Leftist Tree

左偏树

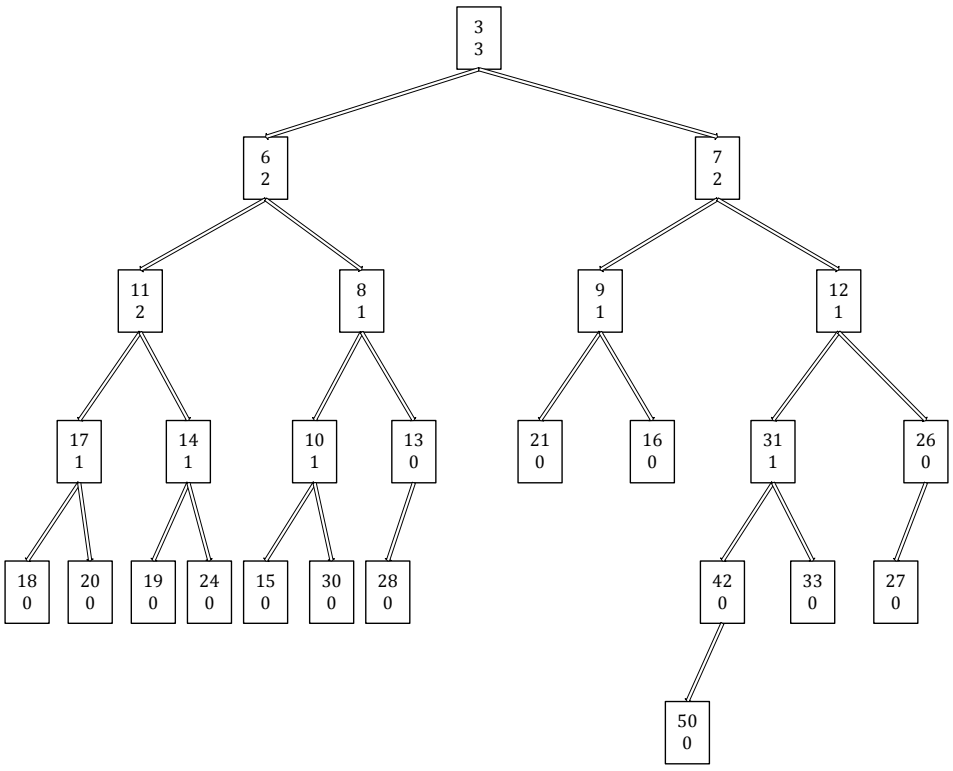
描述：

左偏树是一种接近于堆的二叉树，它的根节点总是树中的最小值或最大值。两个小根堆（或大根堆）无法快速合并，而左偏树可以支持快速合并。本文只考虑根节点为最小值的左偏树。

左偏树的主要操作包括（1）合并两个左偏树；（2）在左偏树上插入新节点；（3）查找最小值（左偏树中根节点即为最小值）；（4）删除最小值（根节点）。其中（2）-（4）操作依赖于（1），因此合并操作是左偏树的核心操作。

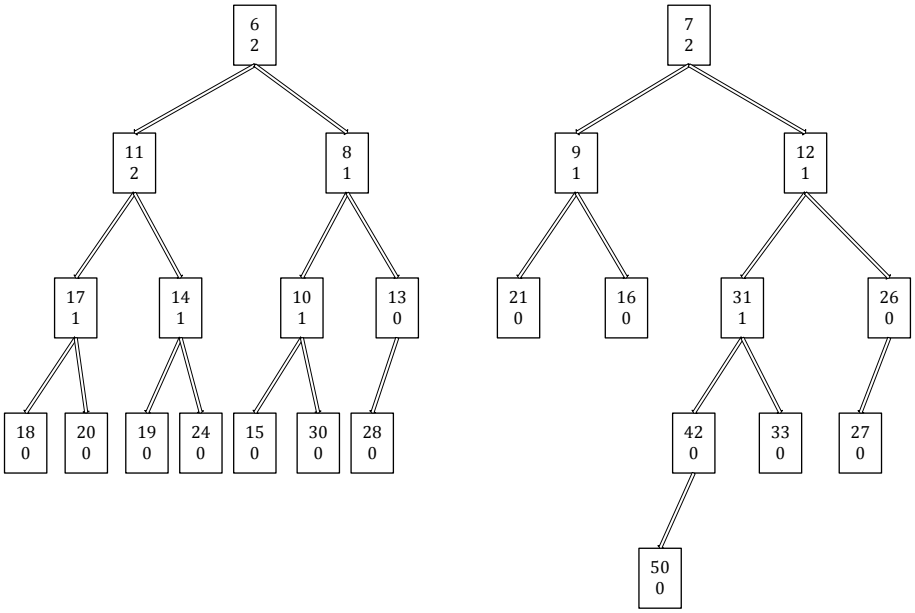
左偏树中，为了保证数据的有序性，父节点的值总是小于左右孩子节点的值，即$father \leq father.left_child$且$father \leq father.right_child$。并且节点的左孩子节点的值总大于右孩子节点的值，即$father.left_child \geq father.right_child$。

定义左偏树中一个节点的距离d，为该节点递归的向右下一直到叶子节点的边的数量。如下图：

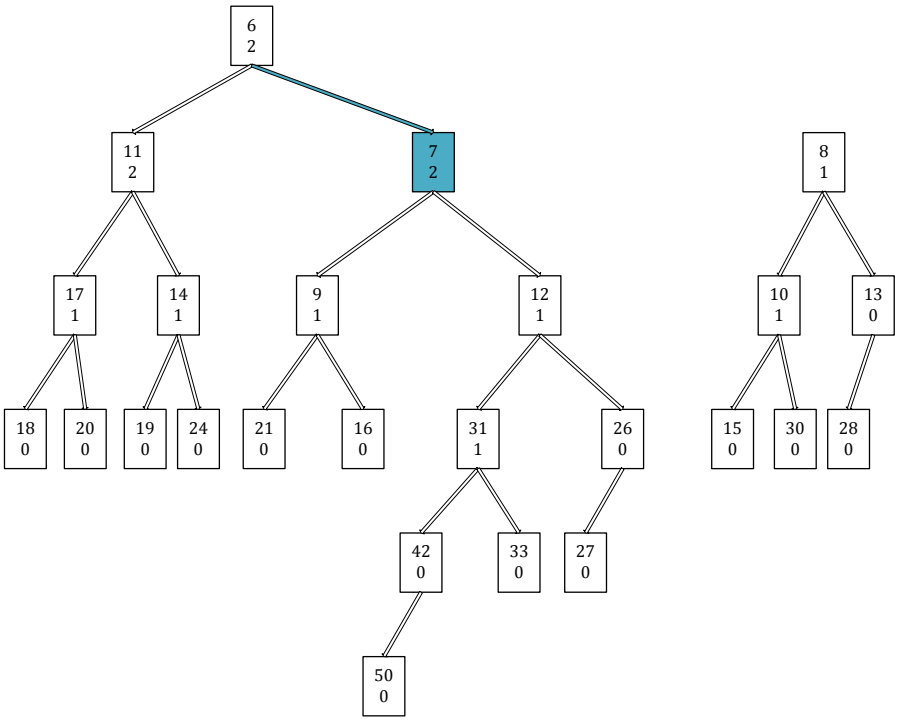


其中每个节点上面的数字为该节点的标号i，下面的数字为该节点递归向右下角，到达叶子节点的距离d。可以得到结论：（1）$father.d = father.right\_child.d + 1$；（2）$father.left\_child.d \leq father.right\_child.d$。

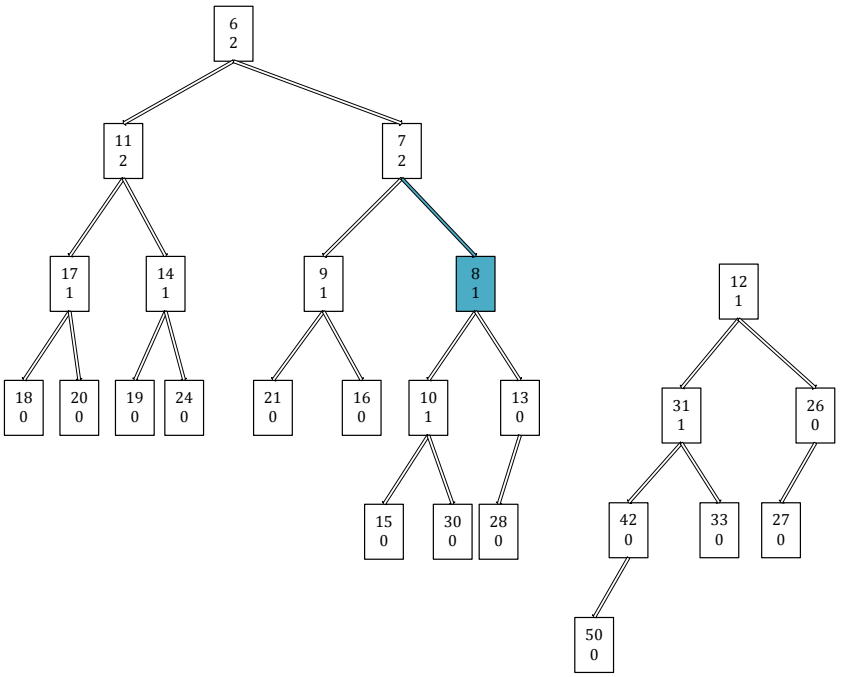
合并下图中的两个左偏树，用根节点来命名一个树，下面两个树即为树 6 和树 7：



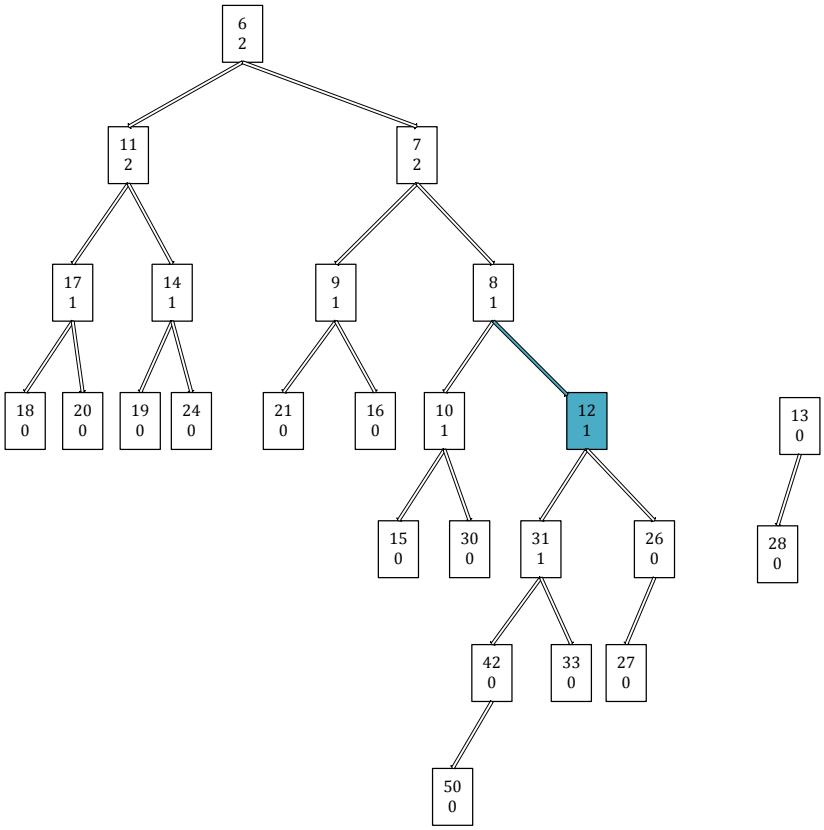
(1) 比较两树根节点的值$6 < 7$，节点 7 沿着节点 6 向右下寻找第 1 个满足$7 < x$的节点x，替换x作为节点 6 的新右孩子节点。该节点为节点 8（$7 < 8$），节点 6 的原右孩子节点 8 暂时脱离；



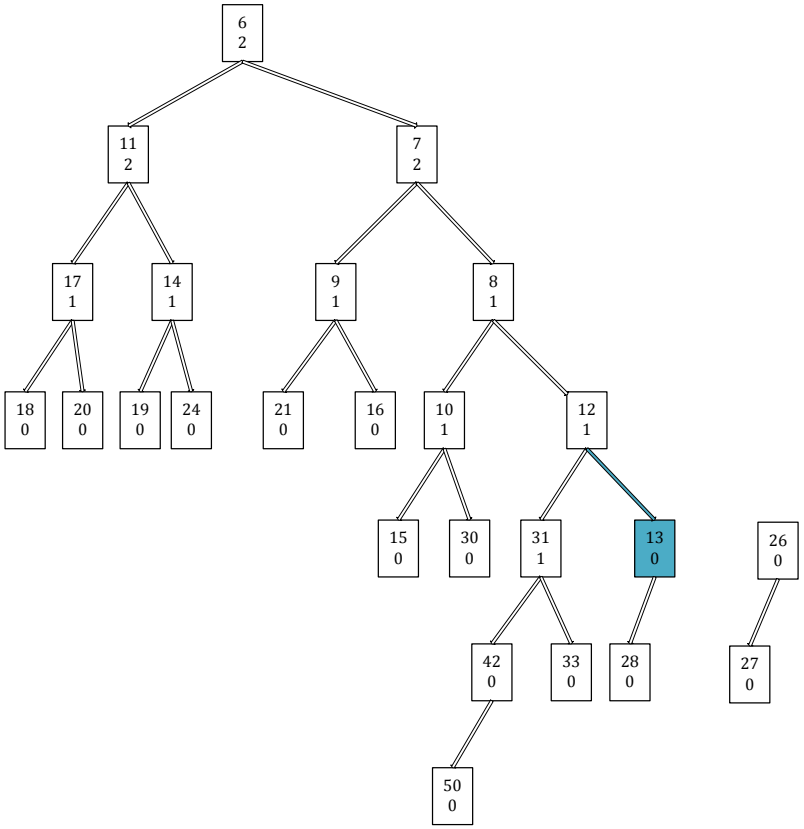
(2) 节点 8 沿着节点 7 向右下寻找第 1 个满足$8 < x$的节点x，替换x作为节点 7 的新右孩子节点。该节点为节点 12（$8 < 12$），节点 7 的原右孩子节点 12 暂时脱离；



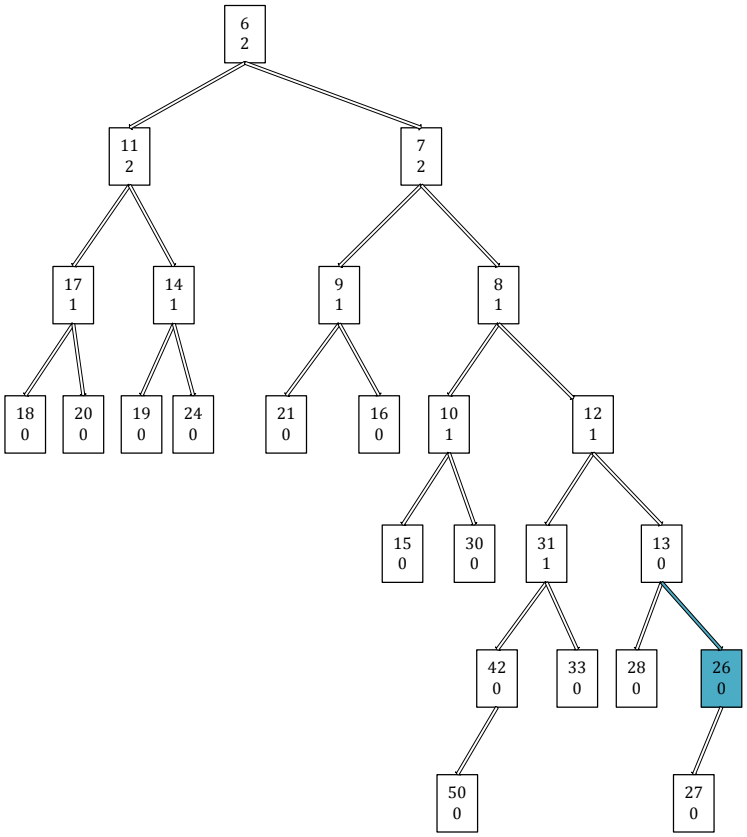
(3) 节点 12 沿着节点 8 向右下寻找第 1 个满足$12 < x$的节点x，替换x作为节点 8 的新右孩子节点。该节点为节点 13（$12 < 13$），节点 8 的原右孩子节点 13 暂时脱离；



(4) 节点 13 沿着节点 12 向右下寻找第 1 个满足$13 < x$的节点x，替换x作为节点 12 的新右孩子节点。该节点为节点 26（$13 < 26$），节点 12 的原右孩子节点 26 暂时脱离；



(5) 节点 26 沿着节点 13 向右下寻找第 1 个满足$26 < x$的节点x，节点 13 没有右孩子节点，因此节点 26 直接成为节点 13 的右孩子节点，不再需要替换，合并操作结束；



左偏树插入新节点的操作，可以看作左偏树与一个只有根节点的左偏树的合并操作；删除根节点的操作，可以看作删除根节点后，左右子树的合并操作。

左偏树的合并操作、插入节点操作、删除根节点操作的时间复杂度都为$O(log_2 N)$。