之前在考虑搜索时，由于对于插入操作的理解不充分忽略了，在插入重复键值并导致分裂，且重复值正好被分割在左右两侧的情况。

如果发生这种情况，意味着进行键值比较时，如果大于或等于向右走，由于叶节点是单向链接，会导致丢失前面拥有相同键值的元素。导致搜索结果不完整。虽然这种问题并不会影响插入。但在删除操作中，搜索结果不完整就会导致删除不完全，这是一种严重的错误。

为了不发生问题，搜索的移动条件从原来的的等于向右走改为等于向左走。这样虽然需要在到达叶节点后继续顺序遍历找到相应键值的起始位置，但保证了搜索结果的完整性。

在考虑进行删除工作时，由于键值有重复，而删除操作正常需要对数据和所有其相应的索引进行修改，对于所有重复键值需要遍历寻找到特定的键值指针，实现有困难。所以，如果要仅拥有删除效果，我们选择了一个假的删除。即不修改任何索引树，而是重新再每条数据前设计了一个条目头由于存储其是否被删除。如果，其条目头显示被删除，则我们在输出时，便会刻意跳过这些数据。