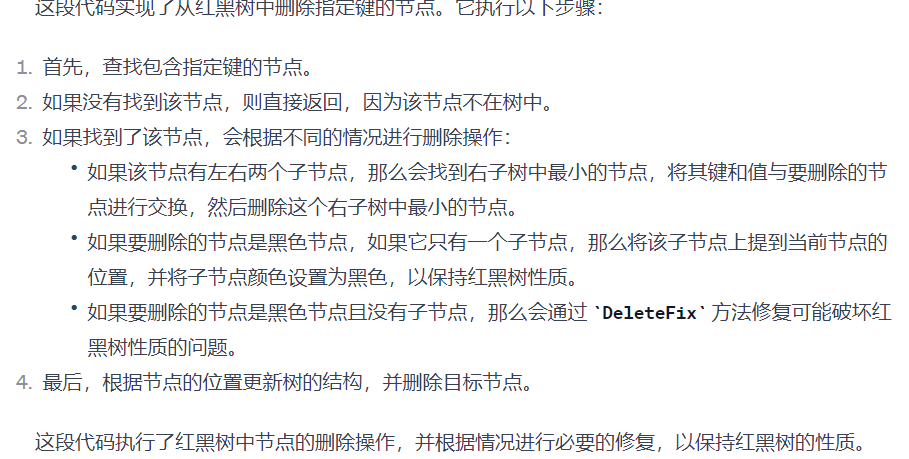
9-1：

一个键值对一个value

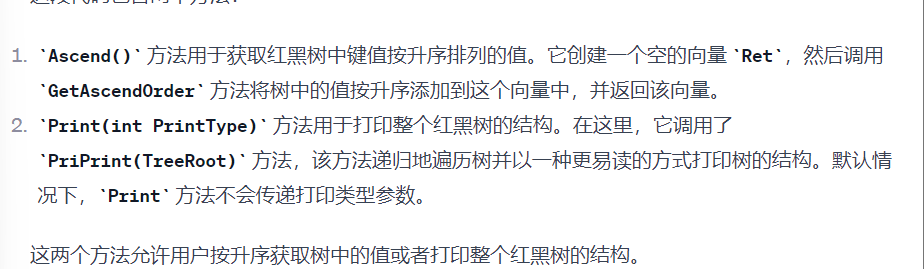
使用向量存储value



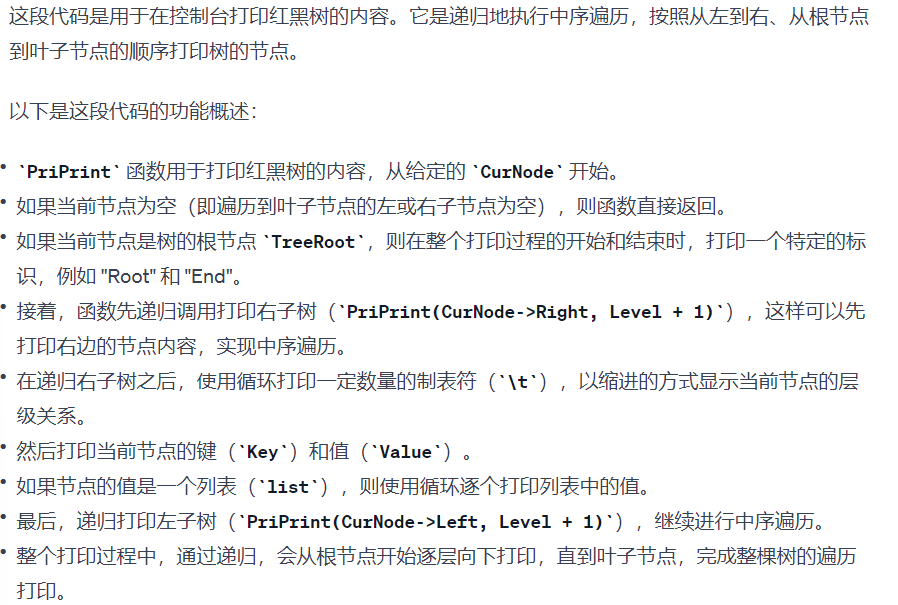


删除：

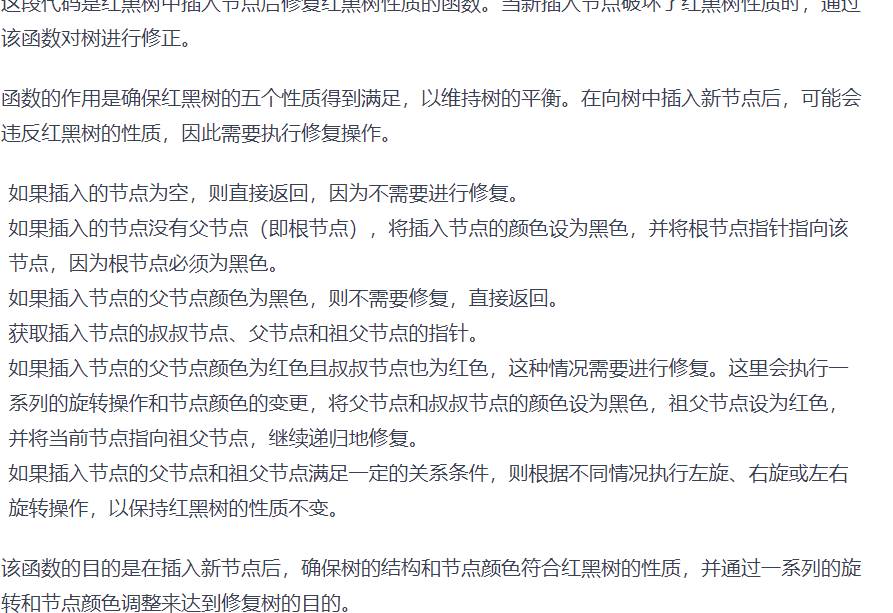
升序获取：



打印：



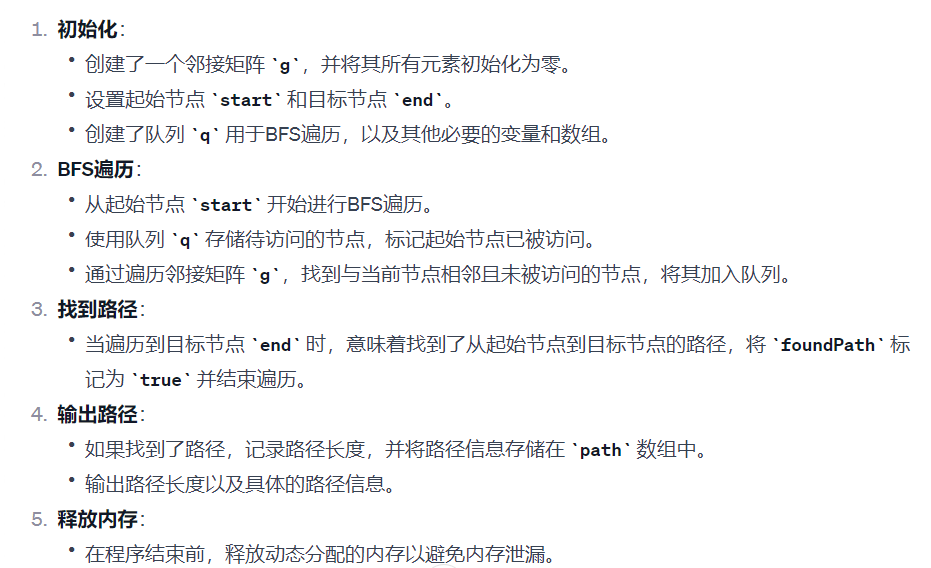
插入后修复：



删除修复：



10-1：



10-2：

