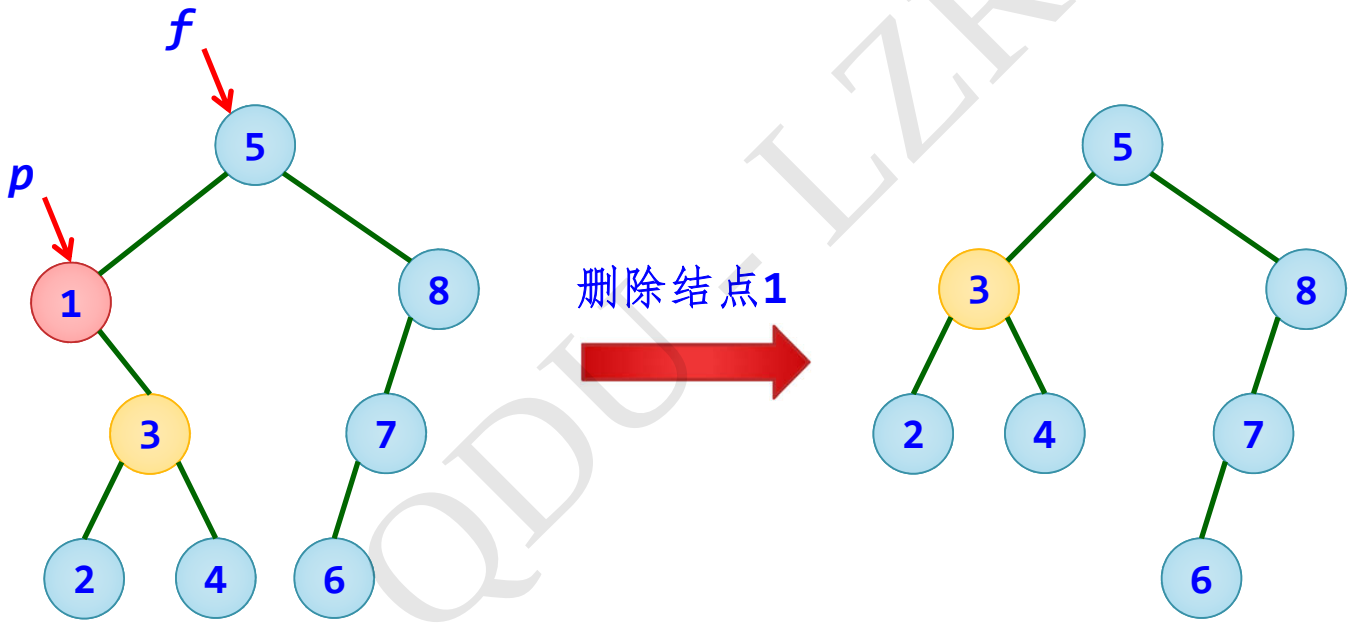
9.2.1 二叉排序树

1. 二叉排序树的删除

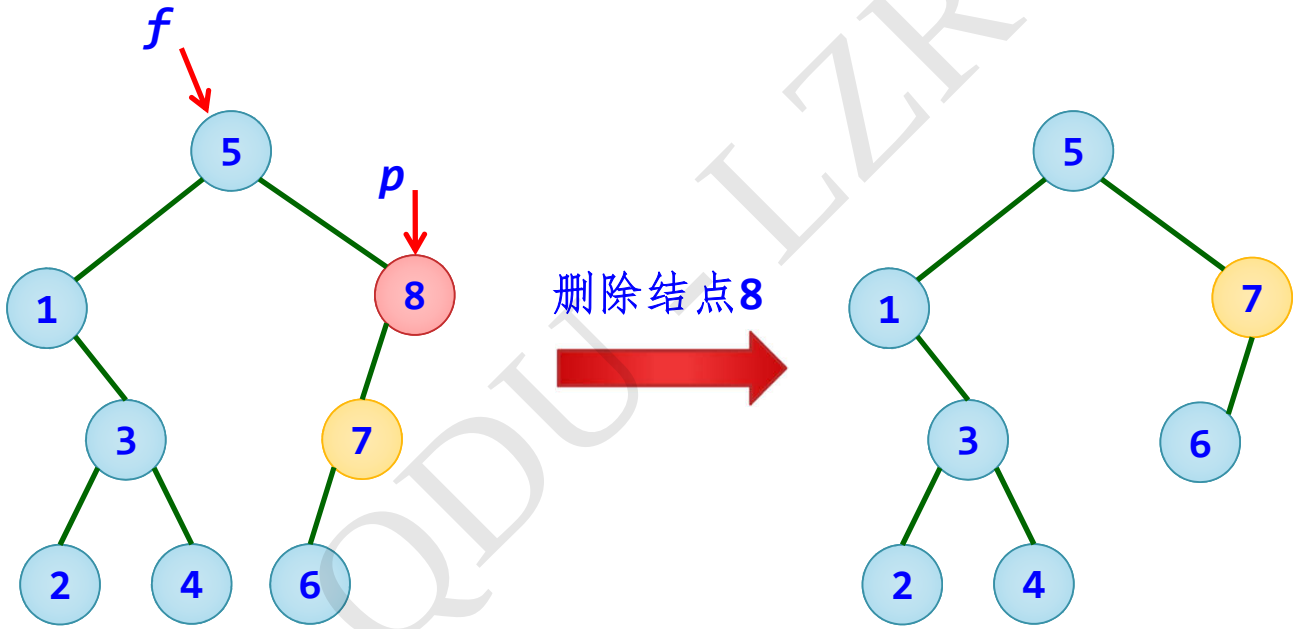
在二叉排序树T中删除关键字为*k*的结点后，仍需要保持二叉排序树的特性。

- 先在二叉排序树T中查找关键字为*k*的结点*p*，用*f*指向其双亲结点。
- 删除*p*结点分以下三种情况。

（1）若*p*结点没有左子树（含*p*为叶子结点的情况），则用*p*结点的右孩子替换它。



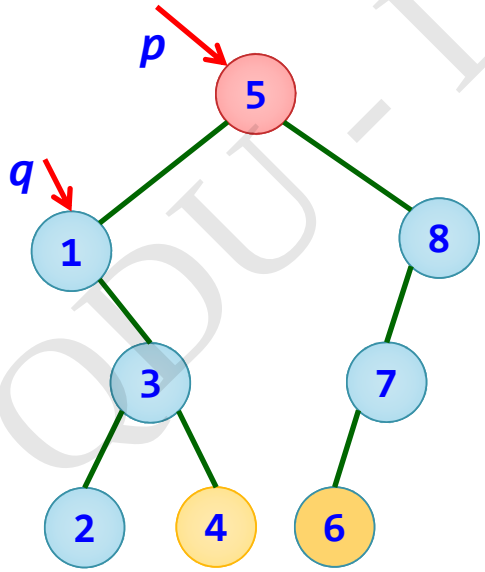
（2）若*p*结点没有右子树，则用*p*结点的左孩子替换它。



（3）若*p*结点既有左子树又有右子树。

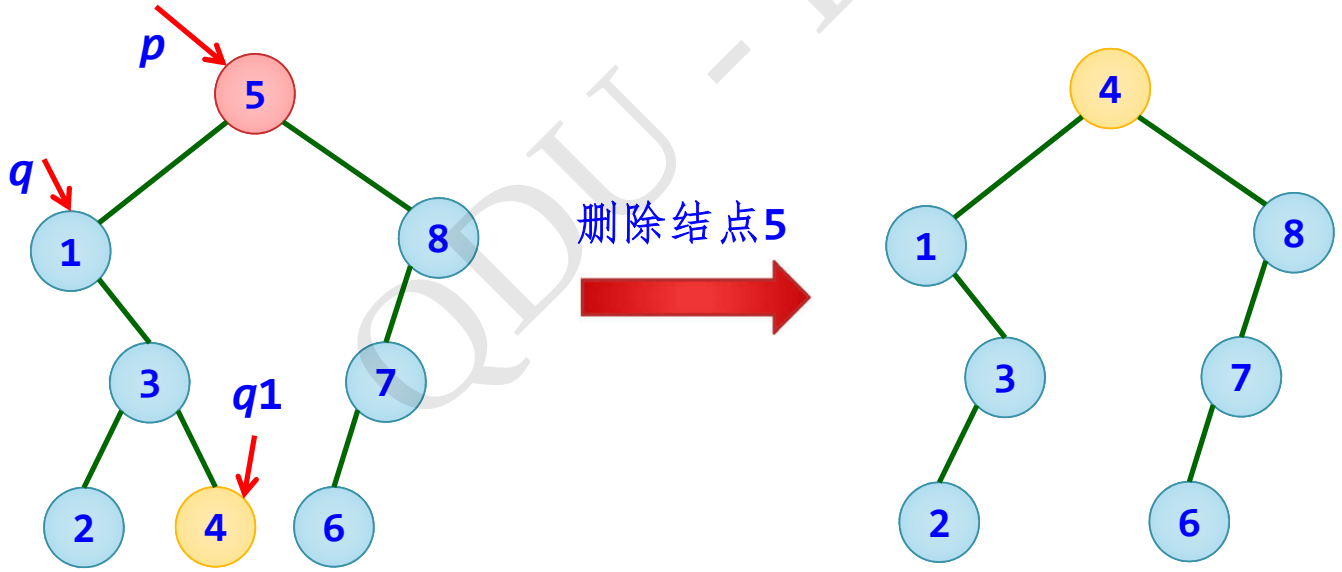
对于如图的二叉排序树，其中序序列为：

{1 2 3 4 5 6 7 8}



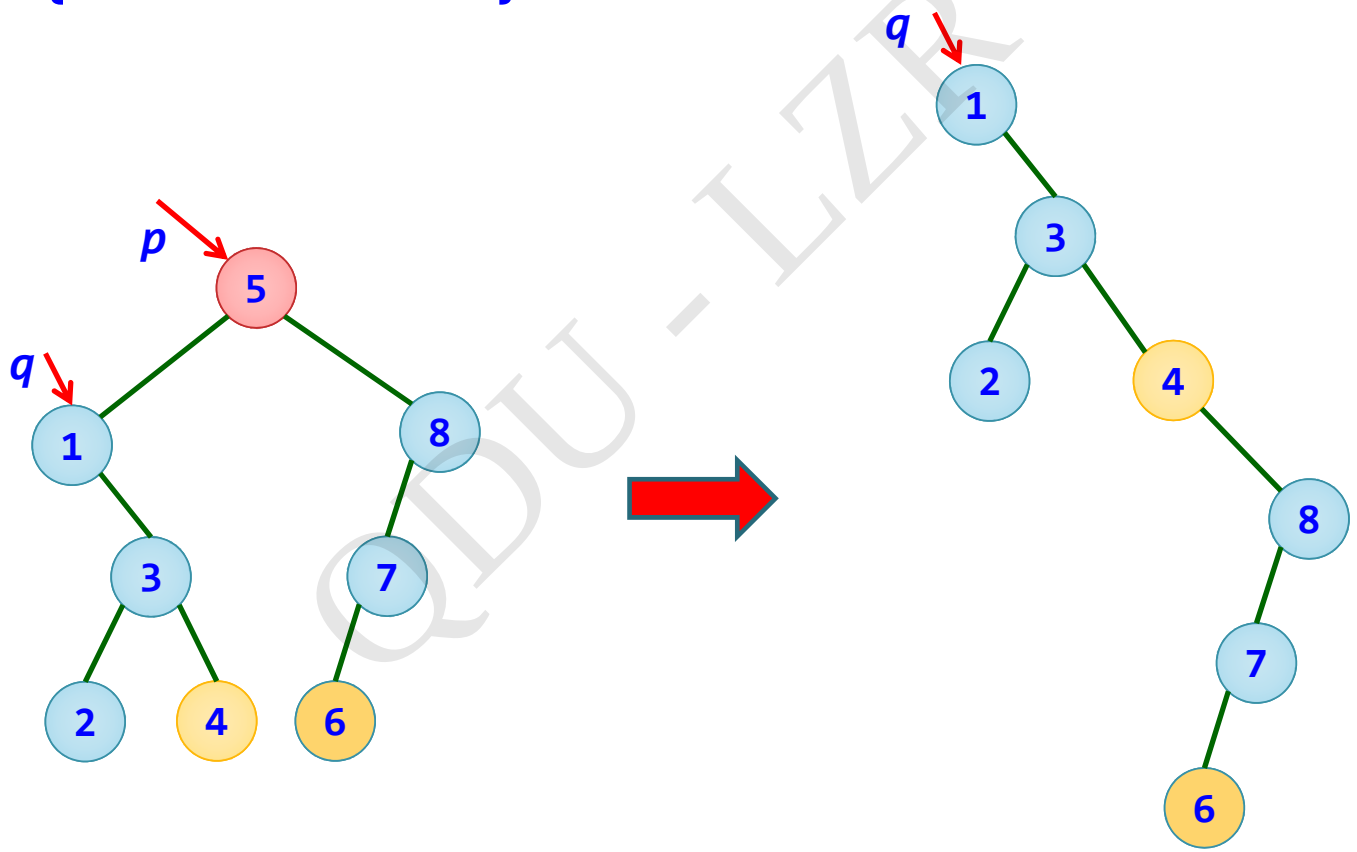
（3）若*p*结点既有左子树又有右子树，用其左子树中最大的结点替代它。

通过*p*结点的左孩子*q*找到它的最右下结点*q*1，*q*1结点就是*p*结点左子树中最大的结点，将*q*1结点值替代*p*结点值，然后将*q*1结点删除。由于*q*1结点一定没有右孩子，可以采用（2）的操作删除结点*q*1。



（3）若p结点既有左子树又有右子树。

对于如图的二叉排序树，其中序序列为：{1 2 3 4 5 6 7 8}。P230



删除结点算法

```
void Delete(BiTree &p)
{
    // 从二叉排序树中删除结点p，并重接它的左或右子树。算法9.8
    BiTree q, s;
    // p的右子树空则只需重接它的左子树(待删结点是叶子也走此分支)
    if(!p->rchild) {
        q = p;
        p = p->lchild;
        free(q);
    }
    else
        if(!p->lchild) { // p的左子树空，只需重接它的右子树
            q = p;
            p = p->rchild;
            free(q);
        }
```

删除结点算法

```
        else { // p的左右子树均不空
                q = p;
                s = p->lchild;
                // 转左，然后向右到尽头(找待删结点的前驱)
                while(s->rchild) {
                    q = s;
                    s = s->rchild;
                }
               // s指向被删结点的"前驱"(将被删结点前驱的值取代被删结点的值)
                p->data = s->data;
                if(q != p)
                    q->rchild = s->lchild; // 重接*q的右子树
                else
                    q->lchild = s->lchild; // 重接*q的左子树
                free(s);
          }
}
```

删除结点算法

```
Status DeleteBST(BiTree &T, KeyType key)
{
    // 若二叉排序树T中存在关键字等于key的数据元素时，则删除该数据元素结点，
    // 并返回TRUE；否则返回FALSE。算法9.7
    if(!T)                          // 不存在关键字等于key的数据元素
        return FALSE;
    else {
        if EQ(key, T->data.key) // 找到关键字等于key的数据元素
            Delete(T);
        else
            if LT(key, T->data.key)
                DeleteBST(T->lchild, key);
            else
                DeleteBST(T->rchild, key);
        return TRUE;
    }
}
```

END