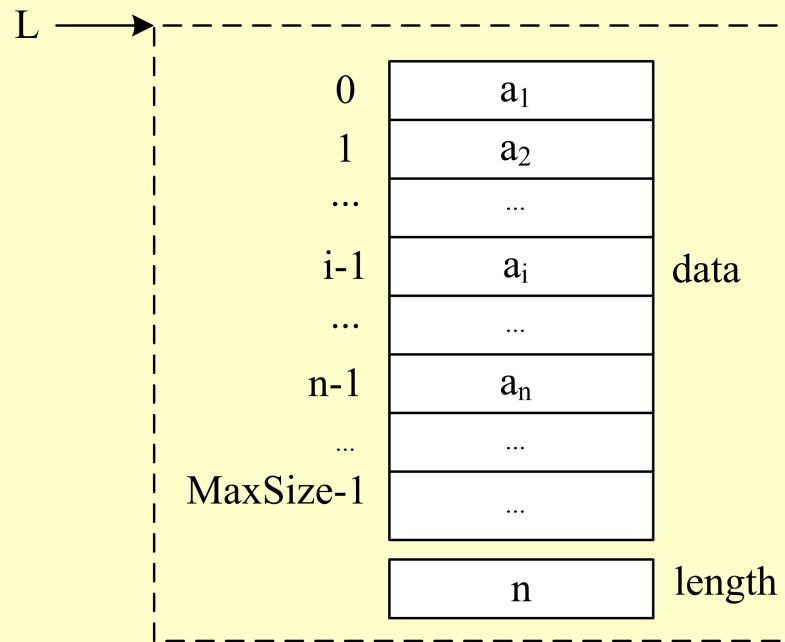




本节主题:

线性表顺序存储的应用

线性表的顺序存储



```
typedef struct
{
    ElemType data[MaxSize];
    int length;
} SqList;
```

例：删除元素

问题

已知长度为 n 的线性表 A 采用顺序存储结构，设计算法，删除线性表中所有值为 x 的数据元素。

要求：时间复杂度为 $O(n)$ 、空间复杂度为 $O(1)$ 的算法

解法0：用基本运算实现

```
void delnode1(SqList *&L, ElemType x)
```

```
{
```

```
    int i;
```

```
    ElemType e;
```

```
    while((i=LocateElem(L,x))>0)
```

```
    {
```

```
        ListDelete(L, i, e);
```

```
    }
```

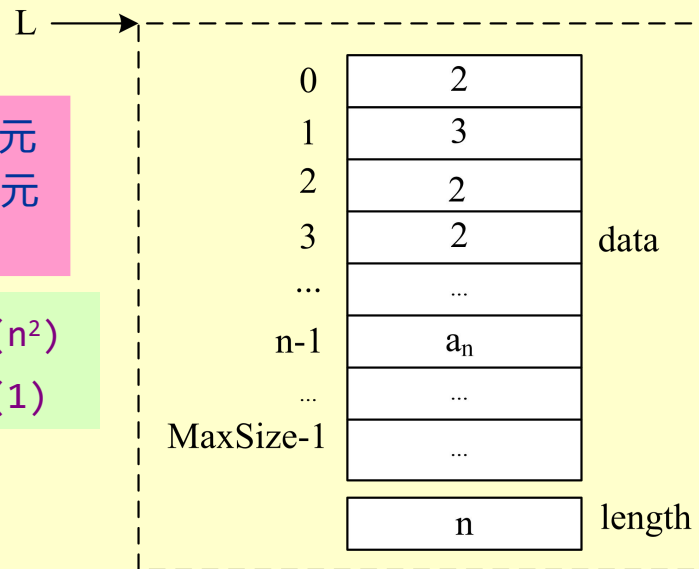
```
}
```

查找第1个值域与 x 相等的元素的逻辑位序。若这样的元素不存在，则返回值为0


时间复杂度为 $O(n^2)$

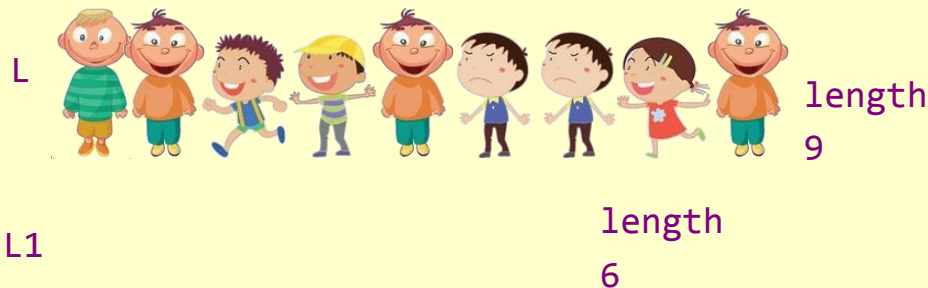
空间复杂度为 $O(1)$

//删除顺序表 L 的第 i 个元素



解法：复制要保留的元素

delnode(L, )



❏ 算法思想

❏ 逐个复制要保留的元素

❏ 要点：L1和L共用空间，即不需要额外空间

❏ 算法描述

```
void delnode1(SqList *&L, ElemType x)
{
    int k=0, i; //k记录非x的元素个数
    for (i=0; i<L->length; i++)
        if (L->data[i]!=x)
        {
            L->data[k]=L->data[i];
            k++;
        }
    L->length=k;
}
```

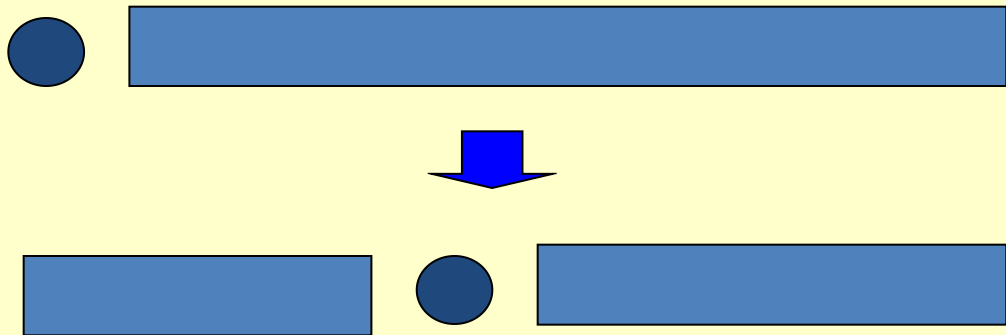
时间复杂度为 $O(n)$

空间复杂度为 $O(1)$

例：分离元素

问题

- 设顺序表有10个元素，其元素类型为整型。
- 设计一个算法，以第一个元素为分界线，将所有小于它的元素移到该元素的前面，将所有大于它的元素移到该元素的后面



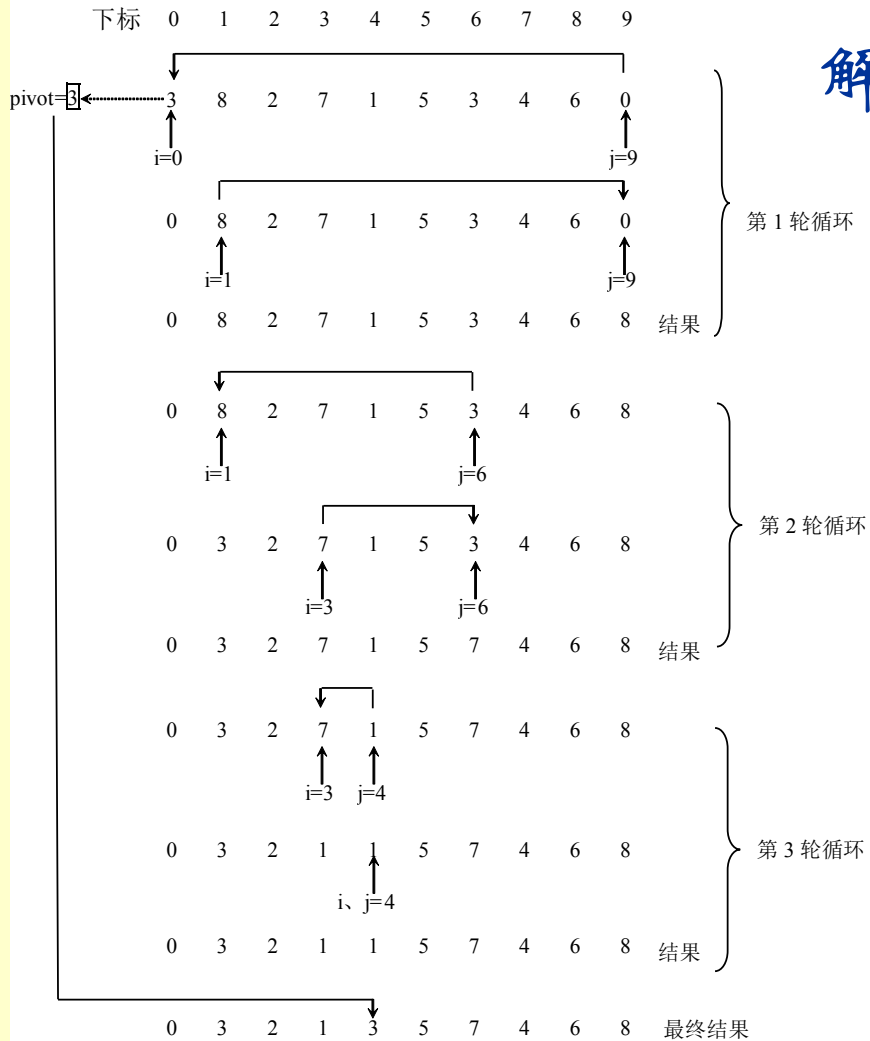
解法1

- 以data[0]为基准
- 从区间两端交替向中间扫描,直至i=j为止,每轮循环
 - 从右向左扫描,找一个小于等于pivot的元素
 - 从左向右扫描,找一个大于pivot的元素
 - 然后将L->data[i]和L->data[j]交换
- 退出循环后,将L->data[0]和L->data[j]进行交换

下标	0	1	2	3	4	5	6	7	8	9	
pivot	← 3	8	2	7	1	5	3	4	6	0	
		↑ i=1								↑ j=9	
	3	0	2	7	1	5	3	4	6	8	交换
	3	0	2	7	1	5	3	4	6	8	
			↑ i=3				↑ j=6				
	3	0	2	3	1	5	7	4	6	8	交换
	3	0	2	3	1	5	7	4	6	8	
				↑ i, j=4							
	3	0	2	3	1	5	7	4	6	8	不交换
	1	0	2	3	3	5	7	4	6	8	循环结束, data[0]与 data[j]交换

```
void move1(SqList *&L)
{
    int i=0,j=L->length-1;
    ElemType pivot=L->data[0];
    ElemType tmp;
    while (i<j)
    {
        while (i<j && L->data[j]>pivot)
            j--;
        while (i<j && L->data[i]<=pivot)
            i++;
        if (i<j)
        {
            tmp=L->data[i];
            L->data[i]=L->data[j];
            L->data[j]=tmp;
        }
    }
    tmp=L->data[0];
    L->data[0]=L->data[j];
    L->data[j]=tmp;
    printf("i=%d\n",i);
}
```

解法2



```
void move2(SqlList *&L)
{
    int i=0,j=L->length-1;
    ElemType pivot=L->data[0];
    while (i<j)
    {
        while (j>i && L->data[j]>pivot)
            j--;
        L->data[i]=L->data[j];
        i++;
        while (i<j && L->data[i]<=pivot)
            i++;
        L->data[j]=L->data[i];
        j--;
    }
    L->data[i]=pivot;
    printf("i=%d\n",i);
}
```

思考题

📁 本题是否有其他改法？