实验报告

问题1

AVL树

基本思想：
在普通平衡搜索树的基础上，定义左子树和右子树高度之差为平衡因子，并要求abs(平衡因子) < = 1，即左右子树的高度差为0、1和 -1，即成为AVL树。
设计细节：
在插入和删除节点后，通过单旋和双旋操作使树恢复平衡，也可以使用统一的重平衡算法完成。

红黑树

基本思想：
相较于普通的AVL树，红黑树在每个节点增加了一个存储位记录节点的颜色，可以是红,也可以是黑；通过任意一条从根到叶子简单路径上颜色的约束，红黑树保证最长路径不超过最短路径的二倍，因而达到近似平衡（最短路径就是全黑节点，最长路径就是一个红节点一个黑节点，当从根节点到叶子节点的路径上黑色节点相同时，最长路径刚好是最短路径的两倍）。它同时满足以下特性：
1.节点是红色或黑色
2.根是黑色
3.叶子节点（外部节点，空节点）都是黑色
4.红色节点的子节点都是黑色
5.从任一节点到叶子节点的所有路径都包含相同数目的黑色节点

设计细节：
插入节点时，由于将其染成红色，故需要进行双红修正，可能出现一下几种情况：
1.没有出现双红
直接判断并返回即可。
2.父亲为红色而父亲的兄弟为黑色(RR-1)
对节点x、p和g及其四棵子树做一次connect34重构。
3.父亲为红色而父亲的兄弟也为红色(RR-2)
与2操作相同，但修正后上溢可能向上传播，故累计最多迭代O(logn)次。

删除节点时，同样要考虑双黑修正的几种情况：
1.删除节点的兄弟为红色(BB-1)：
旋转其父亲p，将兄弟b伸展到p位置，然后染黑b、染红p，于是将问题转化到了BB-2R或BB-3。
2.兄弟和父亲都为黑色，且兄弟没有红儿子(BB-2B)
染红兄弟b，使问题转化为了BB-1、BB-2B、BB-2R或BB-3中的一个。
如果被修正节点为根节点，则直接返回即可。
3.兄弟是黑色，没有红儿子，父亲为红色(BB-2R)
染红兄弟b，染黑父亲p即可。
4.兄弟是黑色，有红儿子(BB-3)
如果兄弟b的儿子c是与父亲p同向，旋转p使兄弟b伸展到p位置，并将b染为p的颜色，p、c染黑即可；反之则伸展c到p的位置，并将c染为p的颜色，然后染黑p即可。

其他功能函数按照题目要求实现即可。

问题2

考虑使用非旋treap(FHQtreap)来完成此题。
FHQtreap是一个二叉搜索树，它的每个节点有两个主要信息：key和val，key是我们fhq-treap要维护的键值，而val是随机生成的数,key信息主要用于我们对于题目信息的处理，而val保证了fhq-treap拥有稳定的logn的高度，不会像splay一样退化。FHQtreap 除了要满足关于key的BST性质之外，还需满足关于val的小根堆性质。

pushdown函数

由于本题是交换区间，而我们若在分裂出对应的树后，从根开始，每一层的每一个节点都将两个儿子调换需要很大的复杂度，因此我们可以引入懒标记的思想来标记该节点是否被翻转(记该节点的lazy值为0或1)。该函数即先交换节点左右子树，再将左右子节点懒标记与1异或(这样后1变0,0变1)，最后该节点懒标记变成0。

split函数

对于一个元素x,我们会将这棵fhq-treap分裂为左右两棵树，左树上每个节点的key值都小于等于x，而右树上的所有节点的key值都大于x，注意分裂前要调用pushdown函数下放懒标记。

merge函数

如果有空子树的话，直接返回即可。若两子树都非空，则比较两子节点的key，当且仅当左子树上所有节点的key值都小于等于右子树中的最小key值才可以合并，注意合并前先下放懒标记。

trav_ln函数

递归地中序输出结果。

参考资料

https://blog.csdn.net/cy973071263/article/details/122543826
https://blog.csdn.net/weixin_42102584/article/details/90138269