Leftist Tree（Leftist Heap）

左偏树（左倾堆）

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。与堆不同的是，两个堆的合并需要遍历被合并的堆中所有元素，依次插入另一个堆中，而左偏树可以支持更快速的合并操作。本问题只考虑根节点为最小值情况的左偏树。

左偏树的主要操作有(1)合并两个左偏树；(2)插入新节点；(3)查找最值；(4)删除最值。其中(2)-(4)的实现依赖于(1)，合并操作是左偏树的核心。

定义左偏树中一个节点的距离，其值为该节点到最右下的叶子节点的距离。左偏树具有以下性质：

1. 父节点键值小于等于其左右孩子节点的键值，且；
2. 父节点的左孩子节点的距离大于等于右孩子节点的距离，；
3. 父节点的距离等于其右孩子节点的距离加1，；
4. 具有个节点的左偏树的根节点的距离小于等于，；

下图中的每个节点中，上面的数字代表节点的下标号，下面的数字代表该节点的距离：



合并两个左偏树的操作，可以简单的看作(1)递归向下合并子树的根节点；(2)递归向上更新所有被修改过的节点的距离。我们通过合并下面两个左偏树来进行演示：



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



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



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



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



1. 节点26沿着节点13向右下寻找第1个满足的节点，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；



向右下插入节点的操作会影响到左偏树的平衡性，右子树变得越来越庞大。而且所有节点的距离也是错的（没有更新）。实际上每一步合并操作后还需要检查左右子树的距离属性：(1)对于的情况，交换左右子树；(2)对于的情况，更新。

节点距离的更新必须从叶子节点开始向上进行，对于之前步骤中修改过的节点，重新遍历计算其距离（其中空节点的距离认为）：

1. 对于节点26，，不需要交换左右子树，；



1. 对于节点13，，不需要交换左右子树，；



1. 对于节点12，，不需要交换左右子树，；



1. 对于节点8，，需要交换子树10和子树12，；



1. 对于节点7，，需要交换子树9和子树8，；



1. 对于节点6，，不需要交换左右子树，；

实际编码时可以通过递归的方式将合并子树和更新距离两个操作放在同一个函数中，合并函数Merge返回合并后左偏树的根节点的距离，在Merge中调用左右孩子的合并操作，获取左右孩子的距离，然后再决定是否交换左右子树，并更新父节点的距离。本文的将两个步骤分开是为了方便理解算法。

左偏树的插入操作，可以看作左偏树与一个只有根节点的左偏树的合并操作；删除最值的操作，可以看作删除根节点后，合并左右子树的操作。

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。