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。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。

定义左偏树中一个节点的距离，其值为该节点递归的向右下一直到叶子节点的边的数量。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N个节点的左偏树的根节点的距离小于等于，；

如下图：



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。

合并下图中的两个左偏树：



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。而且节点的距离也没有更新。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*left\_child*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。