## **可并堆-**左偏堆**：**

顾名思义就是可以合并的堆，堆的实现本身不难，而且有stl可用，但左偏堆必须手写。可并堆是并查集和堆的合体，但是它只有2个叉，它和堆的不同是，左偏堆不是一个堆，而是一群堆(也就是个森林)，初始是n个离散节点，各自是一个堆，操作如下：

合并操作：合并两个堆，核心，下面说

插入操作：插入操作也只是把某个单独节点与要插入的堆合并，本质还是合并，删除：只能删除某个堆顶端，要调整堆，不是真的删除节点，而是标记成被删除，为了节省内存，小顶堆可把val赋值成负无穷。

维护属性 ：左右和父亲地址，权值val，距离dis，它表示当前节点到最近叶子结点的距离，本身是叶子dis=0，注意到普通堆里没有dis这个属性。

以下用小跟堆来说。可并堆有如下性质：

1. 节点的权值小于等于它左右儿子的权值。(堆的基本性质)
2. 节点的左儿子的距离不小于右儿子的距离。(所以叫左偏堆)
3. 节点的距离等于右儿子的距离+1。(基于性质2的推论)
4. n个节点的可并堆距离最大为log(n+1)-1 （思考并计算得到）

### 合并：

对于两个堆A B，设A是根节点val较小的树，把A的根节点作为新树的根节点。

之后顺着A的右子节点，走到最右叶子，把新来的插在这个叶子上。

此时，会打破性质2，也就是距离会出现左小于右，就要递归的调整，遍历最右叶子到根节点的链每个节点，发现某个不满足性质2，就交换它的左右子树即可。

由于是左偏堆，这么做，会保证堆的尽可能平衡。

此外，对于给定2个点x和y,合并x和y所在树，应该先找到他们根节点。判断是不是本来就是一棵树，类似并查集那样。

### 删除：

删除一个堆顶后，调用合并操作，合并它的儿子即可。

删除一个堆顶肯定不是真的删除，而是标记成某个数。

## **可并堆-**二项堆**：**

二项堆和左偏堆功能一样，是另一种可并堆的实现方式。二项堆既然是可并堆，同样，二项堆不是一个树，而是一群二项树(也就是个森林)

定义二项树：

1、度数为0的二项树只包含一个结点。

2、度数为k的二项树有一个根结点，且根结点有k个儿子，儿子度数分别为0,1,2…k-1有序排列,且根结点k个儿子为根的树也是二项树

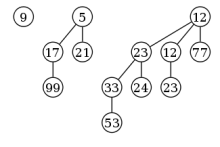
二项树性质：

* 度数为k的二项树共有个结点，高度为k
* 二项树的第d层有C(n,d)个节点，CC代表组合数

二项堆是指满足以下性质的二项树的集合：

* 每棵二项树都满足最小堆性质，即结点关键字大于等于其父结点的值
* 每颗二项树的度数都不同

以上第一个性质保证了二项树的根结点包含了最小的关键字。第二个性质则说明结点数为n的二项堆，最多包含颗二项树

[](https://baike.baidu.com/pic/%E4%BA%8C%E9%A1%B9%E5%A0%86/4771630/0/32fa828ba61ea8d386396afe9b0a304e251f5834?fr=lemma&ct=single)示例：一个含13个结点的二项堆

### 合并