

查找

代码题 1

题干

- 题目：

1、顺序表递增有序，设计算法在最少的时间
内查找值为 x 的元素。若找到，则将其于后继
元素位置交换，否则按照递增顺序插入顺序表

- 总结：

- 第一步：二分查找到值为 x 的元素的位置
- 第二步：如果找到，交换
- 第三步：如果 `low < high`，则表示没找到，后移

解答

```
void Insert(Sqlist& list,int x)
{
    int low = 0;
    int high = list.lenght-1;
    int mid;

    // 找到元素目标位置
    while(low<=high)
    {
        mid = (low+high)/2; // 向下取整

        // 找到目标位置
        if(list.data[mid] == x)
        {
            break;
        }

        if(list.data[mid] < x)
        {
            low = mid+1;
        }
        if(list.data[mid] > x)
        {
            high = mid-1;
        }
    }
}
```

// 如果当前位置的元素就等于当前要找的值，并且不是最后一个元素

```

if(list.data[mid] == x && mid!=list.length-1)
{
    // 交换前后元素
    int temp = x;
    list.data[mid] = list.data[mid+1];
    list.data[mid+1] = temp;
    return;
}

// 如果当前x元素不在顺序表中，将其插入顺序表，并且将插入位置后的元素全部后移
if(low>high)
{
    int i;
    for(i = list.length-1;i>high;--i)
    {
        list.data[i+1] = list.data[i];
    }

    list.data[i+1] = x;
    list.length++;
}
}

```

代码题 2

题干

- 题目：

2、找到单链表中值为 key 的元素，将其与前一个结点位置互换。

- 总结：

-

解答

```

void Fun(LNode* list,int key)
{
    // 如果第一个结点的值就为key，因为不能和头节点交换，因此直接返回
    if(list->next->data == key)
    {
        return;
    }

    while(list->next->next!=null)
    {
        // 找到目标结点
        if(list->next->next->data == key)
        {
            break;

```

```

    }
    else
    {
        list = list->next;
    }
}

if(list->next->next!=null)
{
    LNode* p = list->next;
    LNode* q = p->next;
    p->next = q->next;
    q->next = p;
    list->next = q;
}
}

```

代码题 3

题干

- 题目：
 - 3、在顺序表中二分查找值为 key 的元素
- 总结：
 -

解答：方法一 - 非递归

```

void Fun(Sqlist &list,int key)
{
    int low = 0;
    int high = list.length-1;
    int mid;

    while(low<=high)
    {
        mid = (low+high)/2;
        if(list.data[mid] == key)
        {
            return mid+1;
        }
        else if(list.data[mid] > key)
        {
            low = mid+1;
        }
        else
        {
            high = mid-1;
        }
    }
}

```

```
}  
}
```

解答：方法二 - 递归

```
void Fun(Sqlist &list,int key,int low,int high)  
{  
    if(low>high)  
    {  
        return -1;  
    }  
  
    int mid = (low+high)/2;  
  
    if(list.data[mid] > key)  
    {  
        Fun(list,key,mid+1,high);  
    }  
    else if(list.data[mid] < key()  
    {  
        Fun(list,key,low,mid-1);  
    }  
    else  
    {  
        return mid+1;  
    }  
}
```

代码题 4

题干

- 题目：

4、判断给定二叉树是否是二叉(搜索)排序树

- 总结：

- 二叉排序树的性质：（假设当前是 左<根<右 ）任何一个节点，其结点值都比左子树中最大的结点的值还要大。因此如果不满足此性质，则不为二叉排序树；
- 补充：空节点也是二叉排序树；
- LVR；
- 先左递归，判断左子树是否满足；
- 然后判断当前根节点是否满足，即比较 pre 和当前结点的值大小。如果当前节点大小大于 pre，则符合性质；
- 然后右递归；

解答

```

// pre表示的是当前已遍历结点中的最大值，比如对左子树递归后、Pre应当为左子树的最大值，然后要
// 进入右子树递归前，pre应当设置为当前根节点的值（当前结点）
int pre = -INT_MAX;
int IsBST(BTNode* bt)
{
    if(bt==null)
    {
        return 1;
    }

    // 左递归判断当前是否是二叉排序树，如果是的话返回1
    int a = IsBST(bt->lchild);

    if(a==0 || bt->data<=pre)
    {
        return 0;
    }
    else
    {
        pre = bt->data;
    }

    // 对右子树递归，递归的结果即为判定的结果
    int b = IsBST(bt->rchild);
    return b;
}

```

代码题 5

题干

- 题目：
 - **5、寻找二叉排序树中最大值和最小值**

- 总结：
 -

解答

```

int GetMax(BTNode* bt)
{
    if(bt==null)
    {
        return 0;
    }

    while(bt->rchild!=null)
    {
        bt = bt->rchild;
    }
}

```

```

        return bt->data;
    }

    int GeiMin(BTNode* bt)
    {
        if(bt==null)
        {
            return 0;
        }

        while(bt->lchild!=null)
        {
            bt = bt->lchild;
        }
        return bt->data;
    }
}

```

代码题 6

题干

- 题目：
 - 6、设求出指定结点在给定二叉排序树的层次
- 总结：
 -

解答

```

int GetLevel(BTNode* bt,int k)
{
    int level = 1;
    BTNode* p = bt;

    while(p!=null)
    {
        if(p->data==k)
        {
            return k;
        }
        // 根节点比目标更大，在左子树中找
        if(p->data>k)
        {
            p = p->lchild;
        }
        // 根节点比目标更小，在右子树中找
        if(p->data<k)
        {
            p = p->rchild;
        }
    }
}

```

```
    }  
    n++;  
}  
  
return -1; // 当前指定节点不存在在二叉排序树中  
}
```

代码题 7

题干

- 题目：
 - 7、输出二叉搜索树中所有值大于 key 的结点
- 总结：
 - 暴力，左右子树全部遍历；

解答

```
void Print(BTNode* bt,int key)  
{  
    if(bt->lchild!=null)  
    {  
        Print(bt->lchild,key);  
    }  
    if(bt->data>key)  
    {  
        cout<<bt->data;  
    }  
    if(bt->rchild!=null)  
    {  
        Print(bt->rchild,key);  
    }  
}
```