分治法的思想：将大问题分成若干（一般是两个）个同类型的小问题来解决。

分治法和二分法的区别：分治法的小问题都要解决，而二分法只需要处理一半，丢弃另外一半。

什么样的数据结构适合分治法：

1. 数组：一个大数组可以分割成若干不相交的子数组，归并排序，快速排序都是基于数组的分治法
2. 二叉树：二叉树的左子树和右子树都是二叉树，天然可以用分治法。

碰到二叉树的问题，就想一想整颗树在该问题上的结果和左右子树在该问题上的结果有什么联系。

二叉树的高度：

最坏情况：，例如二叉树就是一条链，没有分叉

最好情况：，平衡二叉树。

BST（二叉搜索树）的高度和普通二叉树也是相同的情况。

二叉树的考点：





二叉树的第一考察形态：

二叉树上求值，求路径：包括Maximum/Minimum/Average/Sum/Paths

递归的三要素：

1. 递归的定义
2. 递归的拆解
3. 递归的出口

lintcode 596中最小子树的时间复杂度：会遍历到所有的结点，每个结点只访问一次，时间复杂度为：\*(每个结点的访问时间)，通常情况下每个结点的访问时间为，所以时间复杂度为。

二叉树的第二考察形态：

二叉树的结构变化，lintcode 453

二叉树的第三考察形态：

二叉搜索树，BST：左子树的值比root值小，右子树的值不小于root值。定义BST时，如果碰到值等于root值时，一般全部放到右边（也可以都放到左边，但不能放两边）。

二叉查找树的进行查找时，时间复杂度最坏是，此时二叉查找树是一条链。通常情况下记成，其中h为树的高度。

二叉查找树的插入时间复杂度也是。

要实现二叉查找树的增删改查，用python实现。

红黑树：平衡二叉查找树。

在时间内实现了增删查改。

在时间内实现找最大最小。

在时间内实现找比某个数小的最大值和比某个数大的最小值。

lintcode 902 ：查找BST上第k小元素，时间复杂度分析：

时间复杂度为。其中h为树的高度，这个分析如下：

当k=1时，要找BST的最小值，时间复杂度为。

当k=n时，要找BST的最大值，这时就相当于要进行一次中序遍历，时间复杂度为。

通过上述两个特殊情况，可以猜出时间复杂度为。