数据结构之——平衡二叉树（内容详解）

一、基本概念

平衡二叉树也叫AVL树，它或者是一颗空树，或者具有以下性质的二叉排序树：它的左子树和左子树的高度之差(平衡因子)的绝对值不超过1，且它的左子树和右子树都是一颗平衡二叉树。

二、结构

如基本概念所树，它具有一个左子树和一个右子树，且对于任意一个子树而言，左子树和右子树高度只差不超过1。

2.1 平衡二叉树判别

如下有3棵树，分别判断下哪个是平衡二叉树?

图1:

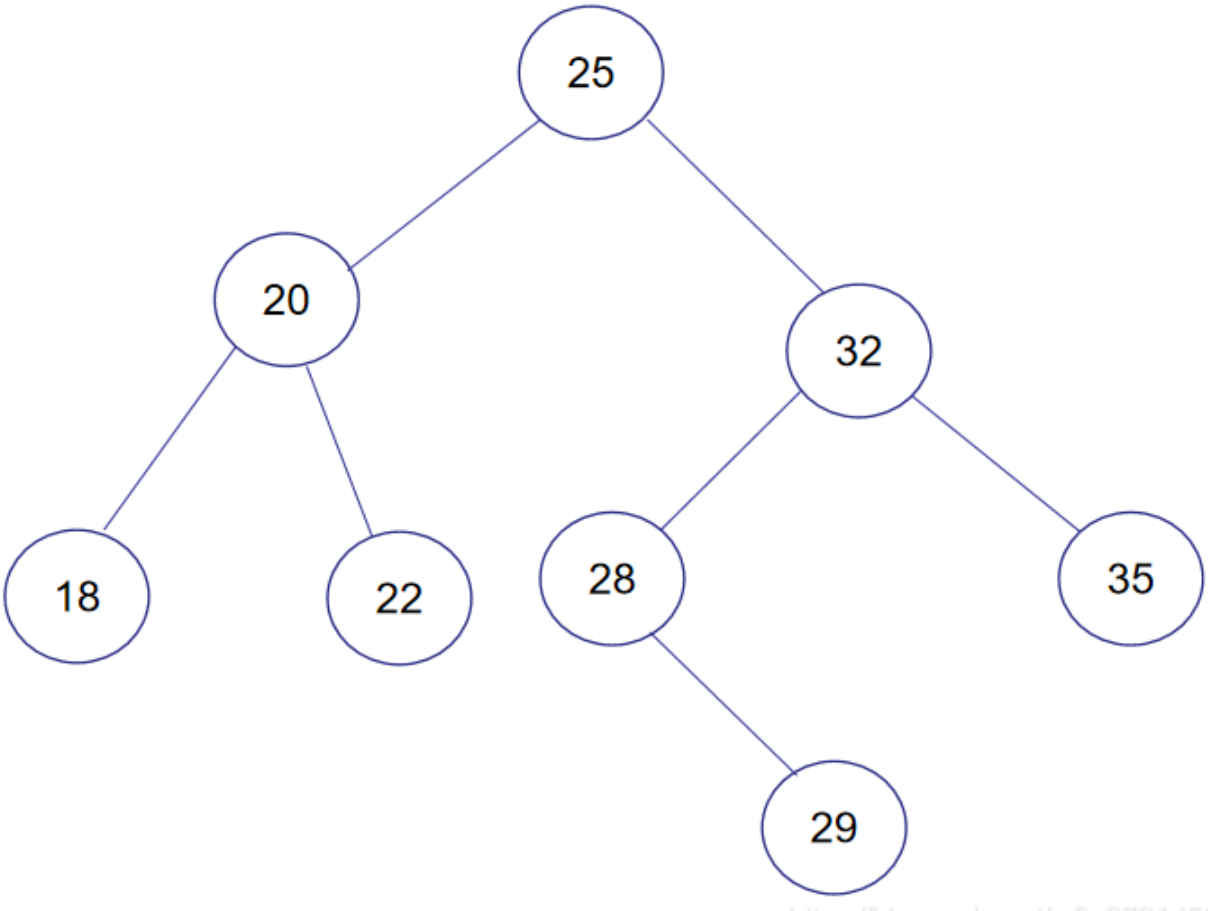


图2:

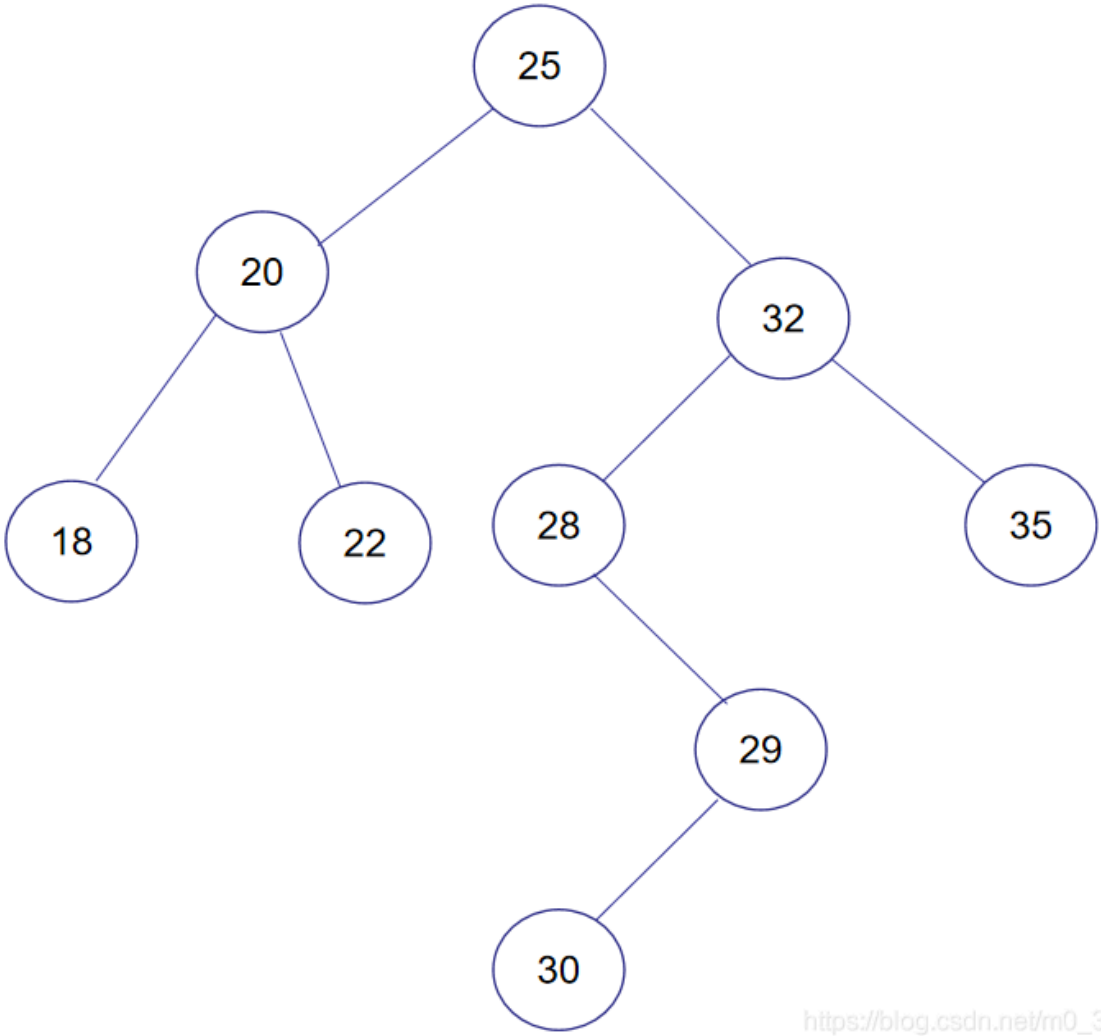
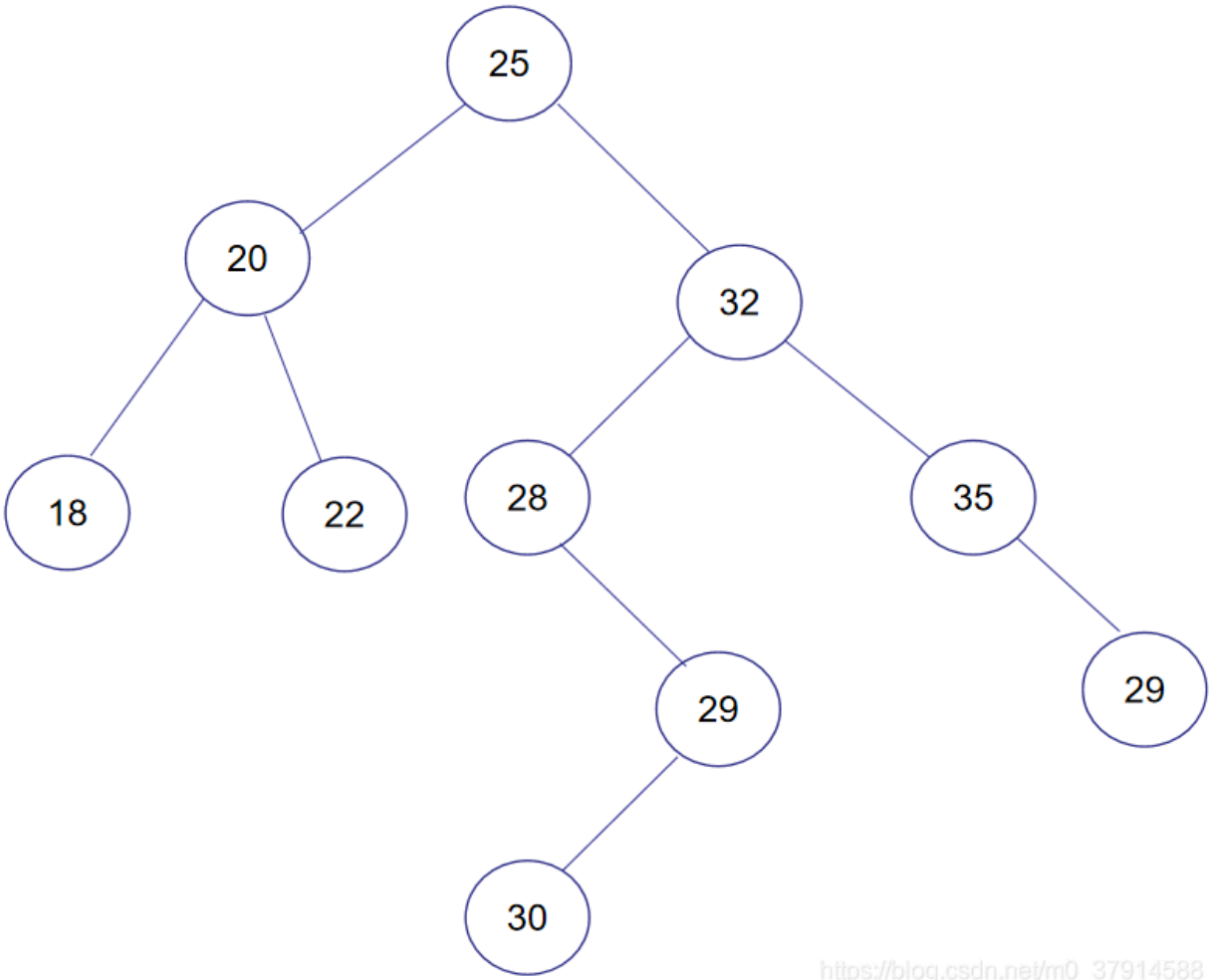


图3：



图一，很明显就能看出图1中任何子树的高度差都在没有超过1，

图二，节点25的左子树高度是3，而右子树高度是5，高度差已经超过1；对于节点32而言，左子树高度是3，右子树高度是1，高度差已经超出1，所以该树不是平衡二叉树。

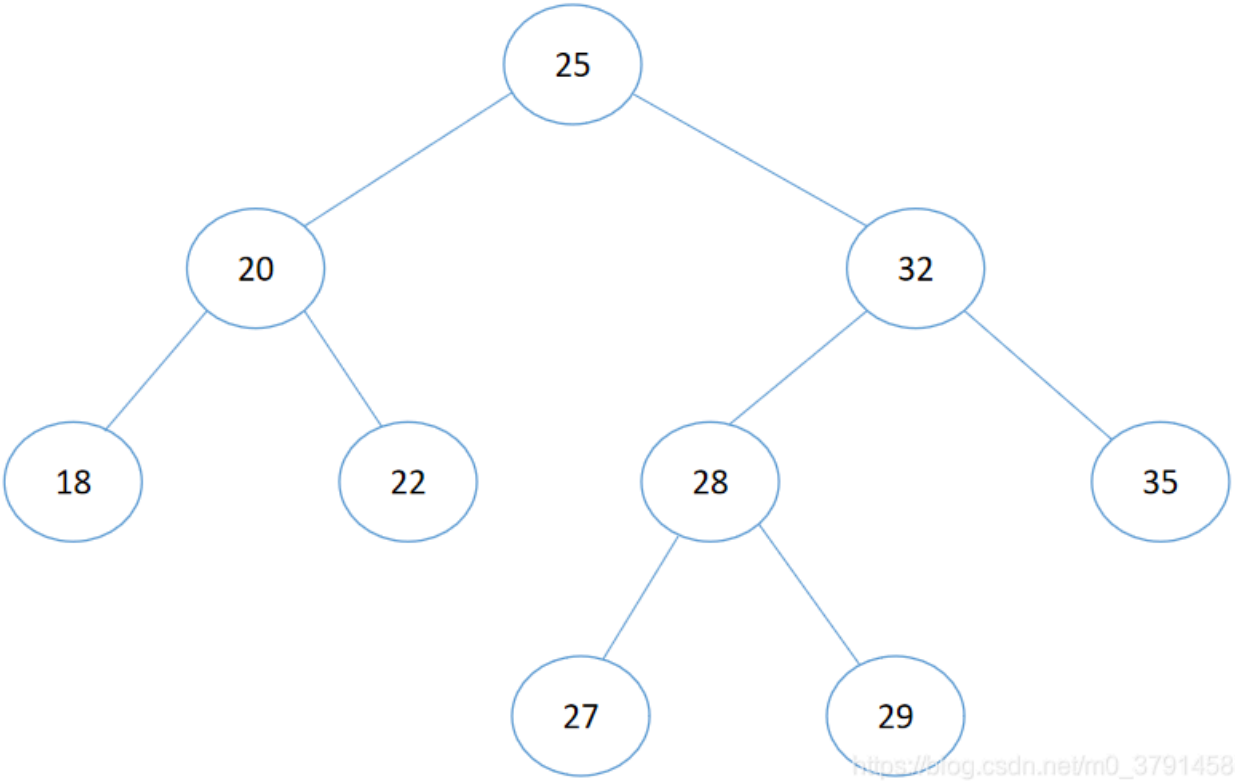
图三，对于节点25的左子树高度是3，而右子树高度是5，高度差已经超过1；对于节点28来说，左子树的高度是0，右指数的高度是2，高度差已经超过1；所以两处不平衡，不是平衡二叉树。

从上图这几个案例，应该能明白什么是平衡二叉树了吧。

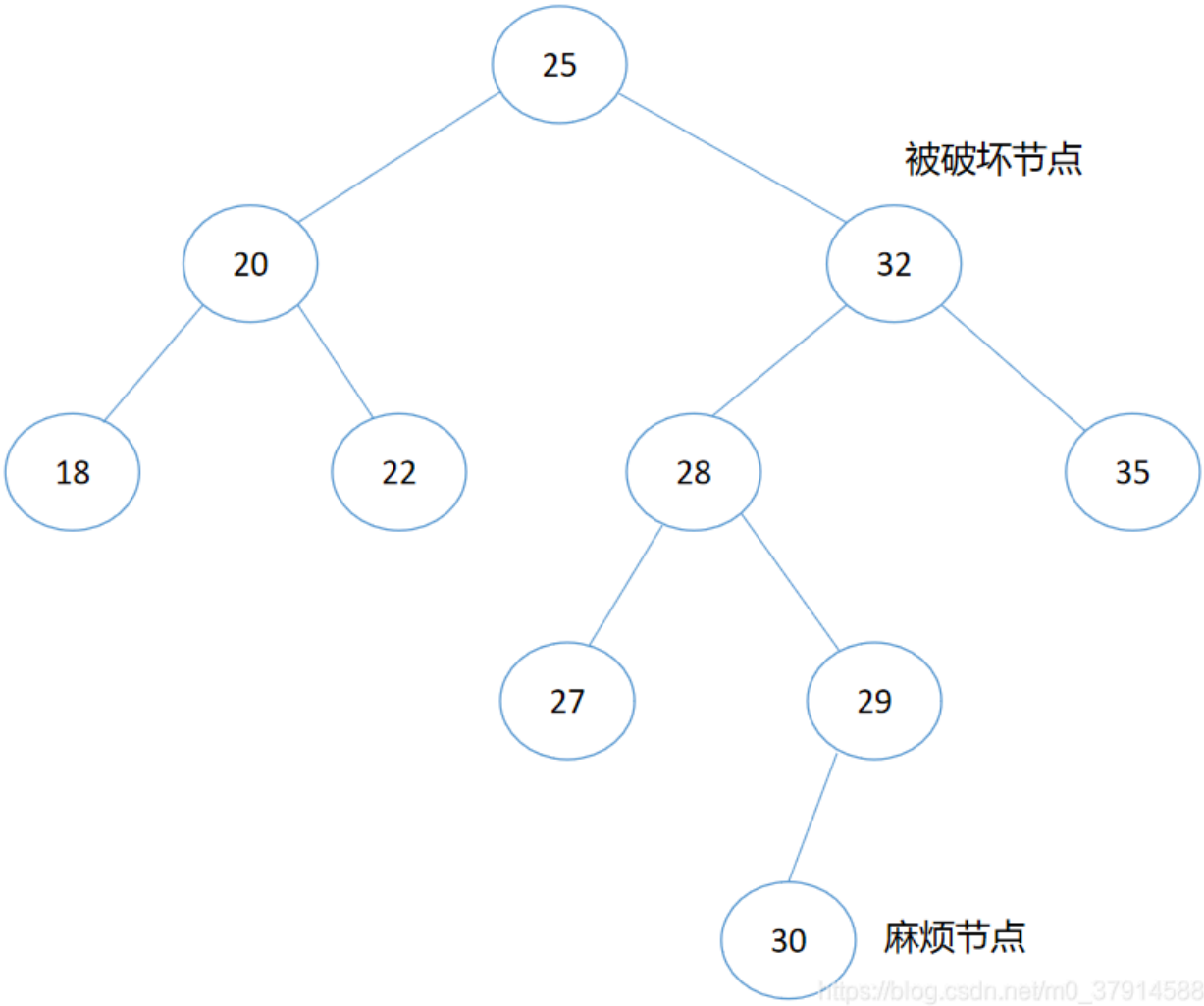
三、插入详解

对于平衡二叉树而言，当插入的新值导致它不是平衡二叉树了，这该怎么办?

插入新值前:



插入新值后：



如上图，本是平衡的，如果我插入了一个新值是30的节点，那么这棵树的平衡性就会被破坏，它一共破坏了节点32和节点25的平衡性，虽然破坏了2个节点的平衡性，但我们只讨论

最近被破坏节点的平衡性（因为底下的处理好了，上面的节点平衡性也会随之处理好），即新插入节点30破坏了节点32的平衡性，因为新插入节点而导致平衡性被破坏的节点也叫麻烦节点，而被其破坏平衡的节点叫被破坏节点。

根据所有结果和经验，一共把破坏平衡型的类型分为了四种，分别是LL型、RR型、LR型和RL型（L=left,R=right）。

3.1 LL型

图1：

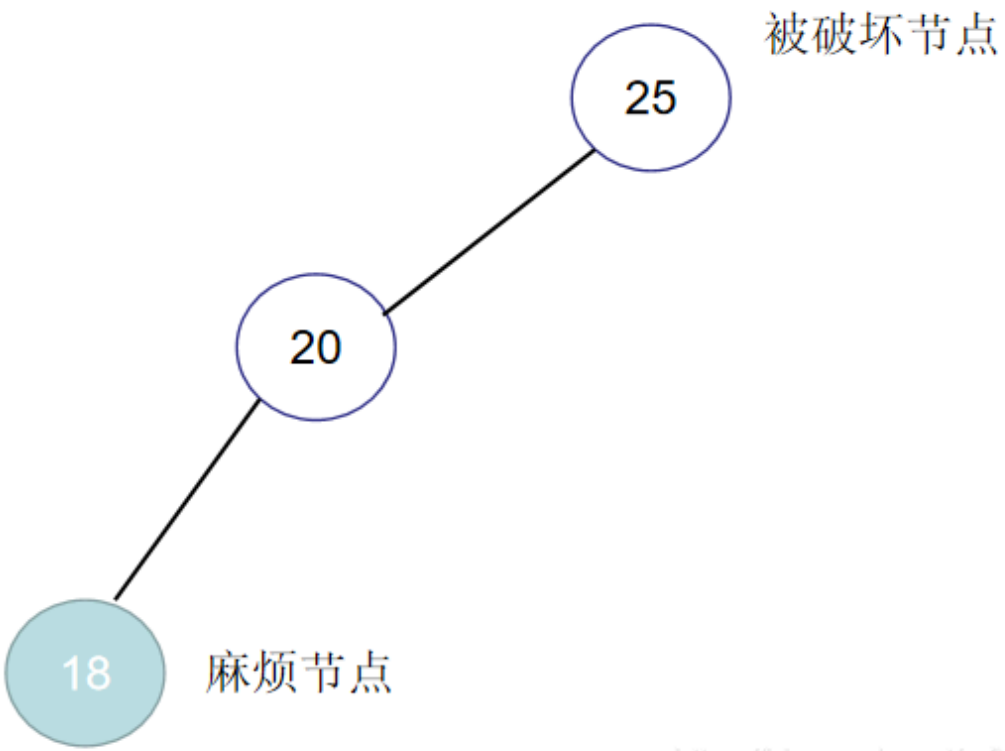


图2：

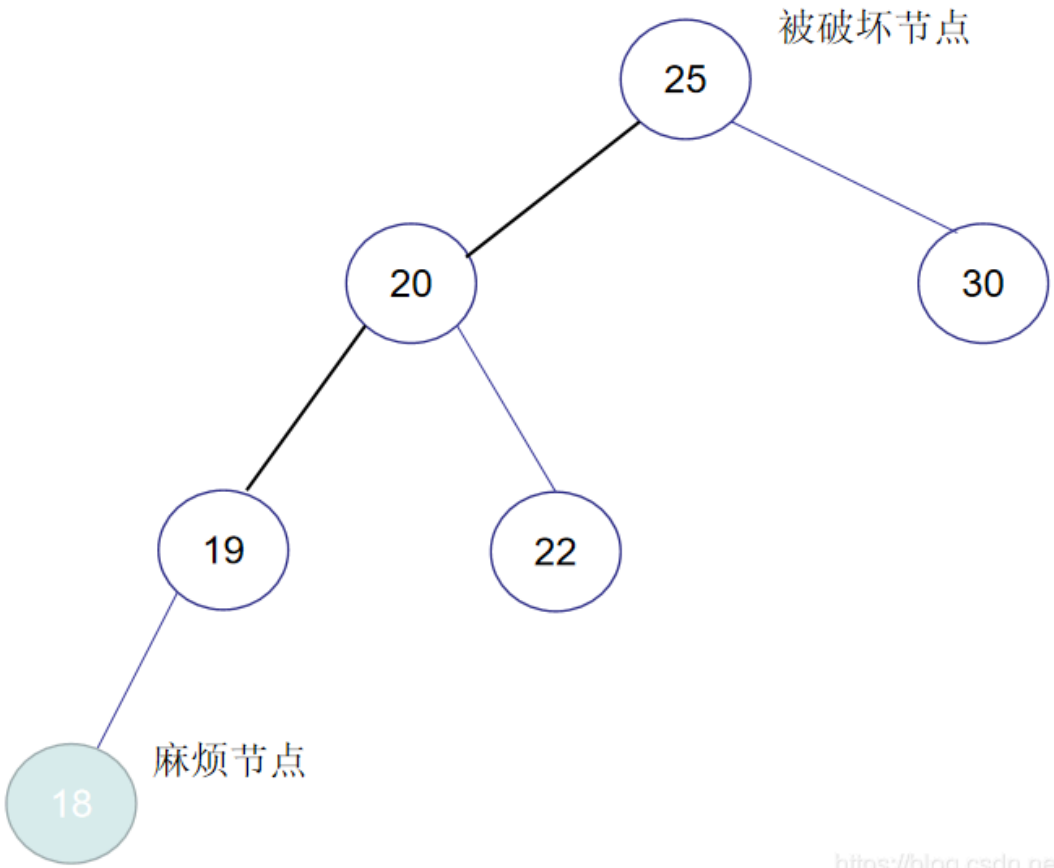
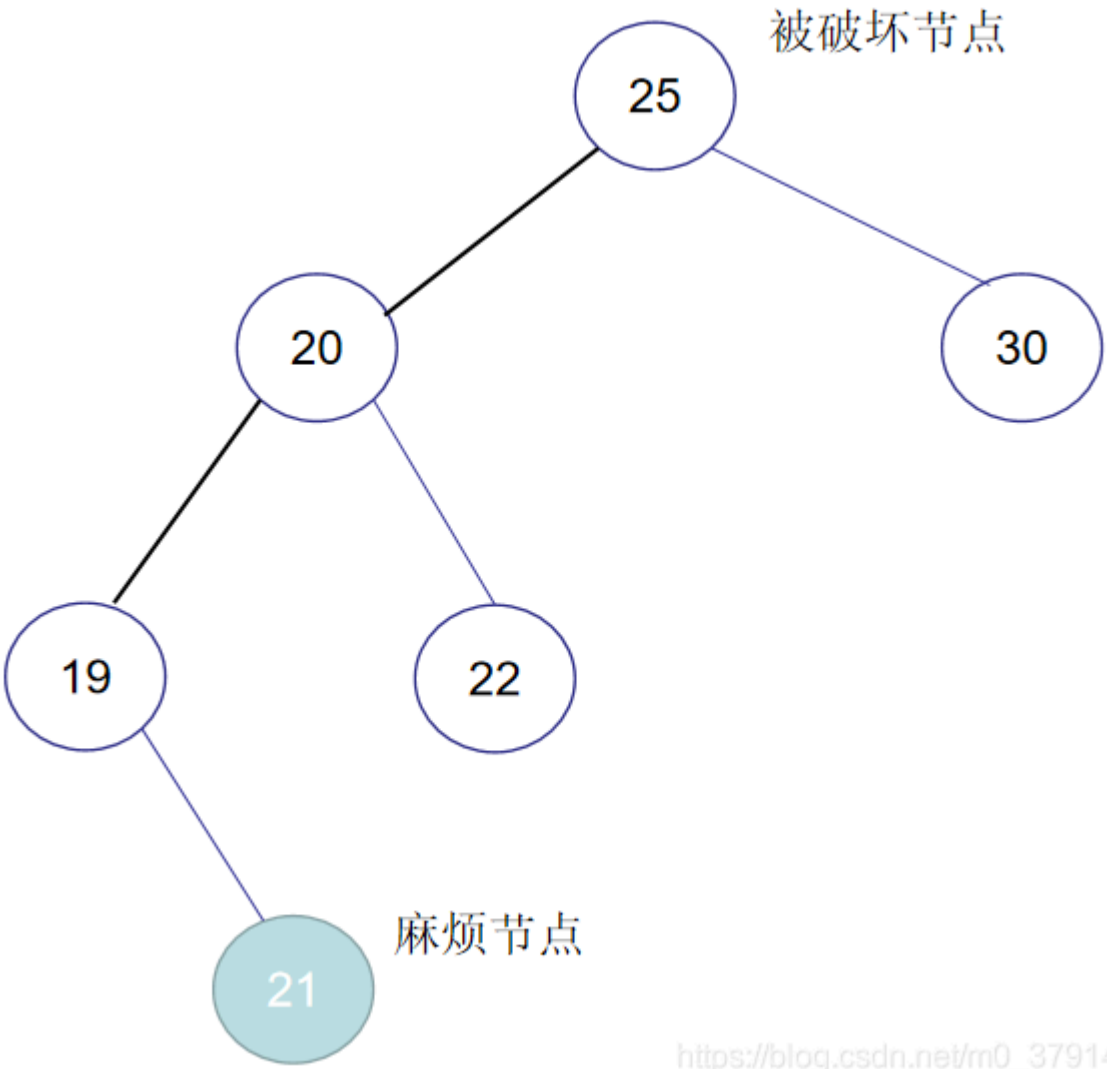


图3：



如上图，LL型也为左左型，在被破坏节点的左边的左边插入而导致失衡，则为LL型，注意上面3个图，黑色的线表示它型号判断的路径，所有型号的判断只从被破坏节点开始判断两次，即不管你第三次新插入节点的位置在左还是在右，如图2和图3，一个左一个右，但是都为LL型，因为只判断两次。

LL型解决方案：以被破坏节点为基础进行右旋

3.2 右旋

什么是右旋，根据某个节点向右旋转，只需要记住下面这个图即可，至于右旋为什么能解决LL型问题，其实这是一个经验总结（就像99乘法表），只需要记住即可，后面会总结一个记录方法。

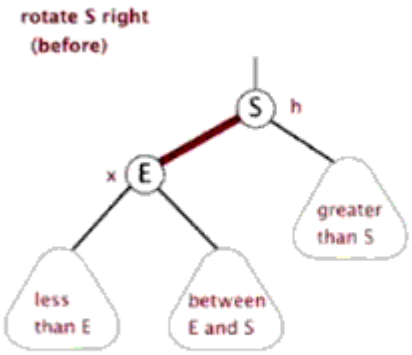
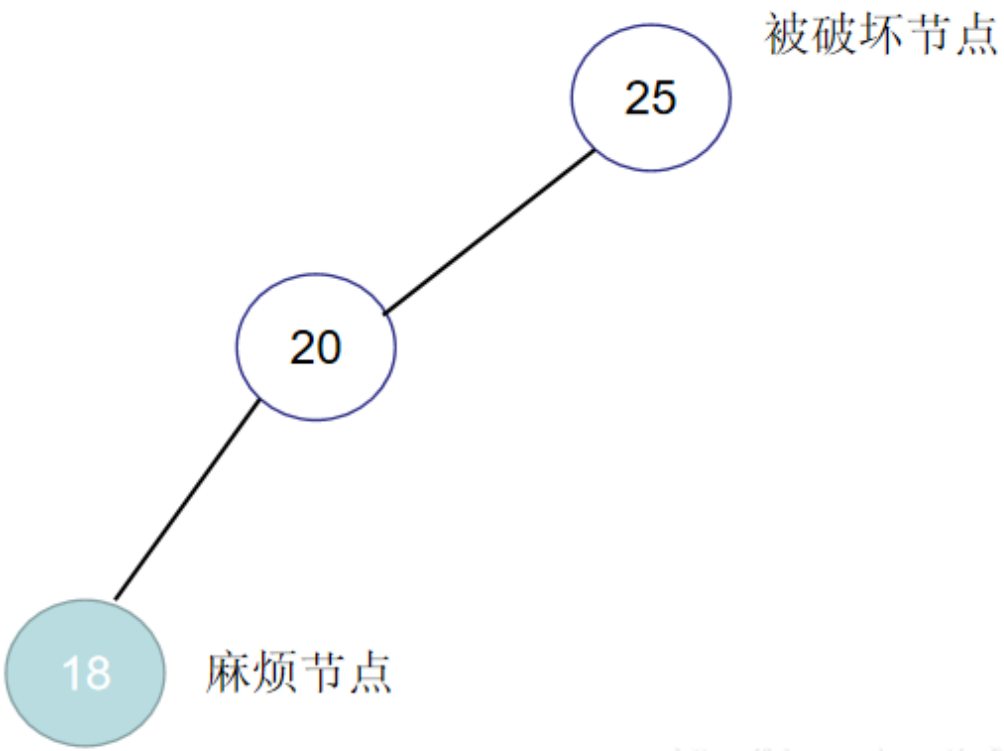


图1右旋：



以25为基础进行进行右旋后结果如下，可以看出，该树处于平衡状态。

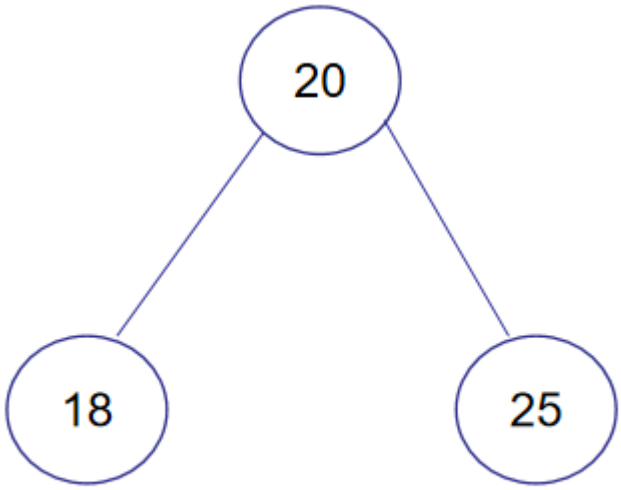
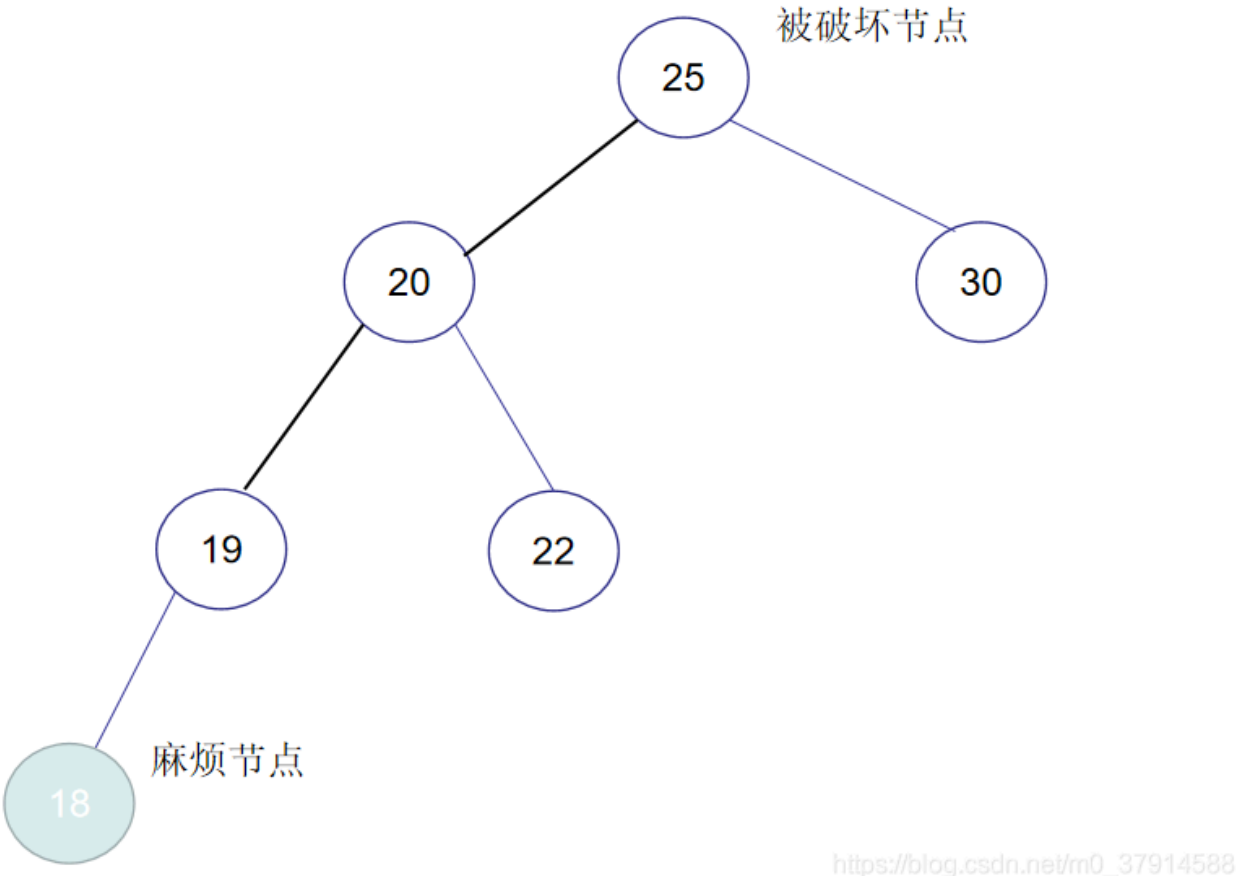


图2右旋：



以25为基础进行进行右旋后结果如下，可以看出，该树处于平衡状态。

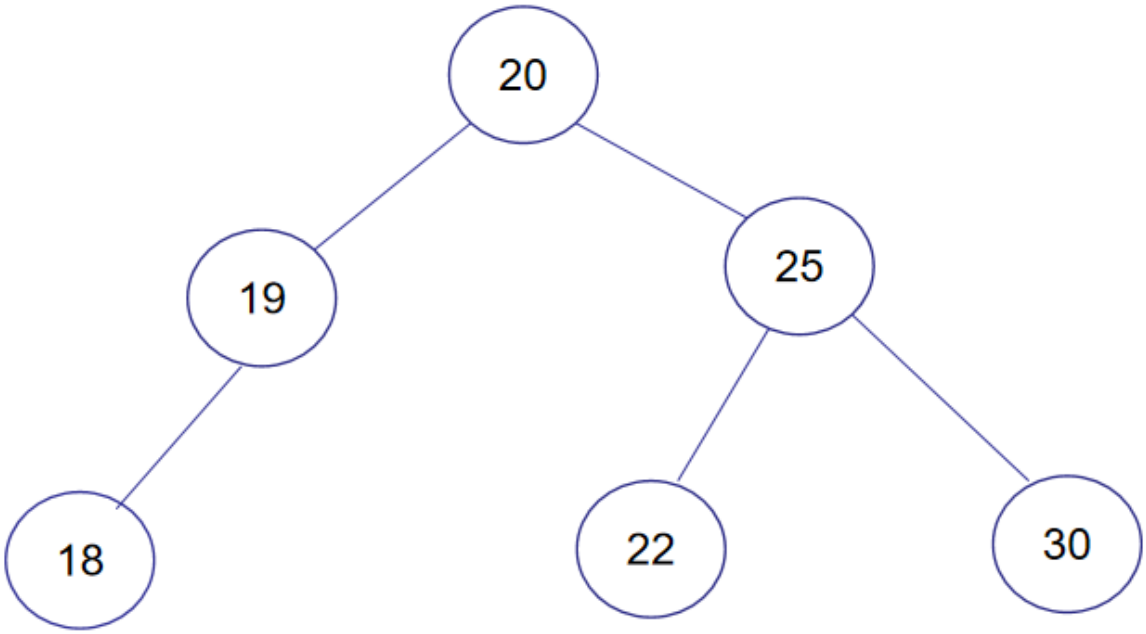
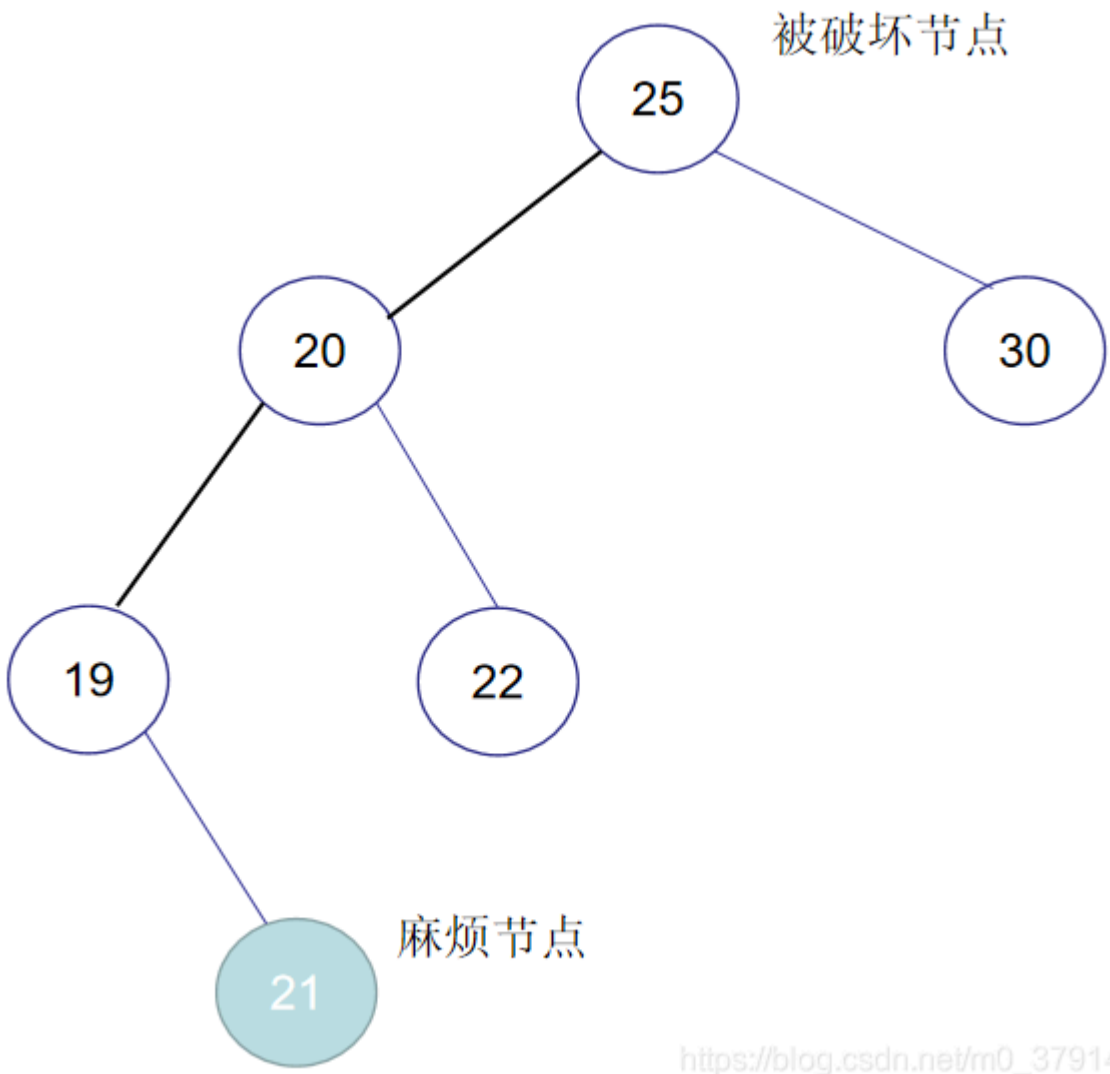
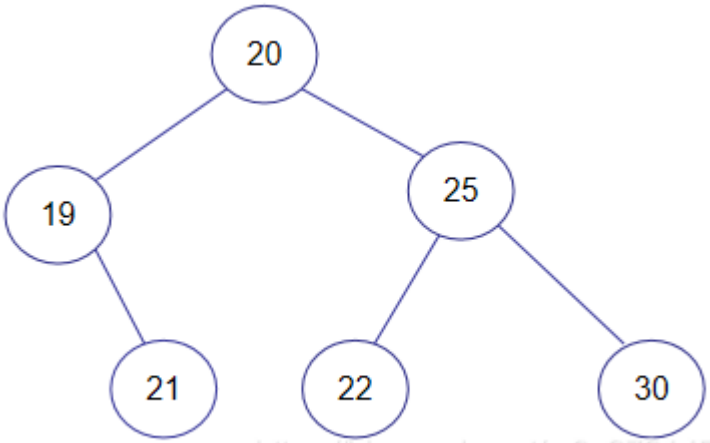


图3右旋：



以25为基础进行进行右旋后结果如下，可以看出，该树处于平衡状态。



3.3 RR型

图1：

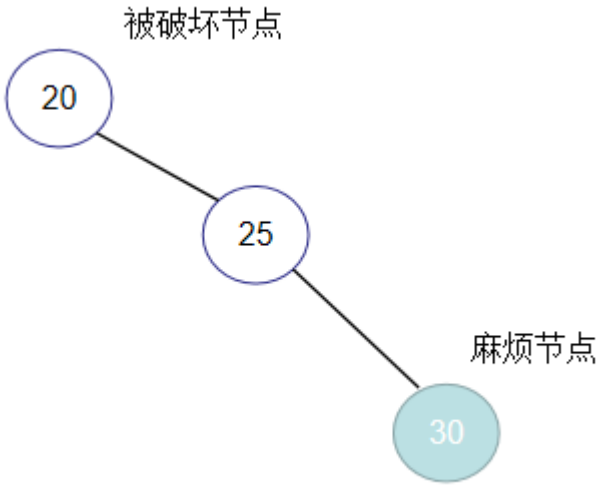


图2：

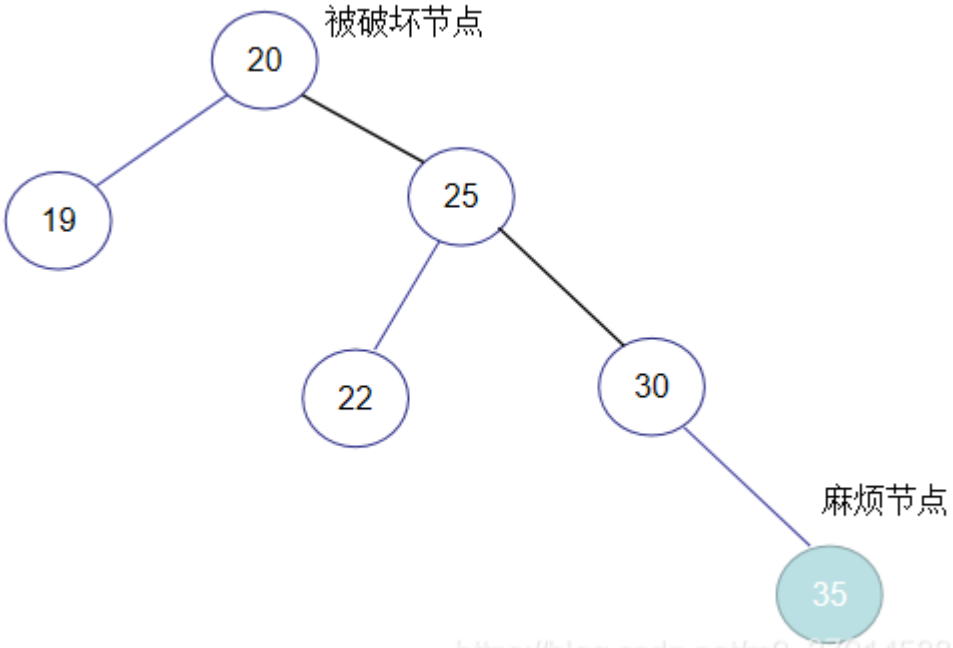
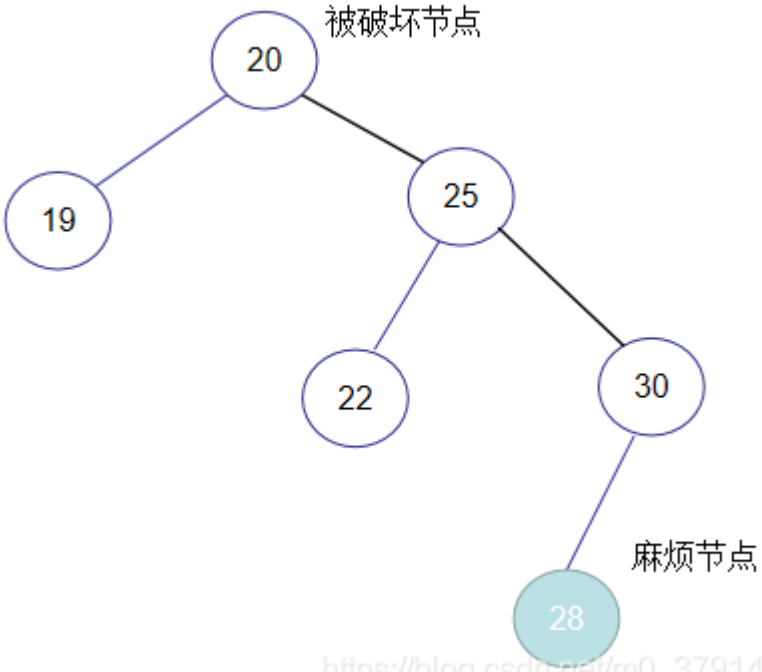


图3：



如上图，RR型也为右右型，在被破坏节点的右边的右边插入而导致失衡，则为RR型，注意上面3个图，黑色的线表示它型号判断的路径，与LL型相反。

LL型解决方案：以被破坏节点为基础进行左旋

3.4 左旋

什么是左旋，根据某个节点向左旋转，只需要记住下面这个图即可，与右旋相反。

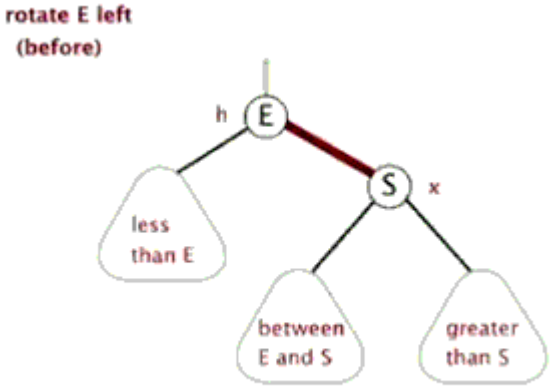
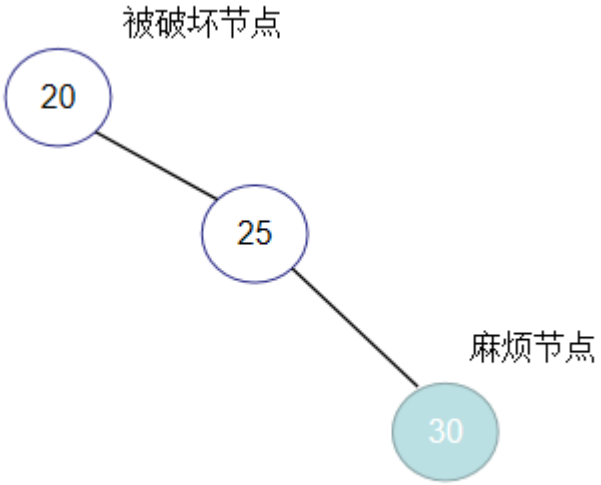


图1左旋：



以20为基础进行进行左旋后结果如下，可以看出，该树处于平衡状态。

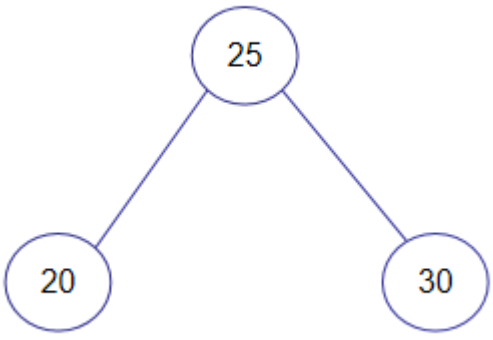
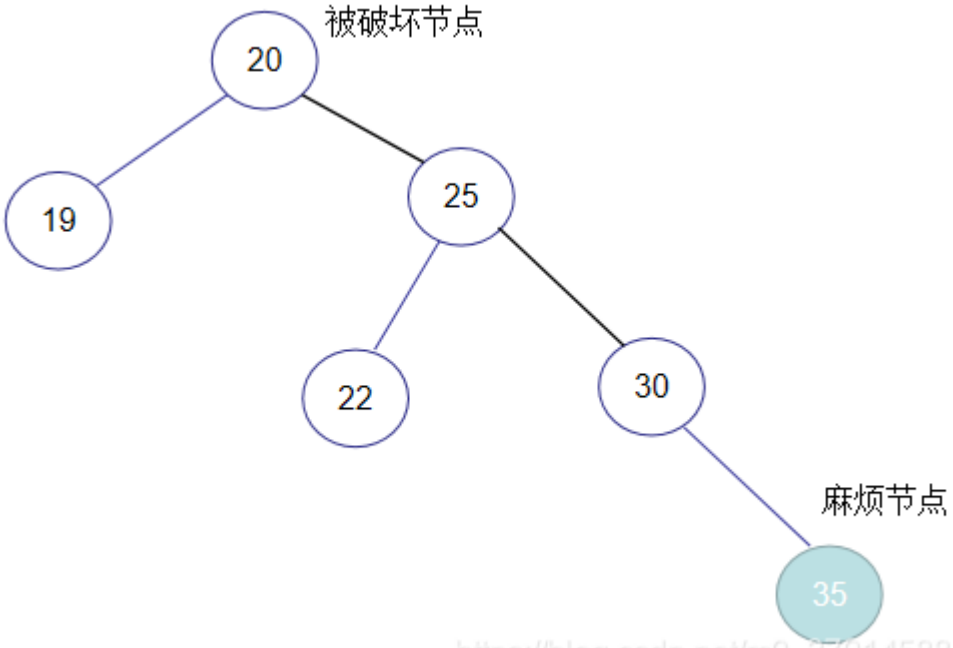


图2左旋：



以20为基础进行进行左旋后结果如下，可以看出，该树处于平衡状态。

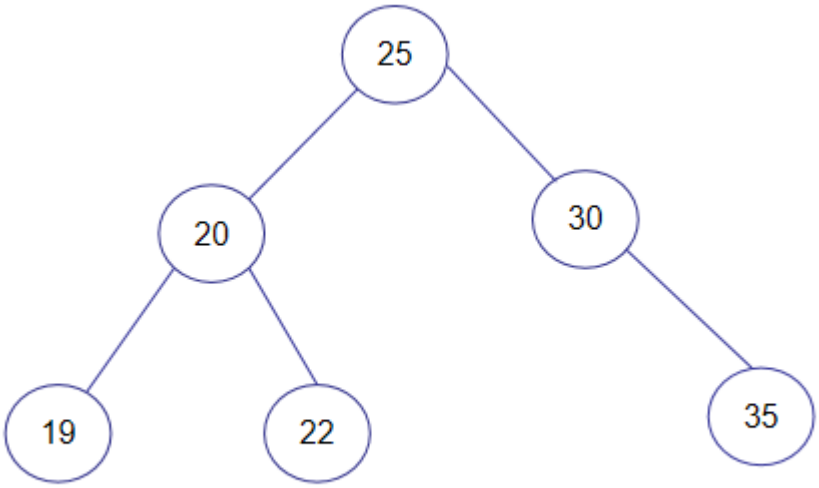
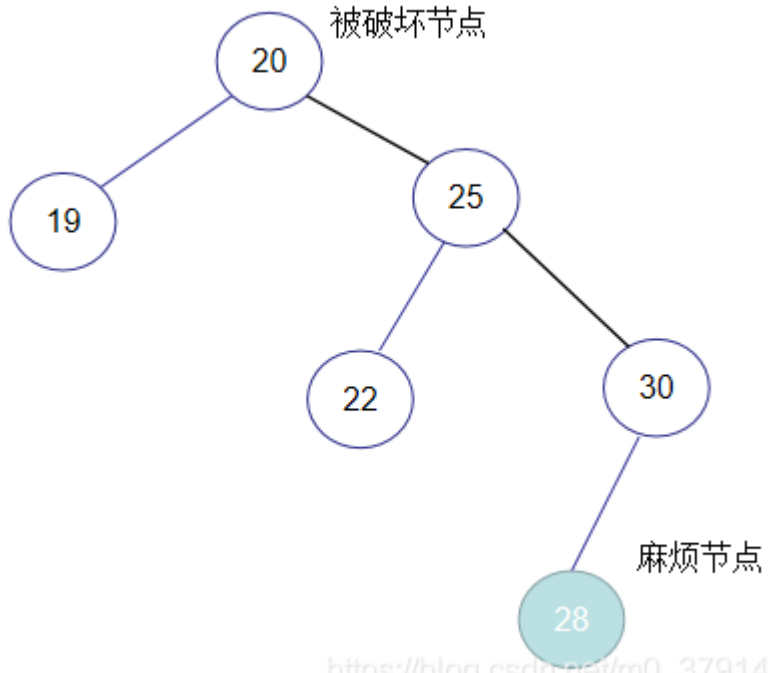
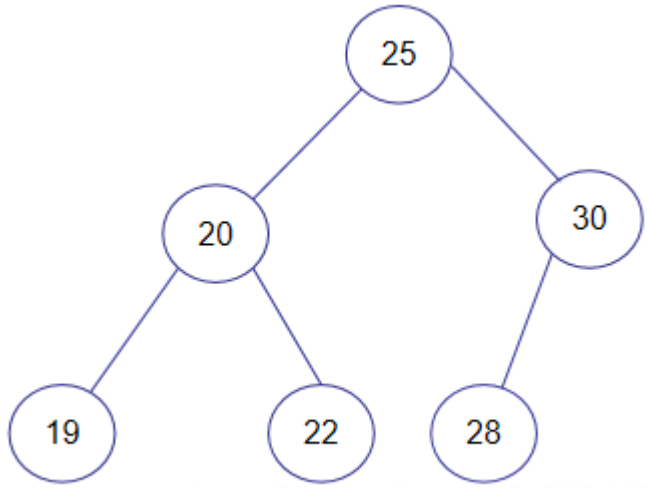


图3左旋：



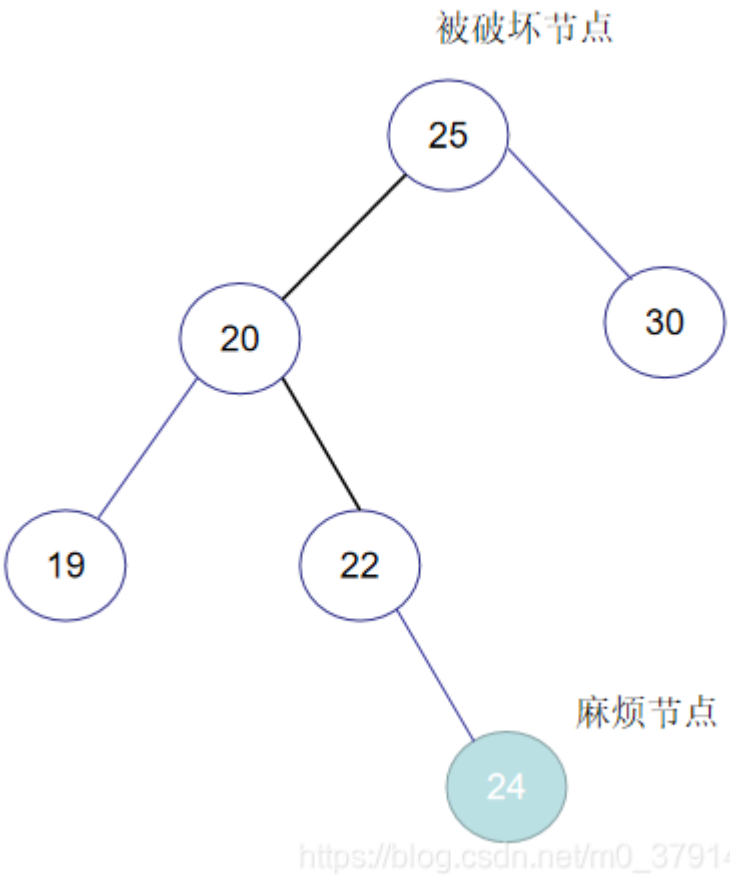
以20为基础进行进行左旋后结果如下，可以看出，该树处于平衡状态。



注意：因为左旋、右旋是基础，所以写了3个例子，为了避免文章过于啰嗦，下面RL型和LR型只举一个例子。

3.5 LR型

如图：

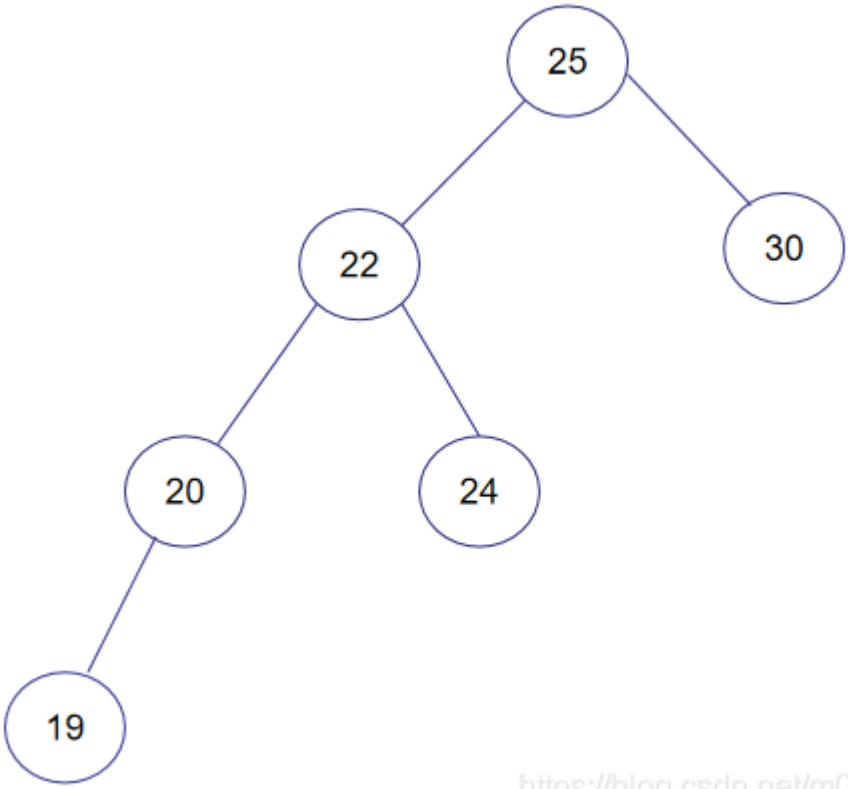


如上图，LR型也为左右型，在被破坏节点的左边的右边插入而导致失衡，则为LR型，注意上图，黑色的线表示它型号判断的路径。

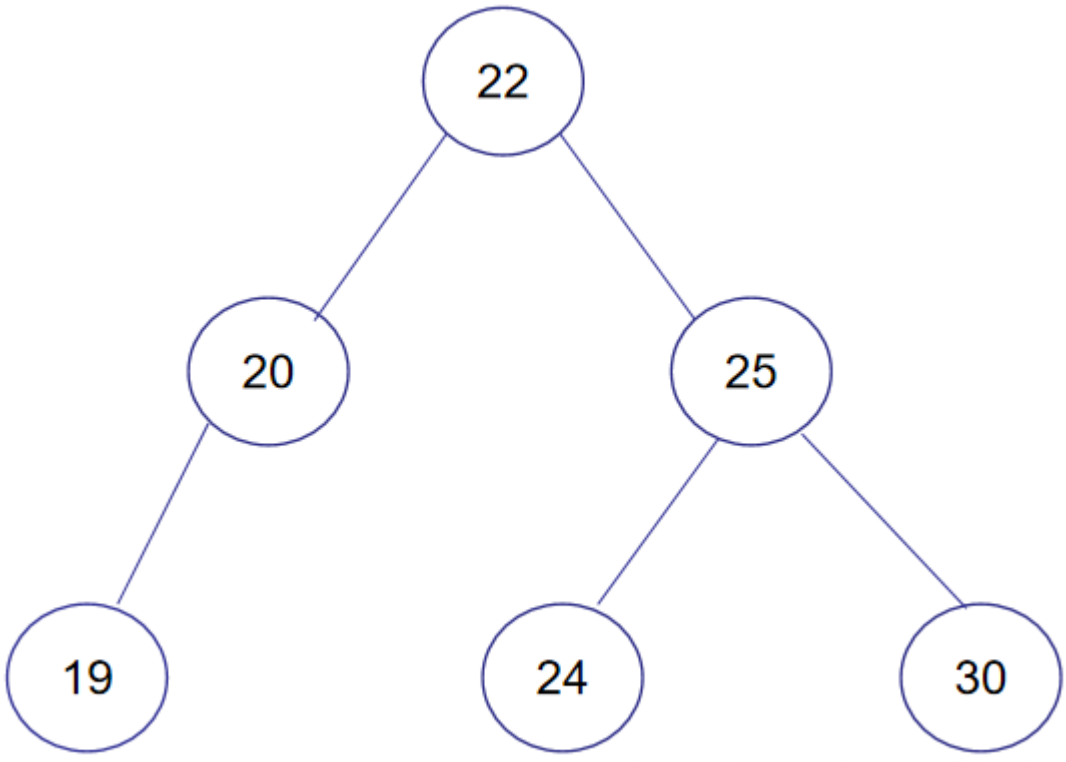
LR型解决方案：以被破坏节点L（左）节点为基础先进行一次L（左）旋，再以被破坏节点为基础进行右旋。

上图调整：

先以20为基础进行一次左旋。

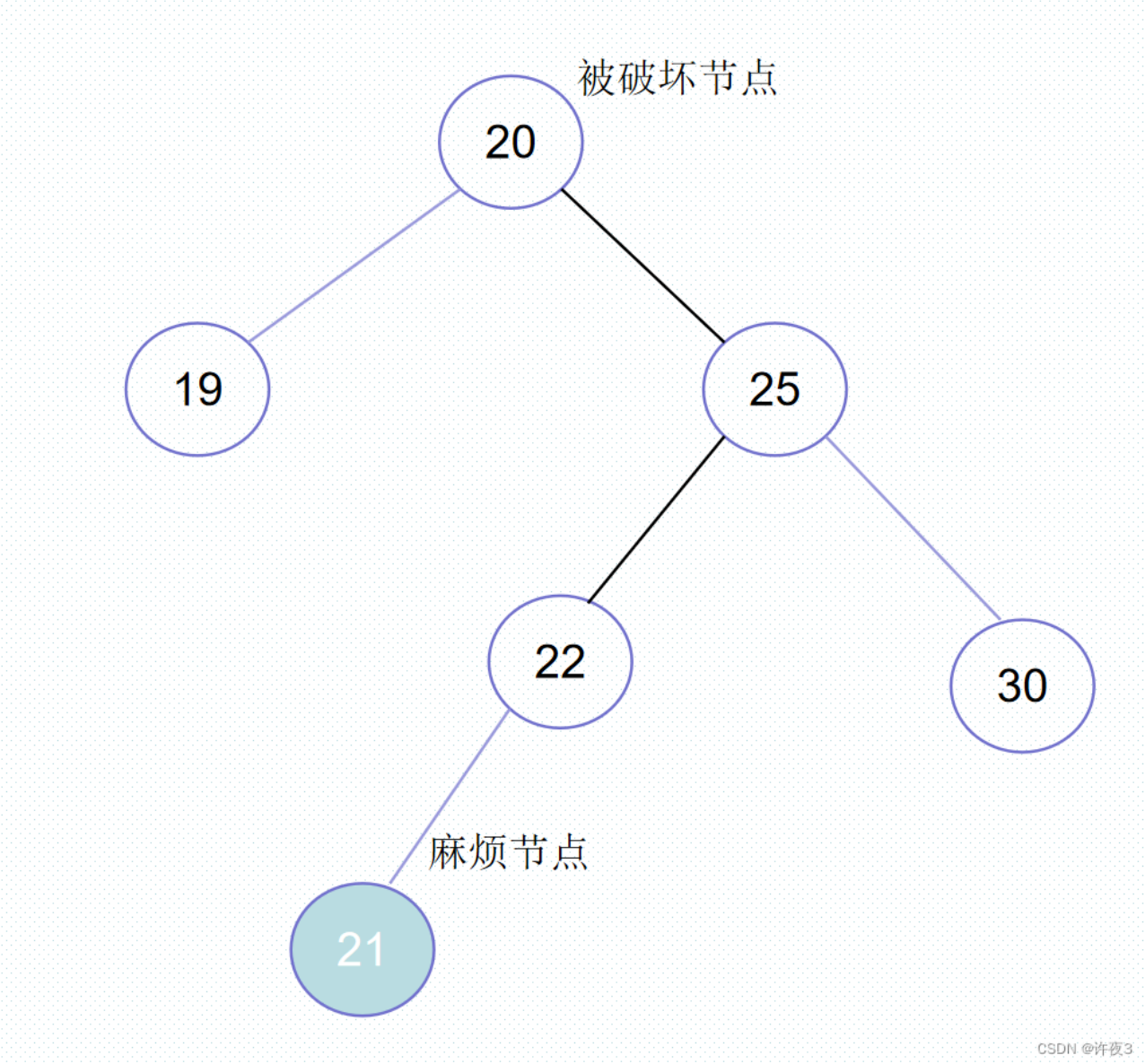


再以25为基础进行一次右旋，可以看出经过两次旋转后，该树已经平衡。



3.5 RL型

如图：

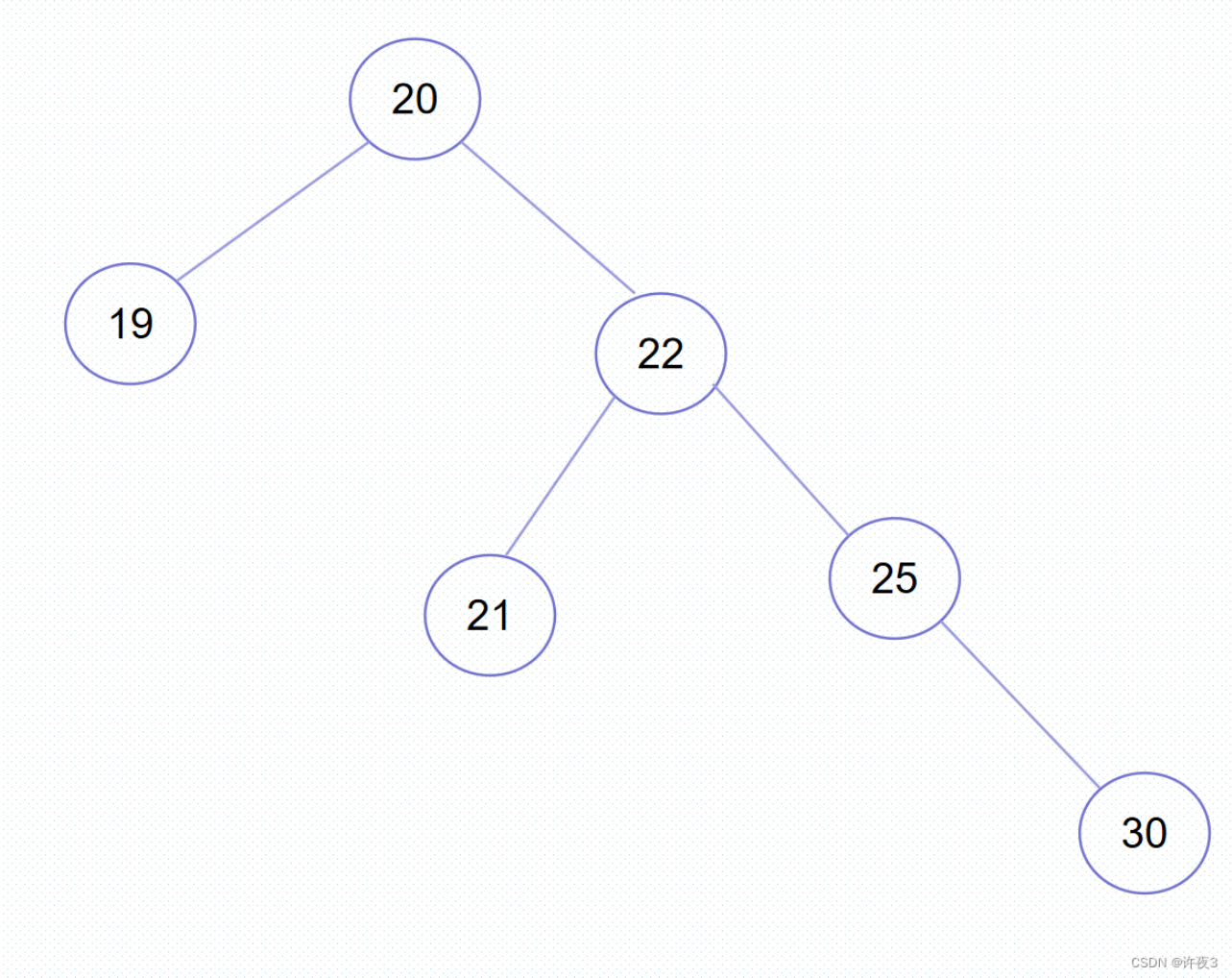


如上图，RL型也为右左型，在被破坏节点的右边的左边插入而导致失衡，则为RL型，注意上图，黑色的线表示它型号判断的路径。

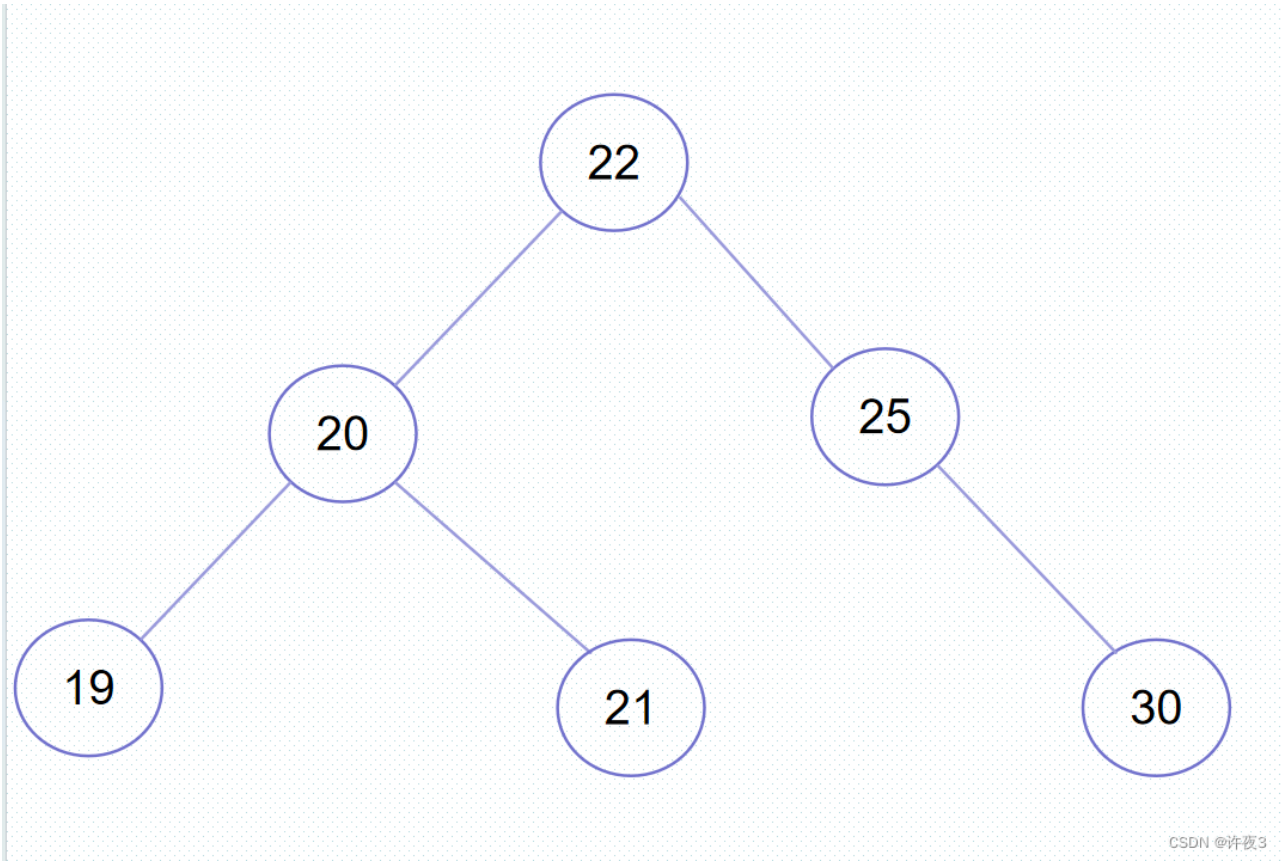
RL型解决方案：以被破坏节点R（右）节点为基础先进行一次R（右）旋，再以被破坏节点为基础进行左旋。

上图调整：

先以25为基础进行一次右旋。



再以20为基础进行一次右旋，可以看出经过两次旋转后，该树已经平衡。



3.6 总结

型号的判断永远是根据被破坏节点到麻烦节点的路径方向来判断，并且只判断两次即可，第三次不管。

处理方式分类两大类：

1.RR型和LL型，以被破坏节点为基础进行其反向的旋转即可，即RR型进行左旋，LL型进行右旋。

2.RL型和LR型，先以被破坏节点的LR或RL首字母的节点进行LR或RL首字母旋转，再以被破坏节点为基础进行LR或RL尾字母旋转，即RL型先以被破坏节点的R（右）节点为基础进行一次R（右）选，再以被破坏节点为基础进行一次L（左）旋；LR旋先以被破坏节点的L（左）节点为基础进行一次L（左）选，再以被破坏节点为基础进行一次R（右）旋。

总结内容，背下即可。

四、删除详解

在第三部分已经对插入进行了详解，其实平衡二叉树的删除比插入更复杂，但也是分为大致六类（其实是三类，左右对称）。

因为平衡二叉树也是一颗二叉查找树，它是在二叉查找树的删除的基础上添加了平衡调整的内容，对于二叉树删除的操作，本文不再概括，如果有对二叉树删除不了解的，可以看我上一篇关于二叉查找树详解博文，本文对于平衡二叉树的删除着重于平衡性的判断和调整。

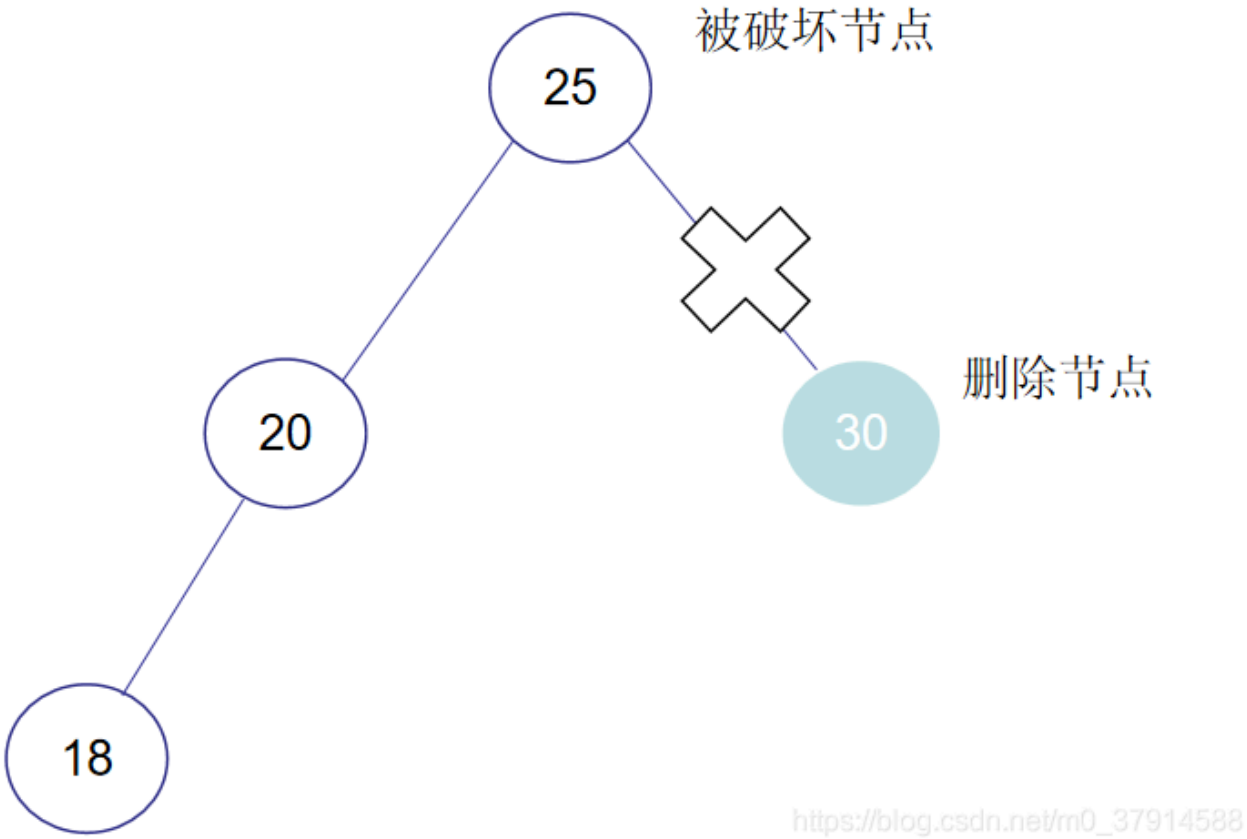
删除的节点在右子树：

即在右边删除而导致树失衡，此时左子树高度会大于右子树，右子树删除有三种调节方式。

注意：在右子树进行删除一个节点而导致失衡，则相当于在左子树插入一个新节点而导致的失衡，所以需要进行平衡性调整应该从左子树入手。

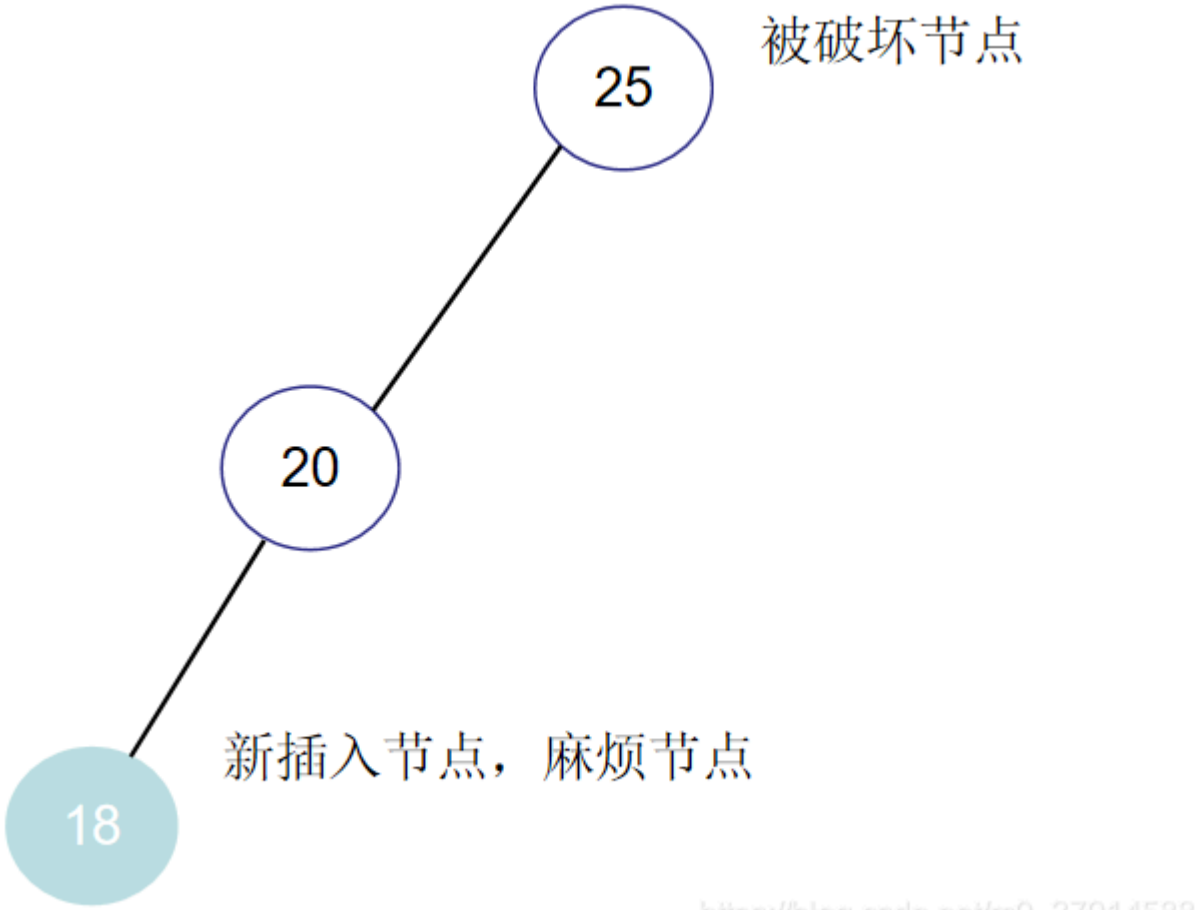
第一种：删除后被破坏节点的左节点的左边高度大于右边高度。

如图：

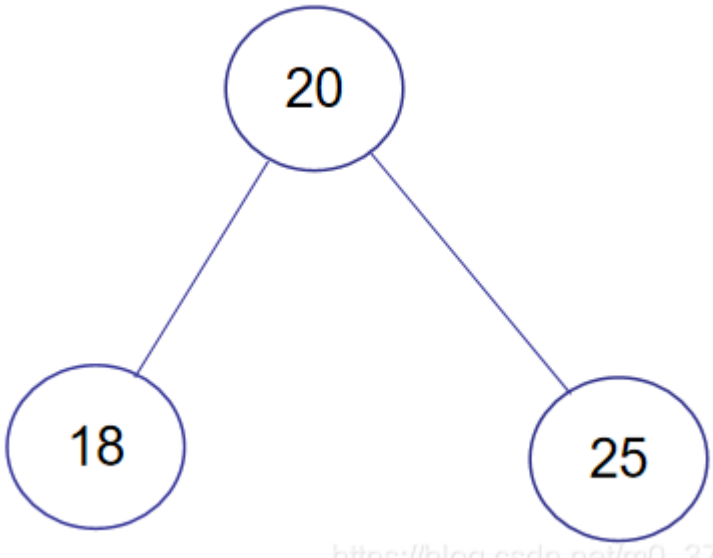


删除节点30后，被破坏节点的左节点，即节点20，它左边高度为1，而右边高度为0.。则相当于在节点20的左边插入一个节点18而导致的不平衡，此时即为LL型，进行右旋即可。

相当于下图：



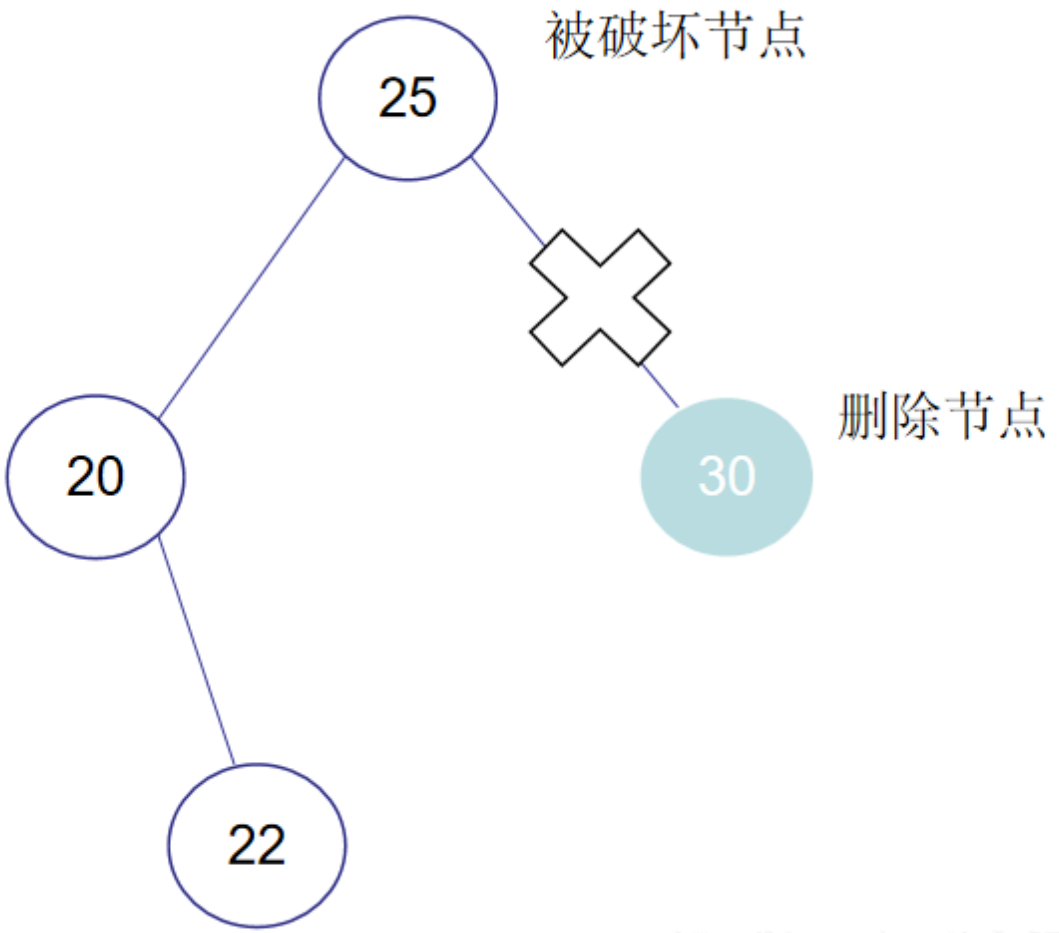
右旋后如图，可以看出已经平衡。



第一种结论：当在右子树删除而导致树失衡时，判断被破坏节点的左节点的左边高度和右边高度，如果左边高度大于右边高度，则相当于在被破坏节点的左节点的左节点下插入一个新的节点而导致的不平衡，即LL型插入，此时进行右旋调整即可。

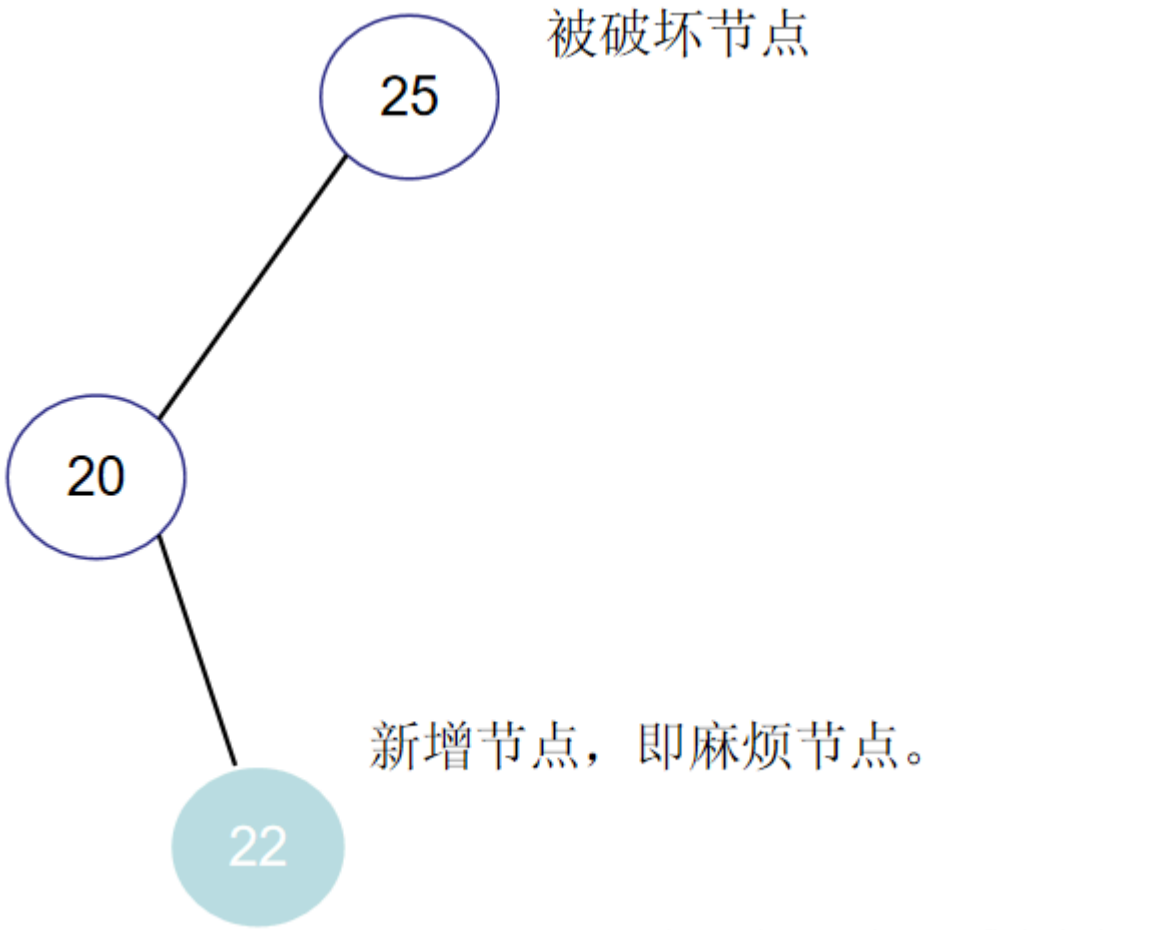
第二种：删除后被破坏节点的左节点的左边高度小于右边高度。

如图：

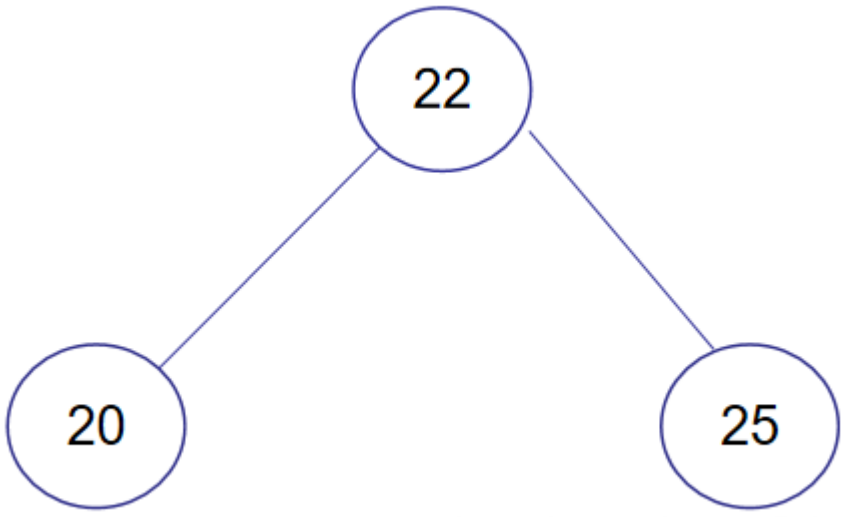


删除节点30后，被破坏节点的左节点，即节点20，它左边高度为0，而右边高度为1.。则相当于在节点20的右边插入一个节点22而导致的不平衡，此时即为LR型，先对被破坏节点的L（左）节点进行L（左）旋，再对被破坏节点进行R（右）旋即可。

相当于下图：



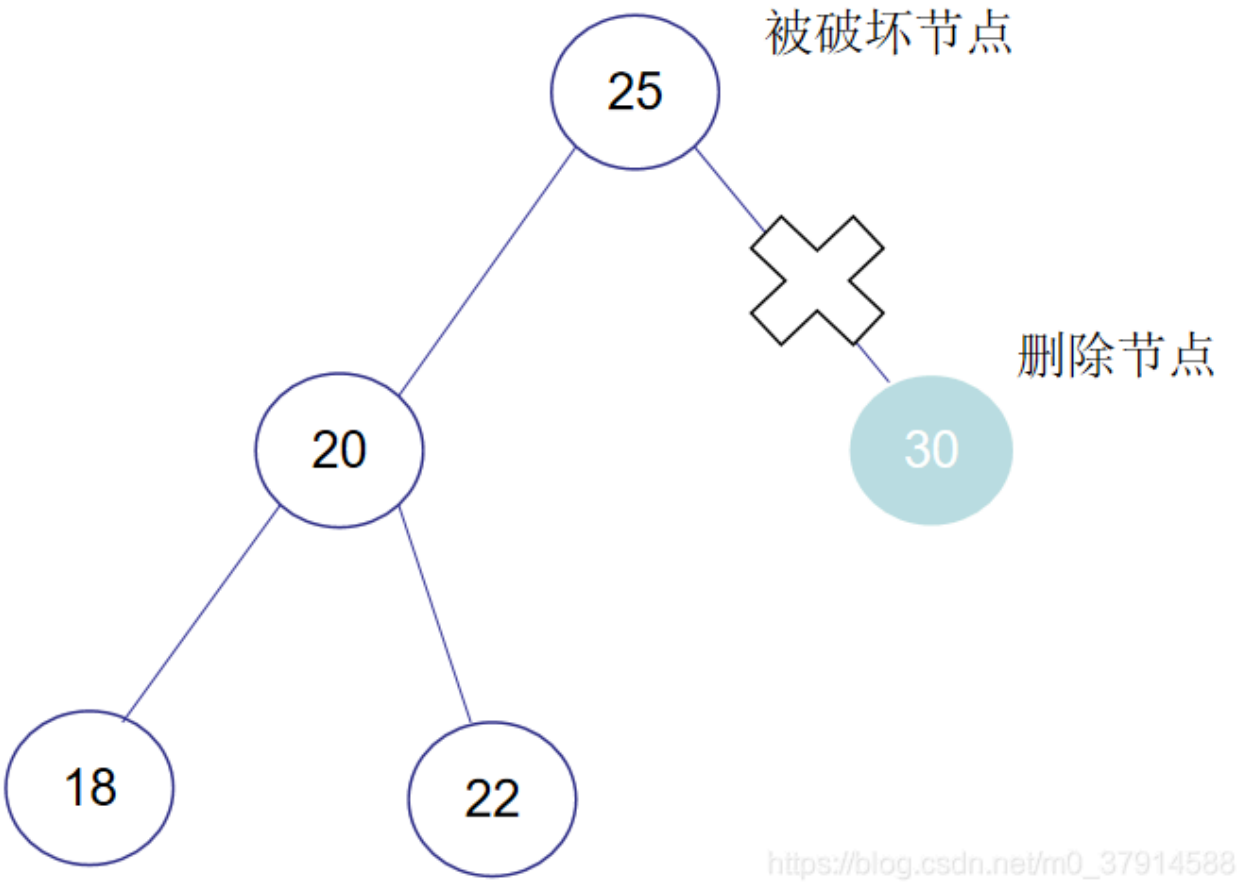
先以20为基础进行左旋，再以25为基础进行右旋后如下图，可以看出已经平衡。



第二种结论：当在右子树删除而导致树失衡时，判断被破坏节点的左节点的左边高度和右边高度，如果左边高度小于右边高度，则相当于在被破坏节点的左节点的右节点下插入一个新的节点而导致的不平衡，即LR型插入，此时先对被破坏节点的L（左）节点进行L（左）旋，再对被破坏节点进行R（右）旋即可。

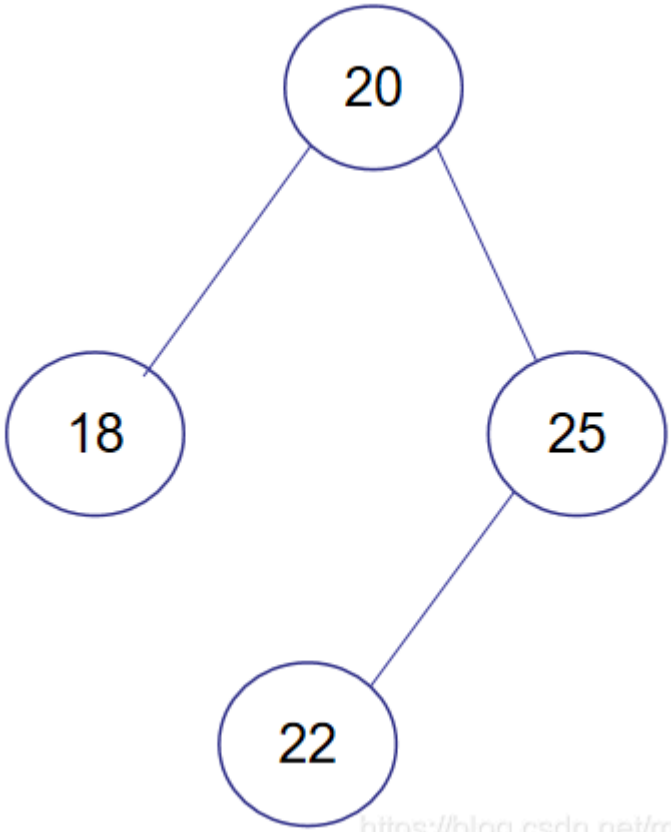
第三种：删除后被破坏节点的左节点的左边高度等于右边高度。

如图：



删除节点30后，被破坏节点的左节点，即节点20，它左边高度为1，而右边高度为1，高度相等，因为在右子树进行删除，此时对被破坏节点进行右旋即可，此种情况背下即可。

以25为基础进行右旋，如下图，可以看出已经平衡。



第三种结论：当在右子树删除而导致树失衡时，判断被破坏节点的左节点的左边高度和右边高度，如果左边高度等于右边高度，因为在右子树进行删除，此时对被破坏节点进行右旋即可。

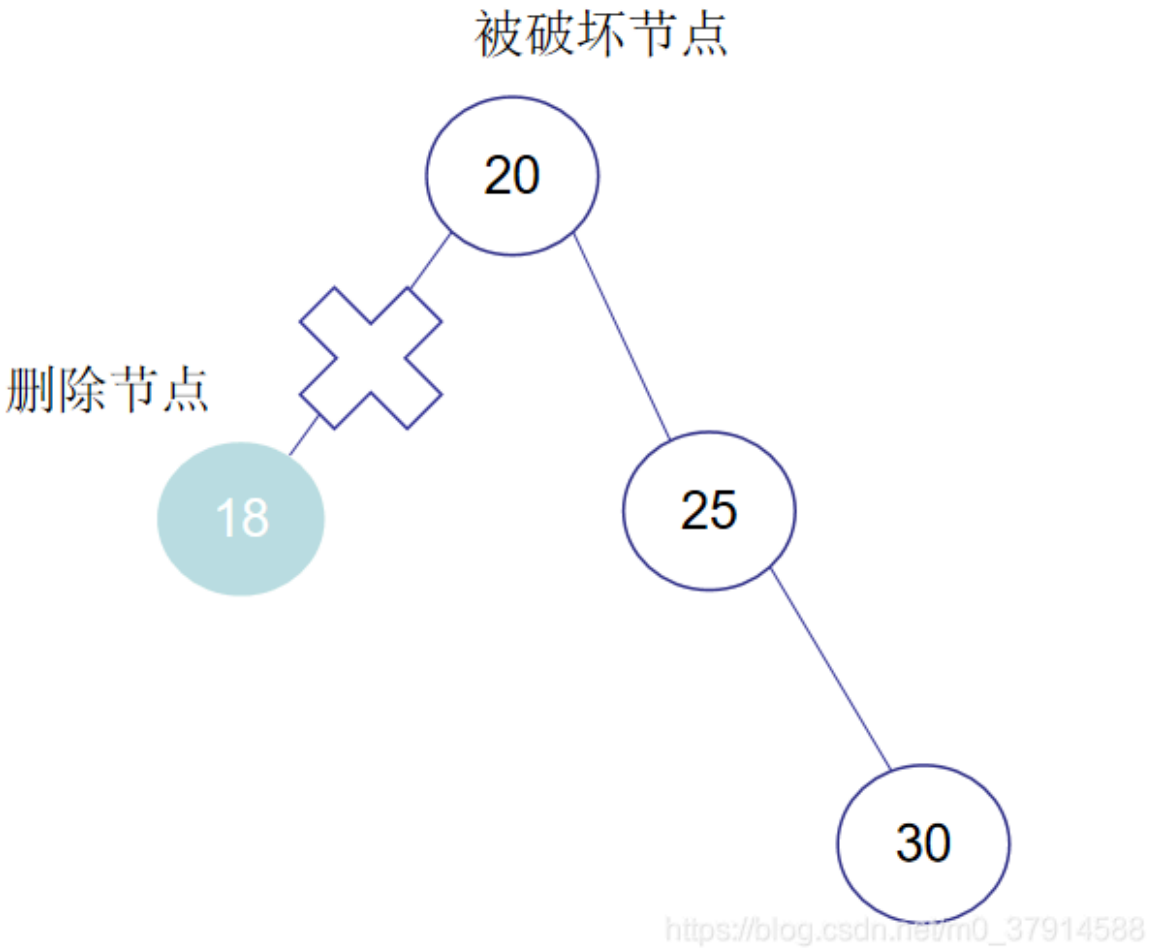
删除的节点在左子树：

即在左边删除而导致树失衡，此时右子树高度会大于左子树，左子树删除有三种调节方式。

注意：在左子树进行删除一个节点而导致失衡，则相当于在右子树插入一个新节点而导致的失衡，所以需要进行平衡性调整应该从右子树入手。

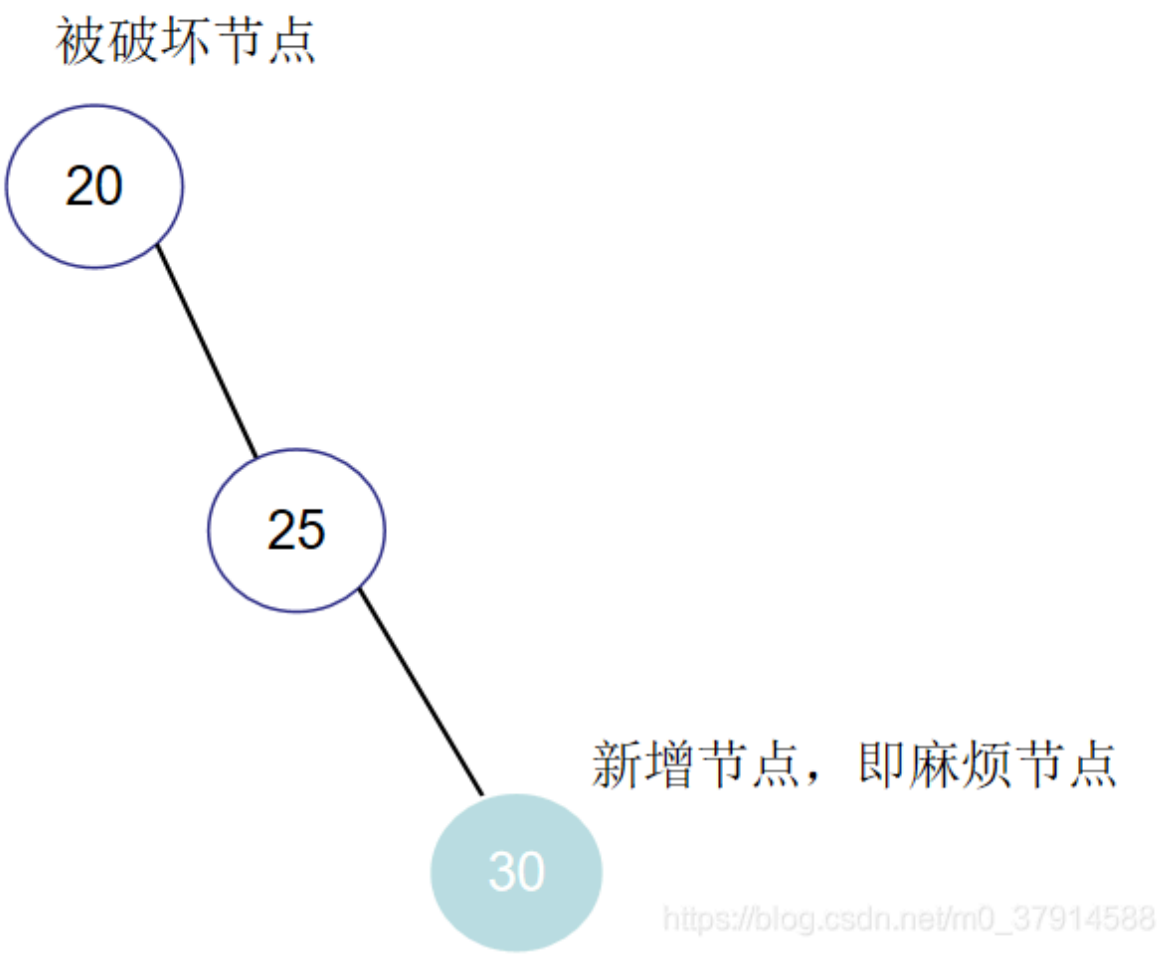
第一种：删除后被破坏节点的左节点的左边高度小于右边高度。

如图：

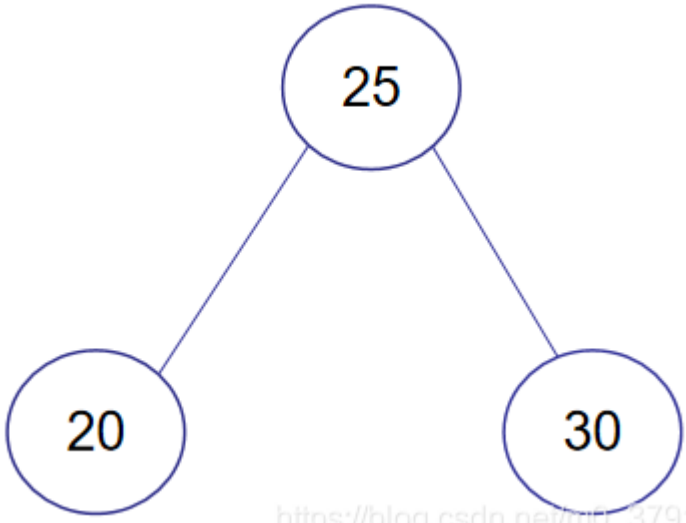


删除节点18后，被破坏节点的右节点，即节点25，它左边高度为0，而右边高度为1.。则相当于在节点25的右边插入一个节点30而导致的不平衡，此时即为RR型，进行左旋即可。

相当于下图：



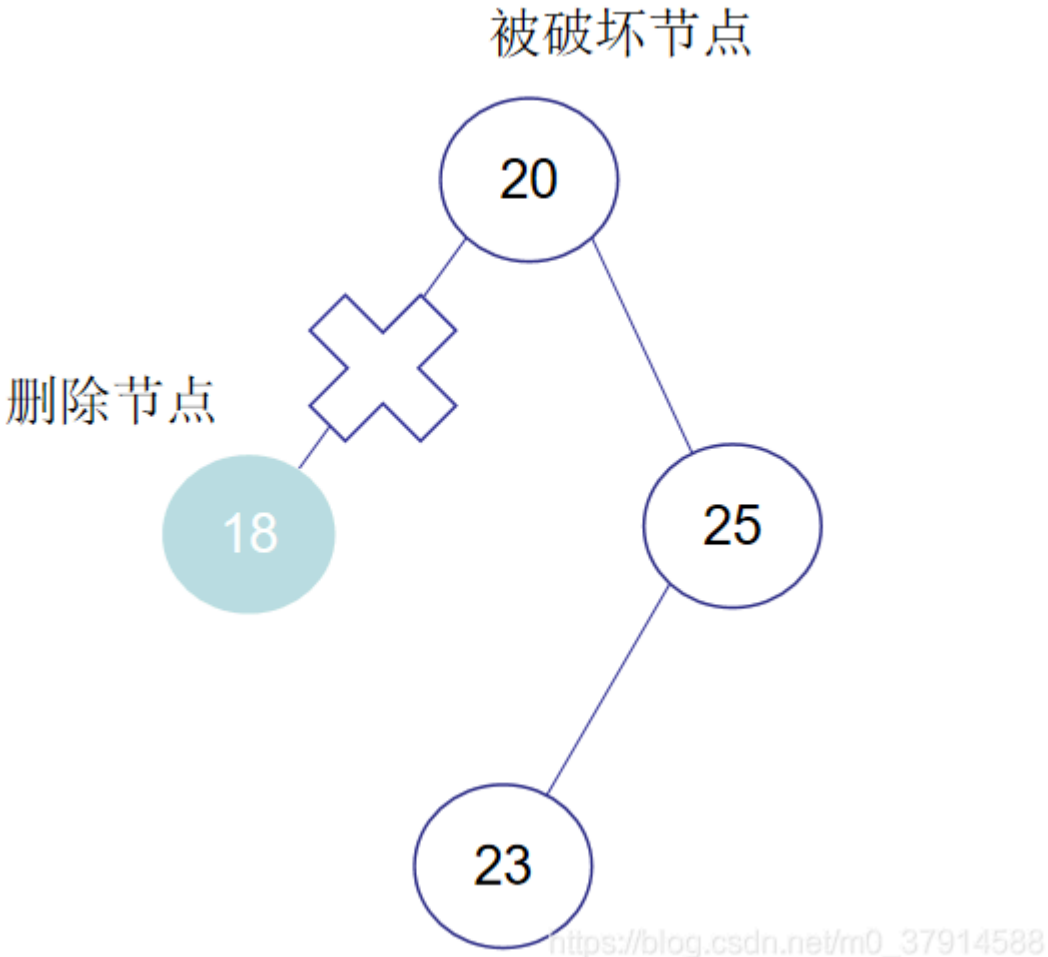
右旋后如图，可以看出已经平衡。



第一种结论：当在左子树删除而导致树失衡时，判断被破坏节点的右节点的左边高度和右边高度，如果左边高度小于右边高度，则相当于在被破坏节点的右节点的右节点下插入一个新的节点而导致的不平衡，即RR型插入，此时进行左旋调整即可。

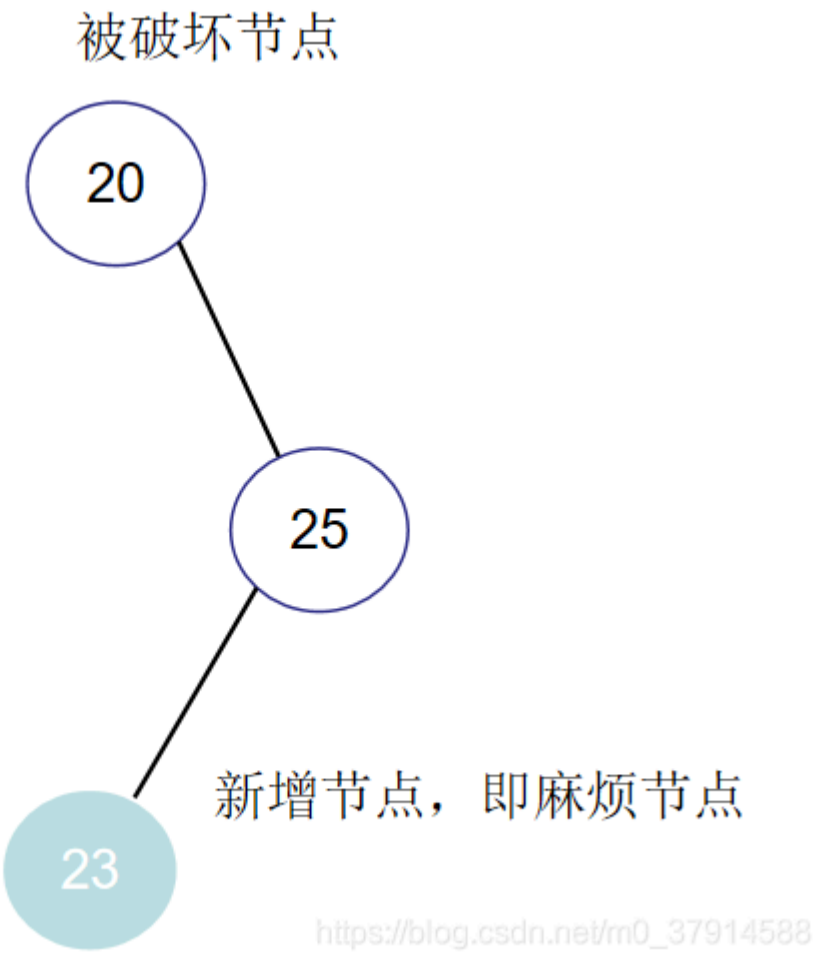
第二种：删除后被破坏节点的左节点的左边高度大于右边高度。

如图：

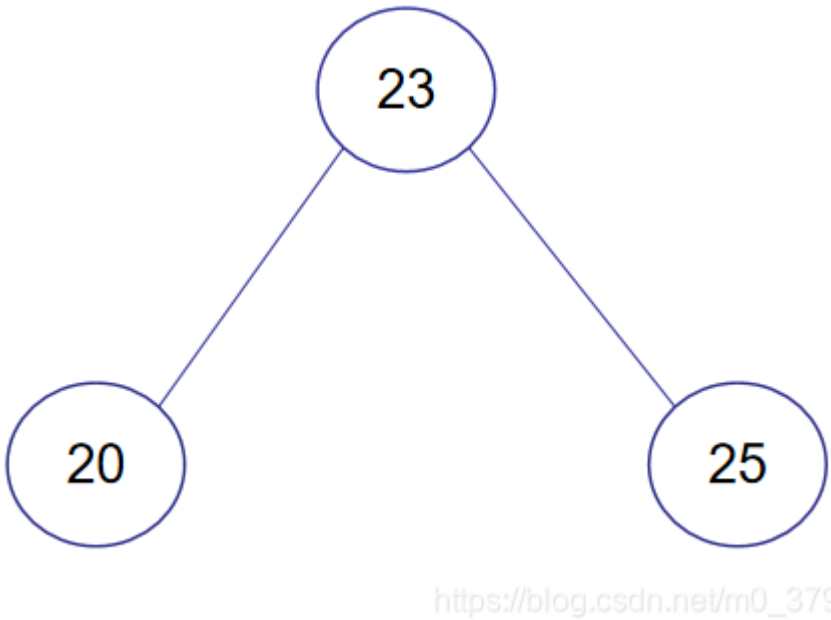


删除节点18后，被破坏节点的右节点，即节点25，它左边高度为1，而右边高度为0。则相当于在节点25的左边插入一个节点23而导致的不平衡，相当于RL型插入，此时先对被破坏节点的R（右）节点进行R（右）旋，再对被破坏节点进行L（左）旋即可。

相当于下图：



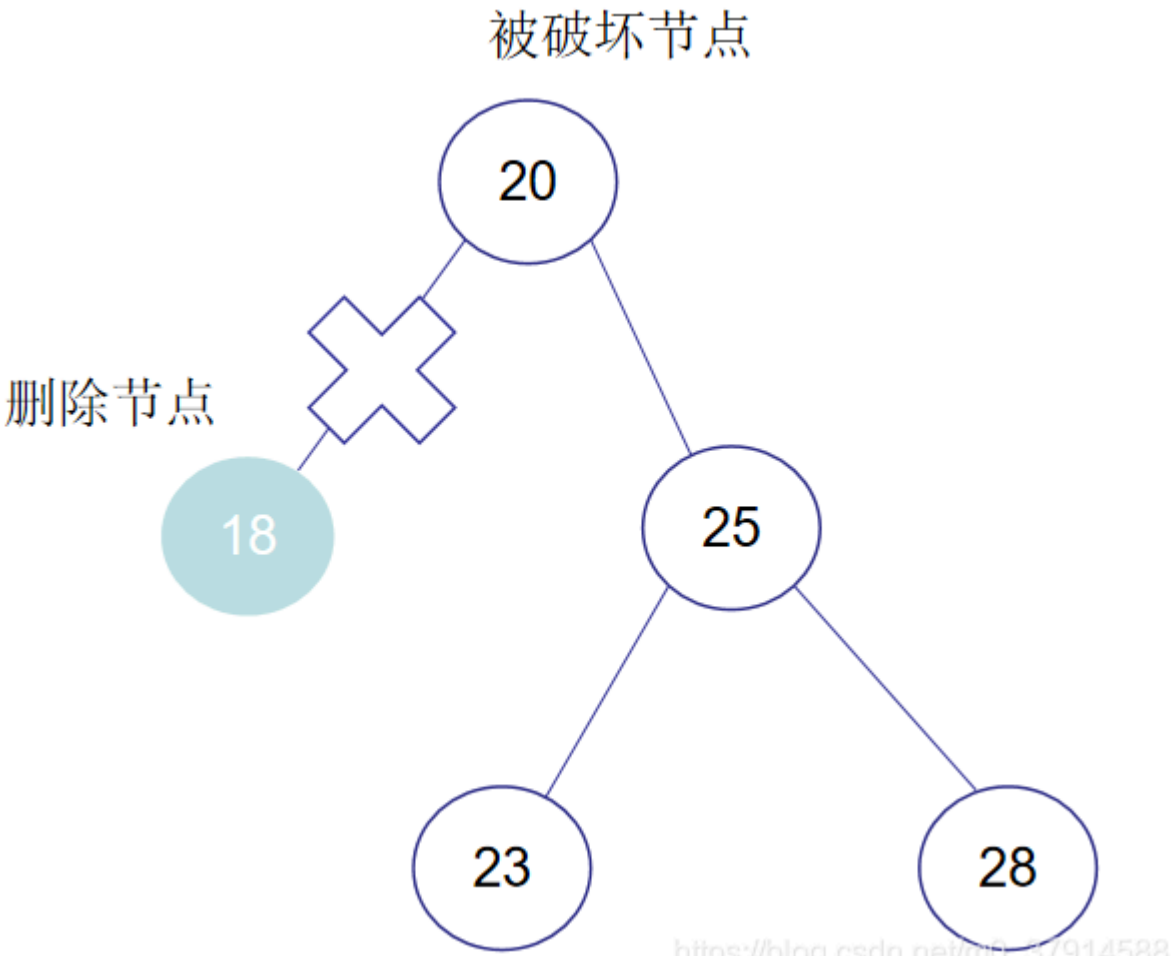
先以25为基础进行右旋，再以20为基础进行左旋后如下图，可以看出已经平衡。



第二种结论：当在左子树删除而导致树失衡时，判断被破坏节点的右节点的左边高度和右边高度，如果左边高度大于右边高度，则相当于在被破坏节点的右节点的左节点下插入一个新的节点而导致的不平衡，即RL型插入，此时先对被破坏节点的R（右）节点进行R（旋）旋，再对被破坏节点进行L（左）旋即可。

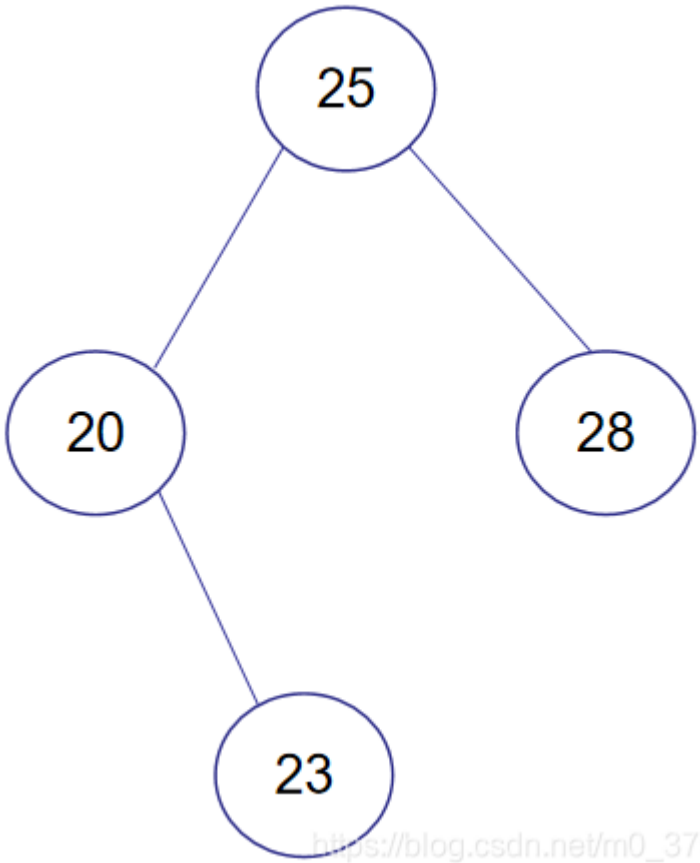
第三种：删除后被破坏节点的左节点的左边高度等于右边高度。

如图：



删除节点18后，被破坏节点的右节点，即节点25，它左边高度为1，而右边高度为1，因为在左子树进行删除，此时对被破坏节点进行左旋即可，此种情况背下即可。

以20为基础进行左旋，如下图，可以看出已经平衡。



第三种结论：当在左子树删除而导致树失衡时，判断被破坏节点的左节点的左边高度和右边高度，如果左边高度等于右边高度，因为在左子树进行删除，此时对被破坏节点进行左旋即可。

结论：

在某一边删除而导致失衡时，则判断被破坏节点的某节点的左右高度，如果高度不一致，则相当于在某边+高度高的那一边进行的插入，根据类型判断对应的旋转；如果高度一致，则对被破坏节点进行删除某旋即可。

注意：某代表左或右，如果在左边删除则为左。

五、遍历概括

平衡二叉树的遍历和二叉搜索树的遍历是一致的，请看上一篇博文的遍历描述即可。