**数据结构**

# 二叉树

## 二叉树的遍历方式

### 先序遍历

#### 递归遍历

#### 非递归遍历

# 平衡二叉树

## 为什么引入二叉平衡树

将{61,88,58,47,35,73,51,99,37,93}构建如图1的二叉排序树。但如果数组元素的次序是从小到大有序，如{35,37,47,51,58,62,73,88,93,99}，则二叉排序树就成了极端的右斜树，注意它依然是一颗二叉排序树，如图2,。此时，同样是查找结点99，左图只需要比较三次，而右图就需要11次比较才可以得到结果，二者差异很大。

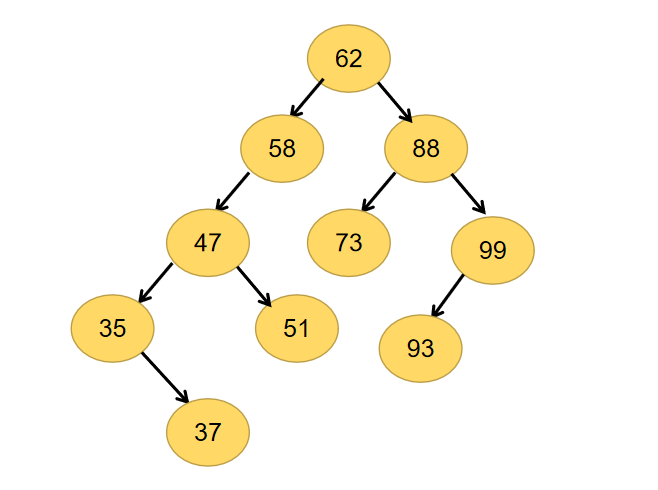


图1

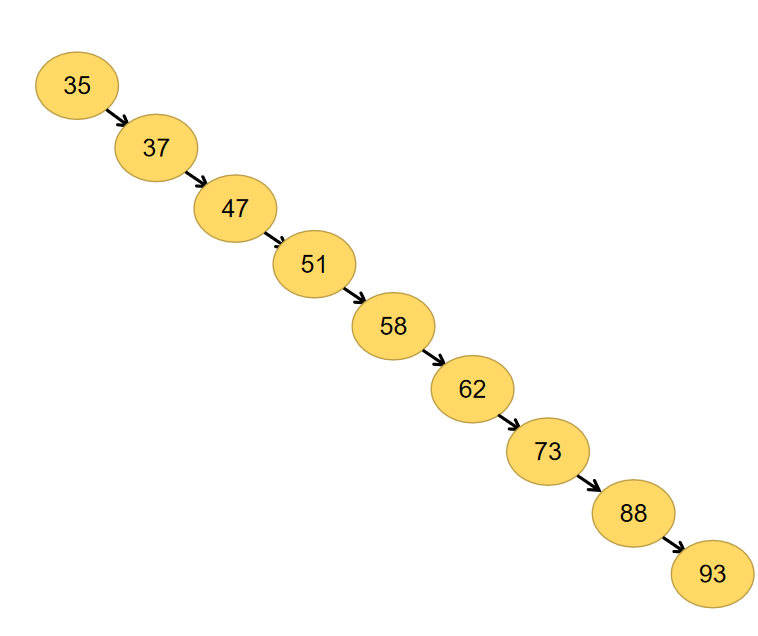


图2

我们希望二叉树是比较平衡的，即其深度与完全二叉树相同，均为，

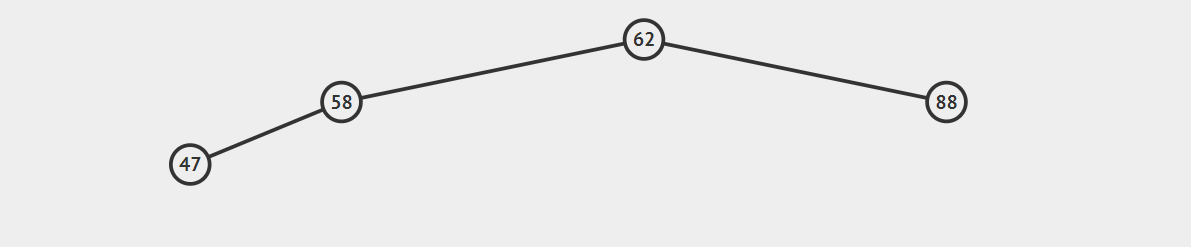
那么查找的时间复杂度也就O()，近似于折半查找。

不平衡的最坏情况就是像图2的斜树，查找的时间复杂度为O(n)，这等同于顺序查找。

因此，如果我们希望对一个集合按二叉排序树查找，最好是把它建成一棵平衡的二叉排序树。这样我们就引申出另一个问题，如何让二叉排序树平衡的问题。

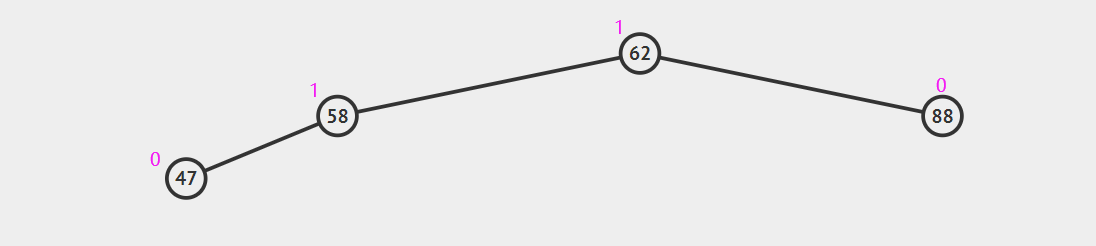
## 平衡二叉树的定义

平衡二叉树是一种二叉排序树，其中每一个节点的左子树和右子树的高度差至多等于1。



## 平衡因子

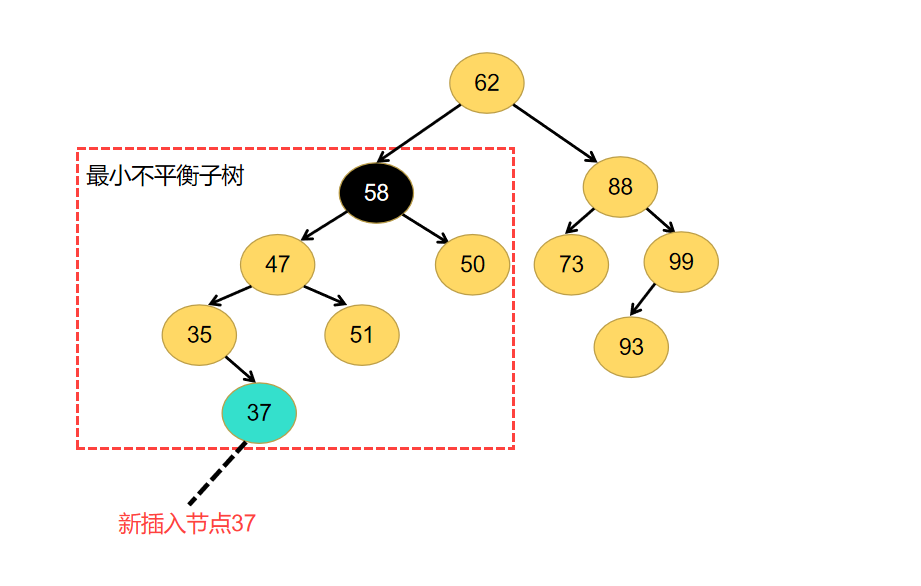
将二叉树上节点的左子树的高度减去右子树深度的值称之为平衡因子BF(Balance Factor),那么平衡二叉树上所有节点的平衡因子只可能是0，-1，1。只要二叉树上有一个节点的平衡因子的绝对值大于1，则该二叉树就是不平衡的。



## 最小不平衡子树

距离插入节点最近的，且平衡因子的绝对值大于1的节点为根的子树，我们称为最小不平衡子树。

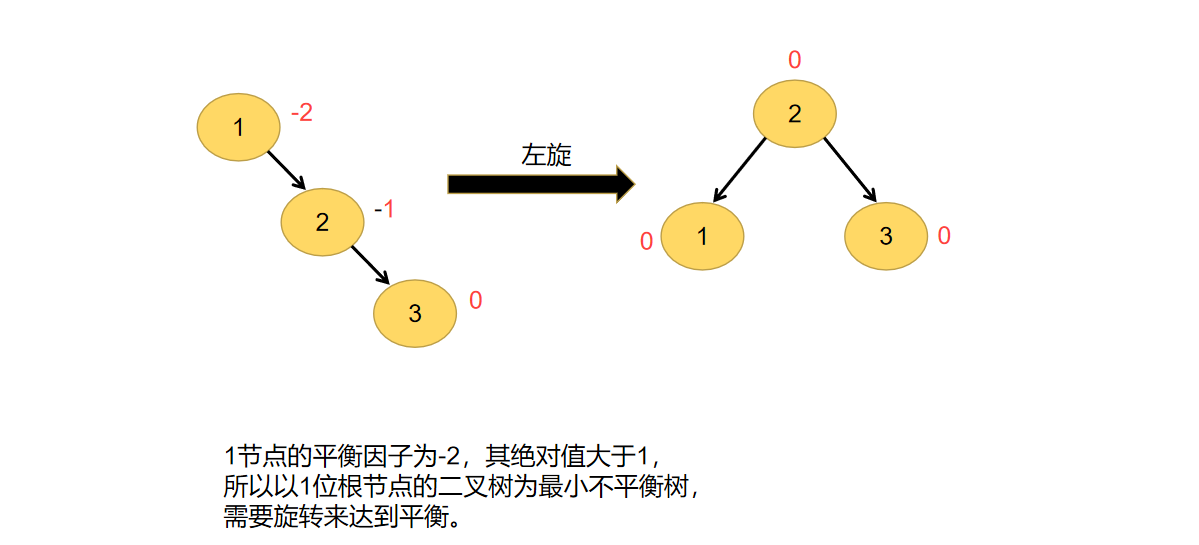
如图，当新插入节点37时，距离它最近的平衡因子绝对值超过1的节点是58(即它的左子树高度3减去右子树高度1)，所以从58开始以下的子树为最小不平衡子树。



## 旋转

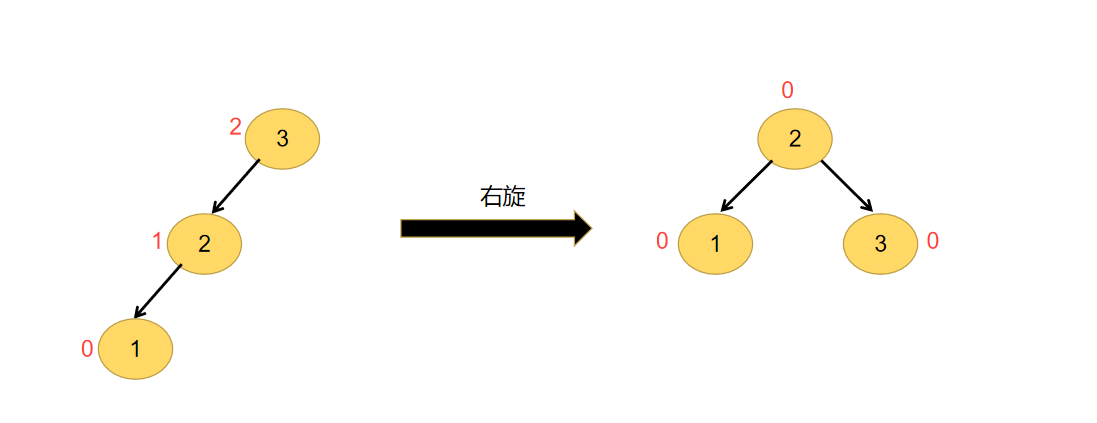
### 左旋

以某个结点作为支点(旋转结点)，其右子结点变为旋转结点的父结点，右子结点的左子结点变为旋转结点的右子结点，左子结点保持不变。



### 右旋

以某个结点作为支点(旋转结点)，其左子结点变为旋转结点的父结点，左子结点的右子结点变为旋转结点的左子结点，右子结点保持不变。

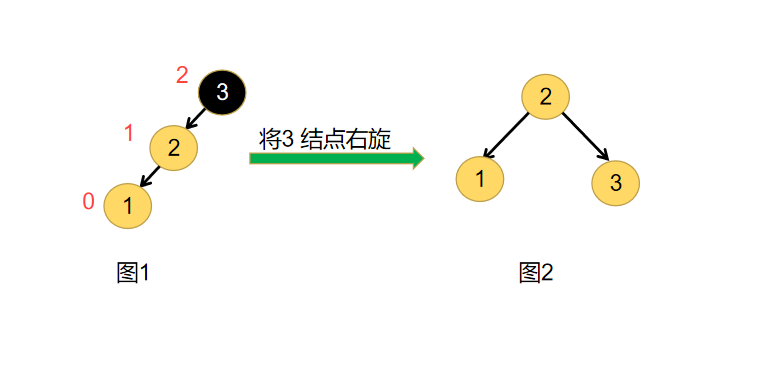


## 平衡二叉树的实现原理

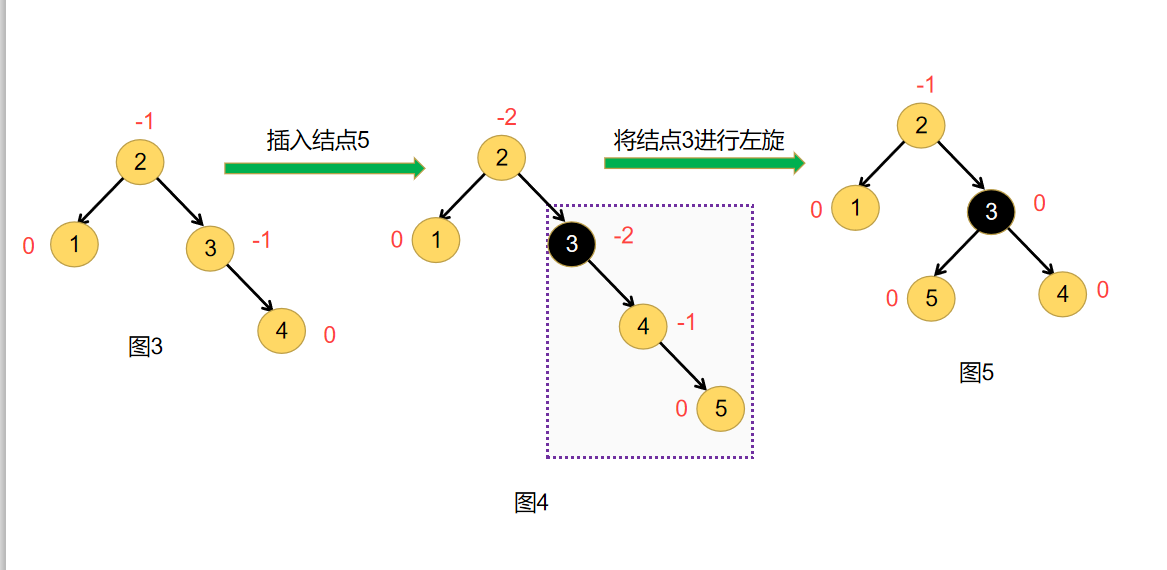
平衡二叉树构建的基本思想就是在构建二叉排序树的过程中，每当插入一个结点时，先检查是否因插入而破坏了树的平衡性，若是，则找出最小不平衡子树。在保持二叉排序树的前提下，调整最小不平衡子树中各结点之间的链接关系，进行相应的旋转，使之成为新的平衡子树。

假设我们现在有一个数组a[10]={3,2,1,4,5,6,7,10,9,8}需要构建二叉排序树。在没有学习二叉排序树之前，根据二叉排序树的特性，我们通常会将它构建成如图1所示的样子。虽然它完全符合二叉排序树的定义，但是对这样高度达到8的二叉树树来说，查找是非常不便利的。我们更期望构建成如图2的样子，高度为4的二叉排序树才可以提供高效得效率。那么我们现在就来研究如果将一个数组构建出图2的树结构。

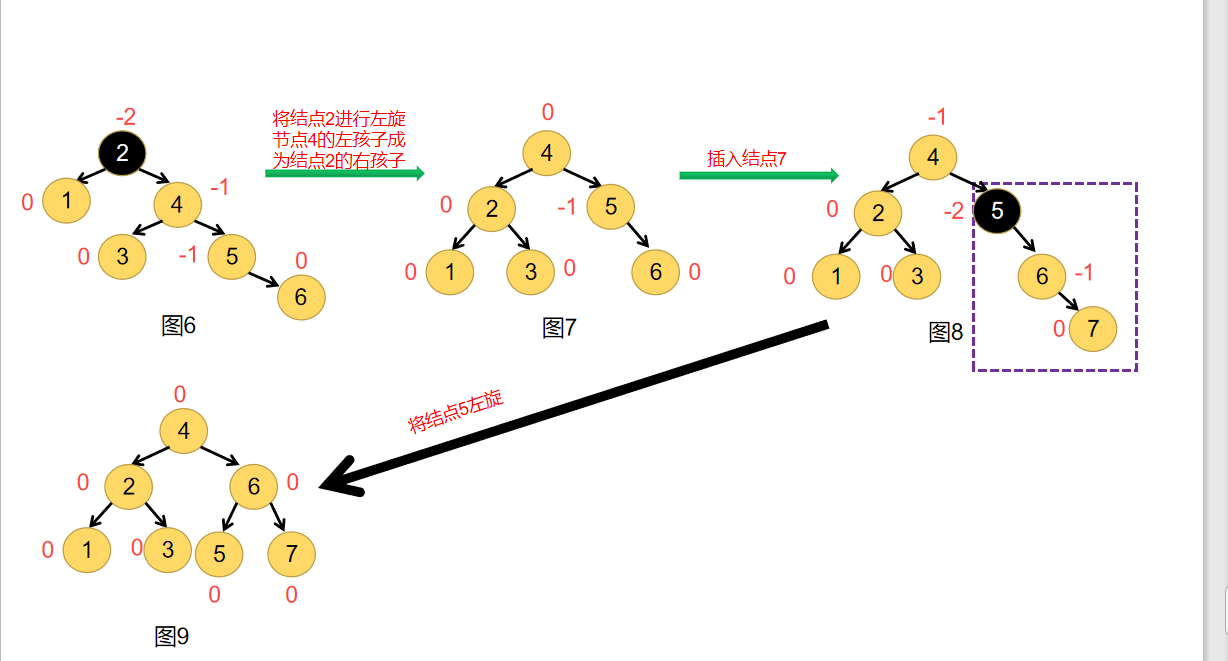
对于数组a[10]={3,2,1,4,5,6,7,10,9,8}的前两位3和2，我们很正常的构建，到了第3个数”1”时，发现此时根节点”3”的平衡因子变成了2，此时整棵树都成了最小不平衡子树，因此需要调整，如图1(结点左上角数字为平衡因子BF值)。因为BF值为正，因此我们将整个树进行右旋(顺时针旋转)，此时结点2成了根节点，3成了2的右孩子，这样三个结点的BF值均为0，非常的平衡，如图2所示。



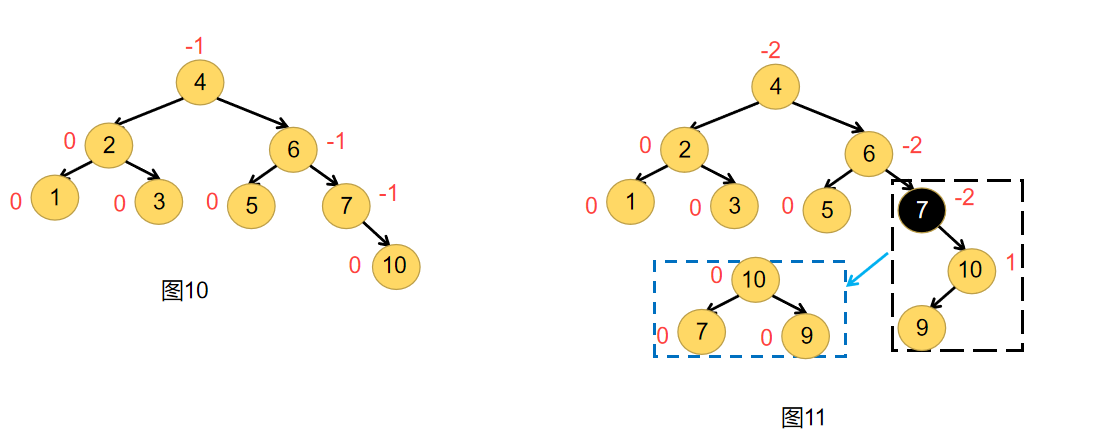
然后我们再增加结点4，平衡因子没有超出限定范围(-1,0,1)，如图3,。增加结点5时，结点3的BF的值为-2，说明要旋转了。由于BF是负值，所以我们对这棵最小平衡子树进行左旋(逆时针旋转)，如图4，此时我们整个树又达到了平衡。



继续，增加结点6，发现根节点2的BF值变为了-2，如果6。所以我们对根节点进行左旋，注意此时本来结点3是4的左孩子，由于旋转后需要满足二叉排序树特性，因此它成了结点2的右孩子，如图7.增加结点7，同样的左旋，使得整棵树达到平衡，如图8和图9所示。

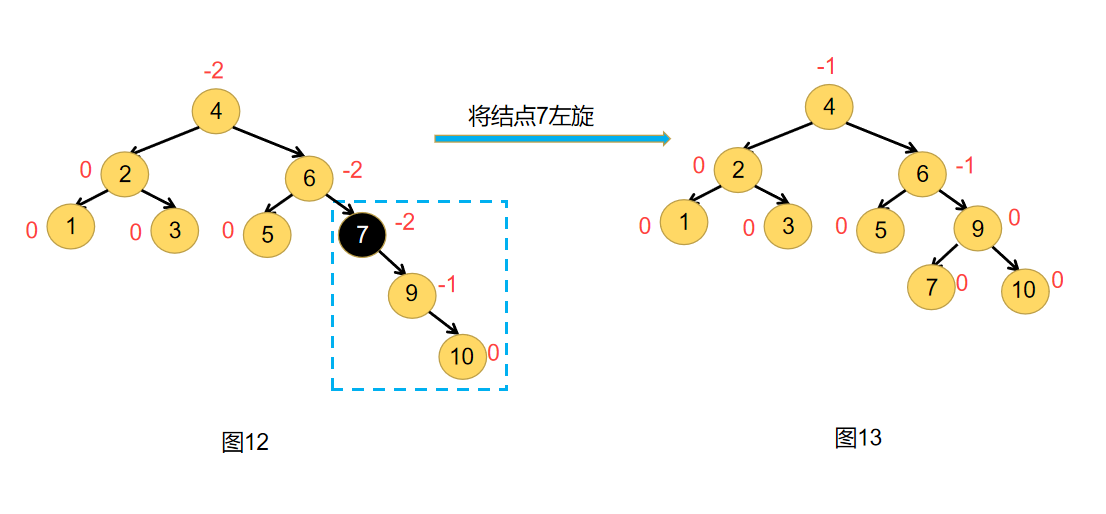


当增加结点10时，结构无变化，如图10。当再增加结点9时，此时结点7的BF变成了-2，理论上我们只需要旋转最小不平衡子树7，9,10即可，但是如果左旋转后，结点9变成了10的右孩子，这是不符合二叉排序树的特性的，此时不能简单的左旋，如图11所示。



仔细观察图11，发现根本原因在于结点7的BF的是-2，而结点10的BF是1，也就是说，它们俩一正一负，符号不统一，而前面的几次旋转，无论是左旋还是右旋最小不平衡子树的根节点与它的子节点的符号都是相同的，这就是不能直接旋转的关键，那怎么办呢？

不统一，不统一就把它们先旋转到符号统一再说，于是先对结点9和节点10进行右旋，使得结点10成了9的右子树，及诶单9的BF变为-1，此时就与结点7的BF值符号统一了，如图12所示。



这样我们再以结点7为最小不平衡子树进行左旋，得到图13。接着插入结点8，情况与刚才类似，结点6的BF是-2，而它1的右孩子9的BF为1，如图14，因此首先以9为根结点，进行右旋，得到图15，此时结点6和结点7的符号都是负，再以6位根节点左旋，最终得到最后的平衡二叉树，如图16所示。

