二叉搜索树：

【 思路：

1.二叉搜索树具有一个很好的特点。以当前结点为根节点的左边结点的值都是小于根节点的值，右边结点的值都大于根节点的值。

2.根据这个特点，如果给的两个节点的值都小于根节点，那么它们的最低公共祖先就一定在它左子树。

3.如果给的两个节点的值都大于根节点，那么它们的最低公共祖先就一定在它右子树。

4.如果一个结点的值大于根节点的值，一个结点的值小于根节点的值，那么这个根节点就是它的最低公共祖先。 】

代码:

```
1.  //求两个节点的最低公共祖先(递归解法)
2.  BSTreeNode* GetCommenParent_R(BSTreeNode* root, BSTreeNode* bstn1, BSTreeNod
    e* bstn2)
3.  {
4.      if (root == NULL || bstn1 == NULL || bstn2 == NULL)
5.          return NULL;
6.
7.      if ((bstn1->_data < root->_data) && (bstn2->_data< root->_data))
8.      {
9.          GetCommenParent_R(root->_left, bstn1, bstn2);
10.     }
11.     else if ((bstn1->_data>root->_data) && (bstn2->_data>root->_data))
12.     {
13.         GetCommenParent_R(root->_right, bstn1, bstn2);
14.     }
15.     else
16.         return root;
17. }
```

普通二叉树：

【思路：

如果一个结点为根，另一个结点无论在什么地方它们的最低公共祖先一定为根结点。

如果一个结点在左树，另一个结点在右树，那么它的最低公共祖先一定是根节点。

如果两个结点都在左树，以子问题在左树查找。

如果两个结点都在右树，以子问题在右树查找。】

代码：

```
1.  //找一个结点
2.  struct TreeNode* GetNode(struct TreeNode* root, struct TreeNode* cur){
3.      if(root==NULL){
4.          return NULL;
5.      }
6.      if(root->val==cur->val){
7.          return root;
```

```
    }

    struct TreeNode* ret=GetNode(root->left,cur);
    if(ret){
        return ret;
    }
    return GetNode(root->right,cur);
}
struct TreeNode* lowestCommonAncestor(struct TreeNode* root, struct TreeNode* p, struct TreeNode* q) {
    if(root->val==p->val||root->val==q->val){
        return root;
    }
    bool pleft,pright,qleft,qright;

    if(GetNode(root->left,p)){
        pleft=true;
        pright=false;
    }
    else{
        pleft=false;
        pright=true;
    }
    if(GetNode(root->left,q)){
        qleft=true;
        qright=false;
    }
    else{
        qleft=false;
        qright=true;
    }
    //一个在左，一个在右，返回根
    if((pleft&&qright)||(pright&&qleft)){
        return root;
    }
    //两个都在左
    if(pleft&&qleft){
        return lowestCommonAncestor(root->left,p,q);
    }
    //两个都在右
    if(pright&&qright){
        return lowestCommonAncestor(root->right,p,q);
    }

```

```
51.     return NULL;
52. }
```