BST树，是二叉搜索树：

二叉查找树（Binary Search Tree），（又：[二叉搜索树](https://baike.baidu.com/item/%E4%BA%8C%E5%8F%89%E6%90%9C%E7%B4%A2%E6%A0%91" \t "_blank)，二叉排序树）它或者是一棵空树，或者是具有下列性质的[二叉树](https://baike.baidu.com/item/%E4%BA%8C%E5%8F%89%E6%A0%91" \t "_blank)： 若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值； 若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值； 它的左、右子树也分别为[二叉排序树](https://baike.baidu.com/item/%E4%BA%8C%E5%8F%89%E6%8E%92%E5%BA%8F%E6%A0%91" \t "_blank)。

* 每个点的值d都不同，因此若插入时候某个点的值在树中存在，则插入失败
* 左边的d一定小于当前点，右边点一定大于当前点
* 中序遍历可得到有序序列

# FHQ\_Treap

是平衡的BST树，怎么保持平衡，通过额外维护一个数rnd， 使树保持BST树性质同时，rnd满足堆的性质，即每个节点的rnd都要大于等于它父节点的rnd,rnd是一个随机数，随机值取保证了，树的节点树越多，退化成一个链的几率越小.

前面的介绍表示，他就是个treap，而fhq\_treap比普通treap好在不是通过旋转保持平衡。它有2个特殊操作，定义：

void split(Node \*r,int k,Node \*&tl,Node \*&tr)

把以r为根的树的按key的把比k小的节点组成的树分离，tl,tr是返回结果，分别代表较小和较大的两棵树，注意，拆分操作可以有不同规则，此规则最常用。

Node\* merge(Node \*a,Node \*b)

代表把以a和b为根节点的两棵树合并，返回合并后的根

下面介绍包括split和merge在内的fhq\_treap所有操作

## 分裂(split)

## 合并(merge)