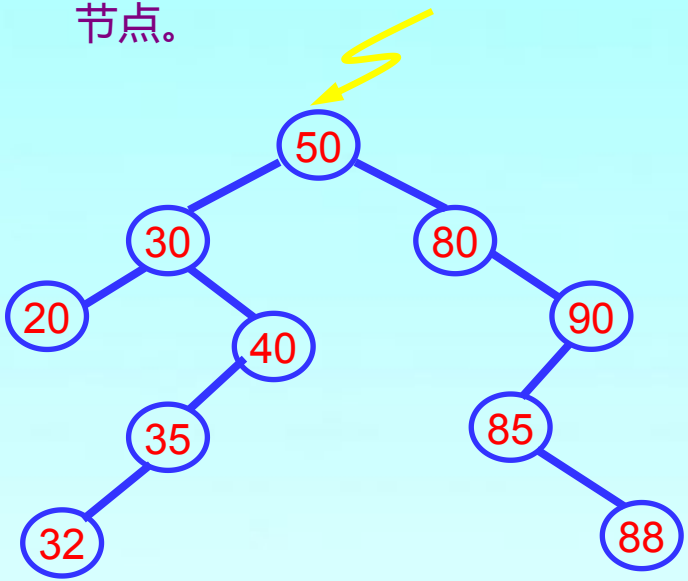
本节主题：

二叉排序树(续)

二叉排序树的节点删除

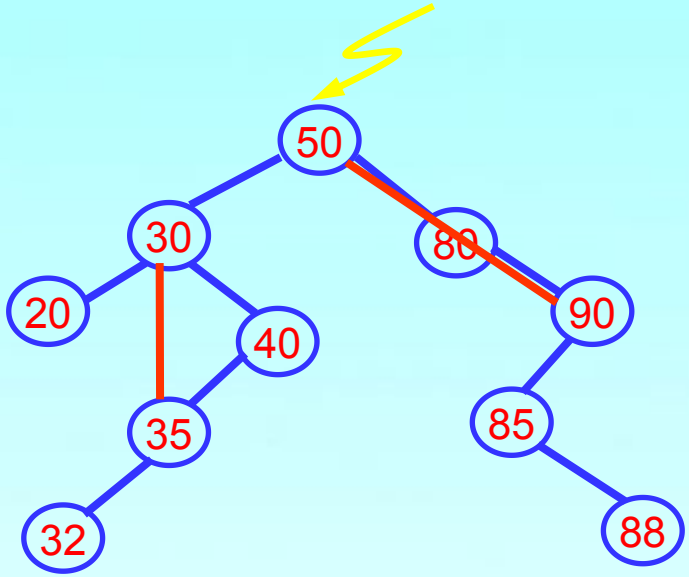
(1)被删除的节点是叶子节点：直接删去该节点。



被删关键字 = 20　88

其双亲节点中相应指针域的值改为“空”

(2)被删除的节点只有左子树，用其左子树代替；只有右子树，用其右子树代替它。



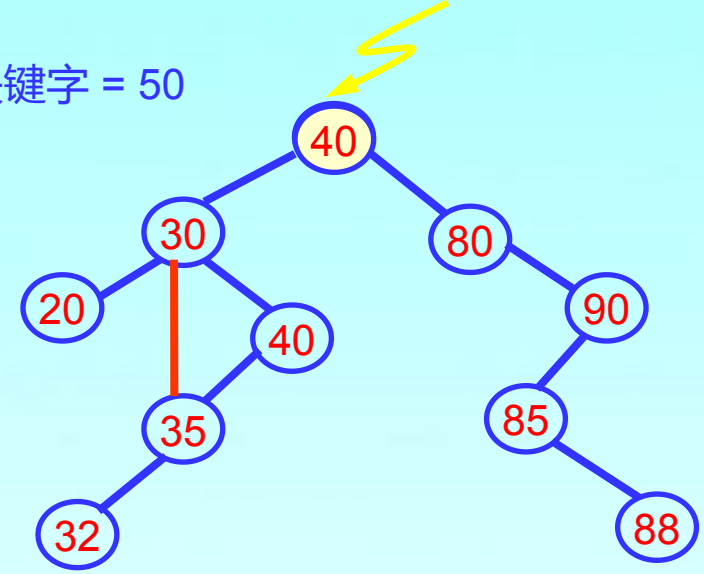
被删关键字 = 40　80

其双亲节点的相应指针域的值改为 “指向被删除节点的左子树或右子树”。

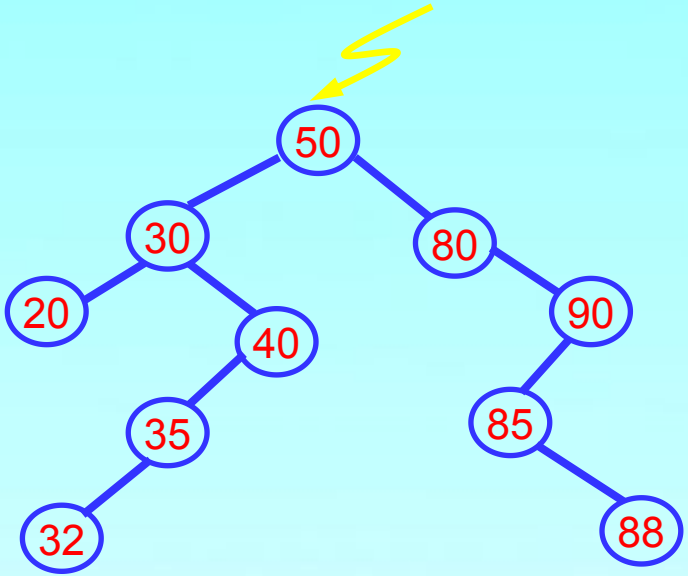
二叉排序树的节点删除(续)

（3）被删除的节点既有左子树，也有右子树

被删关键字 = 50



前驱是左子树中最大的节点。
以其前驱替代之，然后再删除该前驱节点。

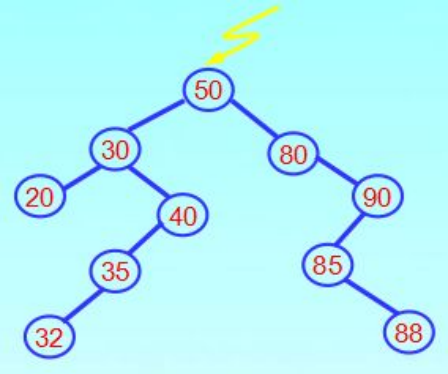


也可以用其后继替代之，然后再删除该后继节点。后继是右子树中最小的节点。

算法实现

```
int DeleteBST(BSTNode *&bt, KeyType k)
{
   if (bt==NULL)
      return 0;
   else
   {
      if (k<bt->key)
         return DeleteBST(bt->lchild,k);
      else if (k>bt->key)
         return DeleteBST(bt->rchild,k);
      else
      {
         Delete(bt);
         return 1;
      }
   }
}
```

```
void Delete(BSTNode *&p)
{
   BSTNode *q;
   if (p->rchild==NULL)
   {
      q=p;
      p=p->lchild;
      free(q);
   }
   else if (p->lchild==NULL)
   {
      q=p;
      p=p->rchild;
      free(q);
   }
   else
      Delete1(p,p->lchild);
}
```




```
void Delete1(BSTNode *p,BSTNode *&r)
{
   BSTNode *q;
   if (r->rchild!=NULL)
      Delete1(p,r->rchild);
   else
   {
      p->key=r->key;
      q=r;
      r=r->lchild;
      free(q);
   }
}
```