## **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

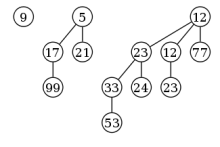
二项树性质：

* 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CC(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(也就是森林)：

* 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

### 维护属性

Node{

Node \*son,\*sibling;//左孩子右兄弟

Int val;//权值

}

Node \*root[] //代表根节点的集合，是个数组链表，按照度数大小排列

Int topRootID //代表堆顶元素中最小的编号，在root[]中

### 二项树的合并

只有度数相同的二项树才能合并，两颗度数为k的二项树，只要找到根权值较小的树r1作为父亲，另一颗树r2作为儿子插入到r1的儿子链表最末，这样新树度数是k+1, 算法时间复杂度O(1)

### 二项堆的合并

只要把一个二项堆的包含的每棵树分别插入另一个即可，现在问题变为怎么合并二项树a和二项堆A，其中二项树a度数为k

方法是：在二项堆A中寻找是否有度数为k的树，如果没有，直接插入结束算法；如果有，则合并二项树a和那颗树，然后变为度数k+1的树，继续检查是否有度数为k+1的树…

最坏情况下，比如度数有1,2,3…k的树，插入度数为1的树，要合并k次，看上去好像是线性时间，其实不然，因为斐波那契树的性质限制了总结点数为n的情况下，树最多颗，

所以对于二项树a和二项堆A，时间复杂度最坏是O(logn),均摊复杂度O(1)

### 插入

把新节点看作度数为1的二项树a，就是相当于把二项树a和二项堆A合并

时间复杂度最坏是O(logn),均摊复杂度O(1)

### 查找最小的点

由于之前维护了最小值的在root[]的下标，只有查询一下即可，时间复杂度O(1),也不可以不维护而是直接遍历root[]数组，时间复杂度O(logn)

### 删除

删除最小的点：也就是它所在的二项树的根节点，只要把根和其他点分离(可以删除也可以不删除)，然后把它的所有儿子，和其他二项树合并

假设这个二项堆有k颗二项树，则最坏情况是删除第k颗二项树的根，它有k-1个儿子，但由于合并度数为1的树后，其余k-2颗树只需要O(1)时间合并，所以总时间复杂度还是O(logn)

删除任意节点：把这个点的值改变为无穷小，然后调制所在堆使这个点到根，然后用删除最小的点的方法操作。