第3周小结

知识点：

- 双链表
- 循环链表
- 有序表

1 双链表

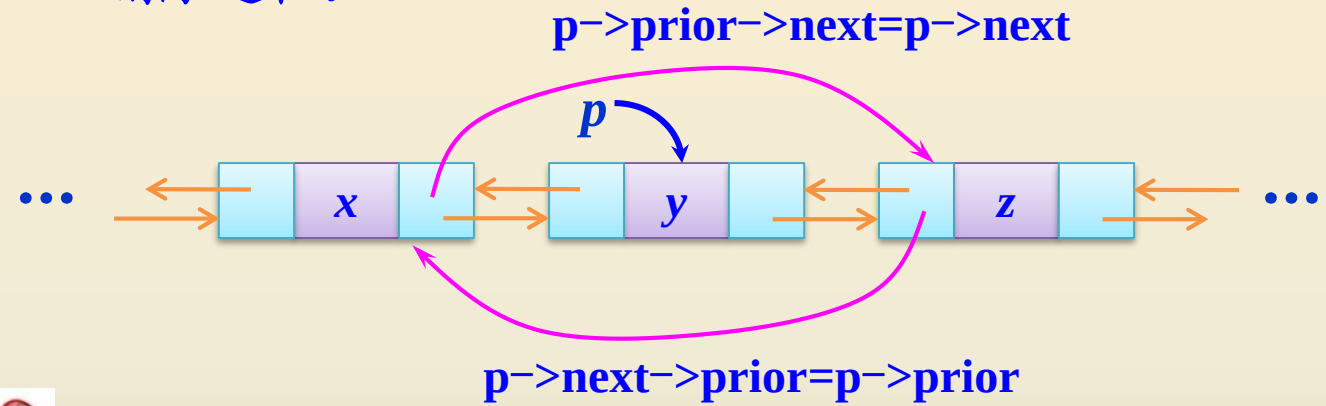
❶ 每个节点有指向前、后相邻节点的指针域。

通常，存储密度低于单链表

❷ 特点：方便查找一个节点的前、后相邻节点。

- 已知某个节点的地址，删除它的时间为O(1)。

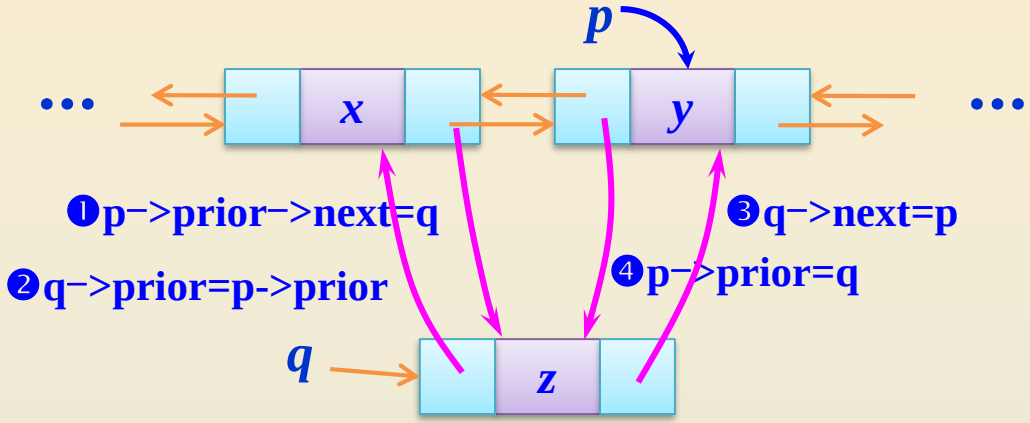
删除过程：



修改p节点前驱节点的next指针和p节点后继节点的prior指针。

- **在某个节点的前、后插入一个节点的时间为O(1)。**

前面插入过程：



❶❷两步必须在❸❹之前完成

2 循 环 链 表

❶ 循环单链表：构成一个环。

可以循环查找

❷ 循环双链表：构成两个环。

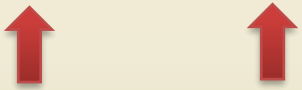
- 可以循环查找
- 可以通过头节点快速找到尾节点

删除尾节点、在尾节点前后插入一个节点的时间均为O(1)。

3 有序表

❶ 从逻辑结构看，有序表是线性表的一个子集。

可以采用顺序表或者链表存储



有序
顺序表

有序
链表

❷ 利用有序表的有序特性可以提高相关算法的效率

假设一个有序表采用顺序表存储。设计一个高效算法删除重复的元素。

例如：L=(1，1，1，2，2，3)

本算法

L=(1，2，3)

解：利用前面介绍过的删除所有值为x的元素的算法思路：

```
void deldupnode1(SqList *&L)
{     int k=1,i;                    //k记录保留的元素个数
      for  (i=1;i<L->length;i++)
           if (L->data[i]!=L->data[i-1])
           {
                 L->data[k]=L->data[i];
                 k++;              //保留的元素增1
           }
       L->length=k;                //顺序表L的长度等于k
}
```

重建顺序表L

❸ 利用二路归并思路可以提高相关算法的效率

假设两个递增有序表采用单链表ha和hc存储（假设同一个单链表中不存在重复的元素）。设计一个高效算法求它们的公共元素，将结果存放在单链表hc中。

例如：ha=(1，2，3)，hb=(2，3，4)

↓ 本算法

hc=(2，3)

算法如下：

```
void InterSect(LinkList *ha，LinkList *hb，LinkList *&hc)
{   LinkList *pa=ha->next，*pb=pb->next，*s，*r;
    hc=(LinkNode *)malloc(sizeof(LinkNode));
    r=hc;                                          //r指向尾节点
    while (pa!=NULL && pb!=NULL)
    {
        if (pa->data<pb->data) pa=pa->next;
        if (pa->data>pb->data) pb=pb->next;
        if (pa->data==pb->data)                    //相同元素
        {  s=(LinkNode *)malloc(sizeof(LinkNode));  //复制节点
           s->data=pa->data;
           r->next=s;  r=s;
           pa=pa->next; pb=pb->next;
        }
    }
    r->next=NULL;
}
```

二路归并+
尾插法建表