第二章 线性表

2.1 线性表的类型定义
2.2 线性表的顺序表示和实现
2.3 线性表的链式表示和实现
2.4 一元多项式的表示和实现

上节复习

链表是线性表的链式存储表示，逻辑上相邻的元素不一定在存储位置上相连

链表由结点组成，每个结点包含数据域Data和指针Next

带头结点的单链表

单链表空：head->next == NULL；单链表末尾p->next=NULL

链表的运算

单链表的查找要从头结点开始往后搜索，时间复杂度为O(n)

单链表的插入和删除的时间复杂度都为O(n)

插入修改指针：P_{i-1}->next=S； S->next=P_i;

删除修改指针：P_{i-1}->next=P_i->next；free(P_i)；

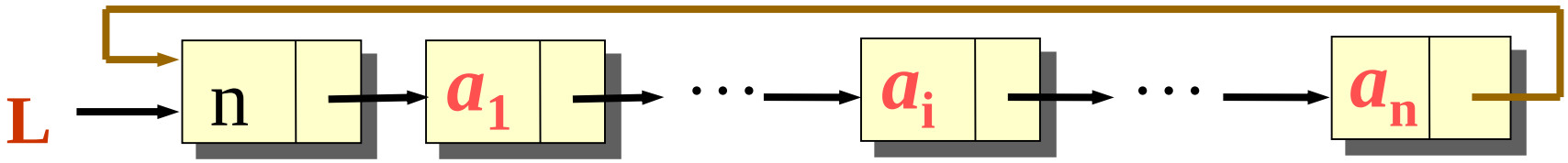
2.3 链表

五. 循环链表

- 循环链表是一种特殊的线性链表
- 循环链表中最后一个结点的指针域指向头结点，整个链表形成一个环
- 循环链表的查找、插入、删除和单链表基本一致
- 与单链表的区别

  (1) 空链表：L → next== L ;

  (2) 表尾结点：p → next== L ;



2.3 链表

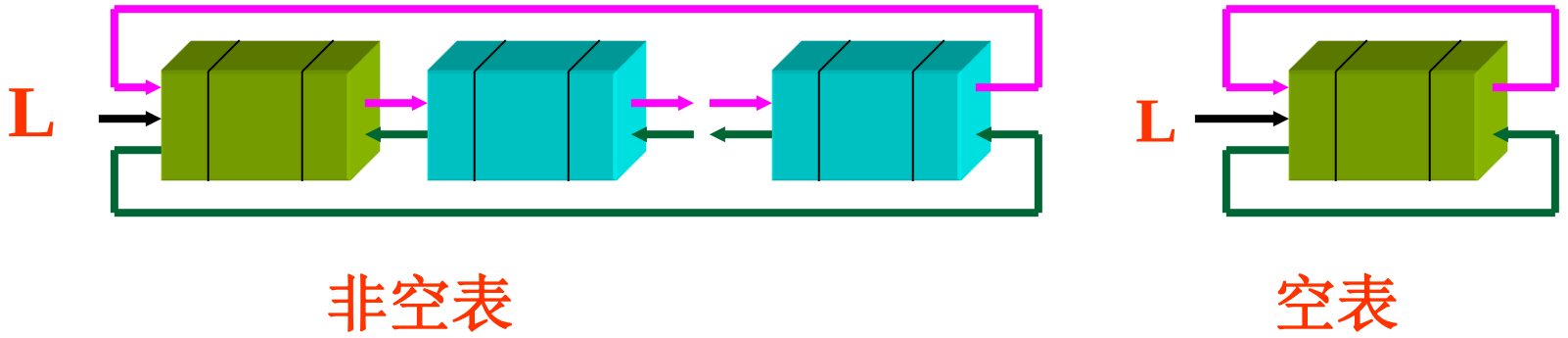
六. 双向链表

- 双向链表也是一种特殊的线性链表,双向链表中每个结点有两个指针，一个指针指向直接后继(next)，另一个指向直接前驱(prior)

2.3 链表

六. 双向链表

- 双向链表中存在两个环(一个是直接后继环，另一个是直接前驱环)



非空表　　空表

(1) 表尾结点：p→ next= L ;

(2) 空链表：L→prior=L ， L→next=L;

2.3 链表

六. 双向链表

◆ 双向链表的定义

```
//定义一个双向链表的结点
Typedef struct DuLNode {
        ElemType            data;
        struct DuLNode      *prior;
        struct DuLNode      *next;
}DuLNode, *DuLinkList;
```

对于任何一个中间结点有：

```
p = p->next->prior
p = p->prior->next
```

2.3 链表

六. 双向链表

◆ 双向链表获取第i个位置的元素

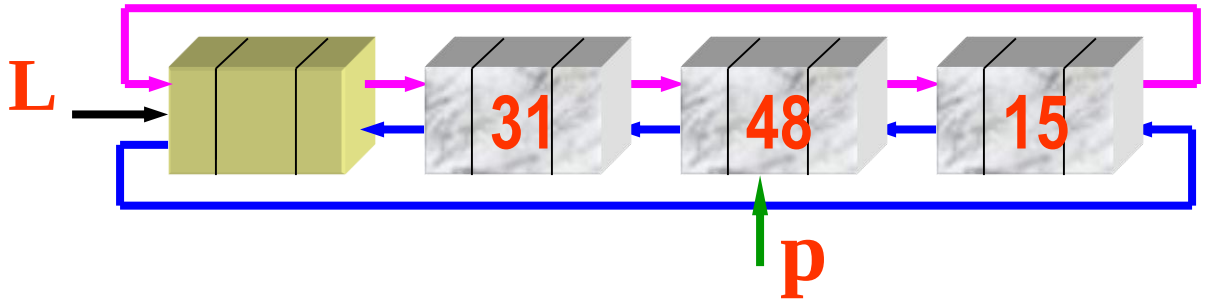
```
//L为带头结点的单链表头指针，当第i个元素存在时，其值赋给e并返回OK，否则返回ERROR
DuLinkList GetElemP_DuL(DuLinkList DL, int i) {
        DuLinkList p;
         p = DL->next;
         int j = 1;  // 初始化，p指向第一个结点，j为计数器
         while ( p!=DL && j<i ) {
                  //顺指针向后查找，直到p指向第i个元素或p指向表头
                   p = p->next;
                   ++j;
         }
         if ( p==DL && j<i )  return NULL;  // 第i个元素不存在
         else return p;
} // GetElem_L
```

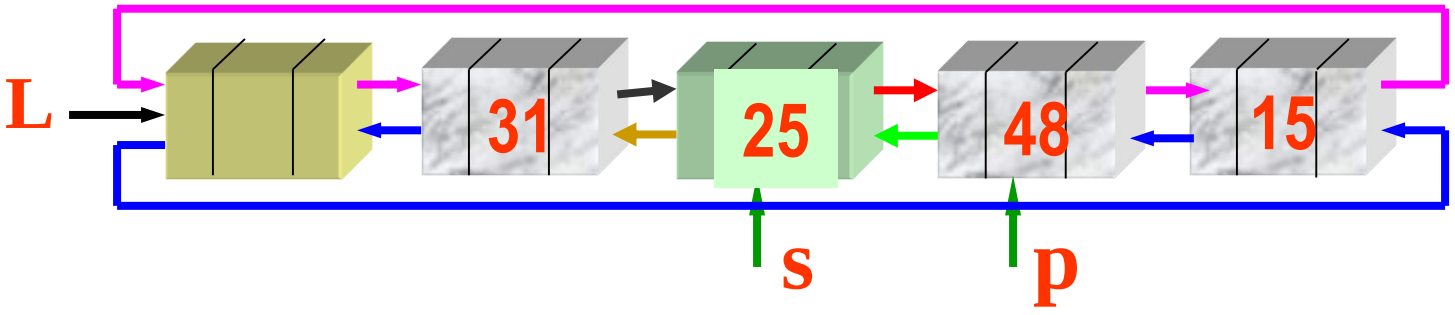
2.3 链表

六. 双向链表

◆ 双向链表的插入需要改变两个方向的指针

```
s->next = p;
s->prior = p->prior;
p->prior->next = s;
p->prior = s;
```





2.3 链表

六. 双向链表

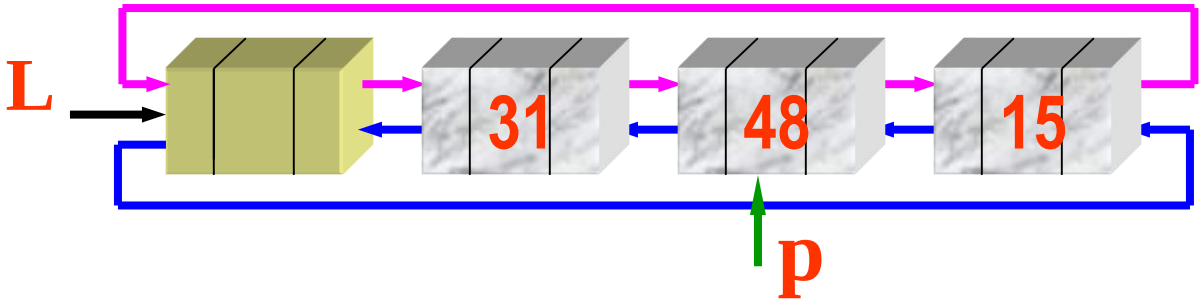
◆ 双向链表的插入

```
Status ListInsert_DuL(DuLinkList &L, int i, ElemType e) { //算法2.18
 // 在带头结点的双链循环线性表L的第i个元素之前插入元素e，
        DuLinkList p,s;
        if (!(p = GetElemP_DuL(L, i)))  // 在L中确定第i个元素的位置指针p
          return ERROR;                 // 第i个元素不存在
        if (!(s = (DuLinkList)malloc(sizeof(DuLNode))))
          return ERROR;
        s->data = e;
        s->prior = p->prior;
        p->prior->next = s;
        s->next = p;
        p->prior = s;
        return OK;
} // ListInsert_DuL
```

2.3 链表

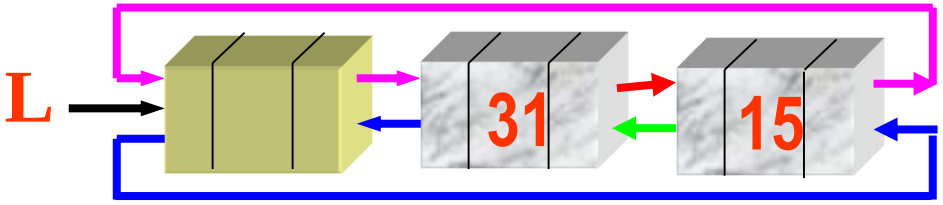
六. 双向链表

◆ 双向链表的删除需要改变两个方向的指针

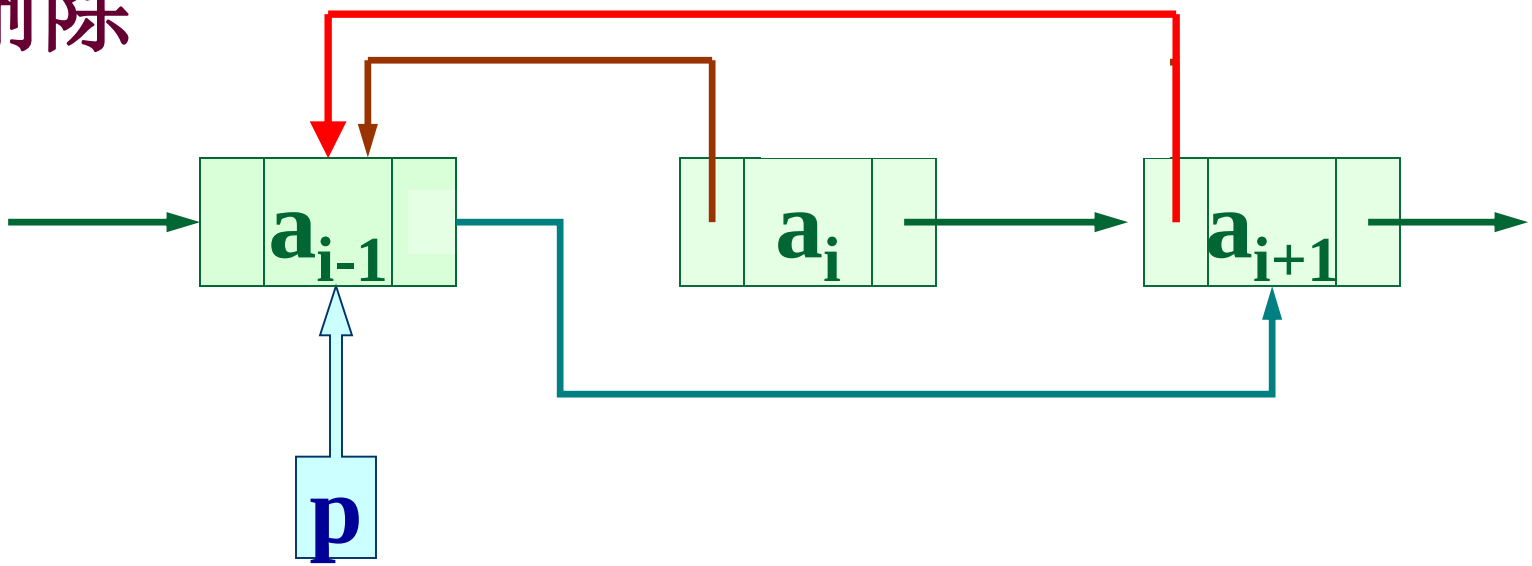


p->prior->next = p->next;

p->next->prior = p->prior;



删除



p->next = p->next->next;

p->next->prior = p;

2.3 链表

六. 双向链表

◆ 双向链表的删除

```
Status ListDelete_DuL(DuLinkList &L, int i, ElemType &e) {//算法2.19
  // 删除带头结点的双链循环线性表L的第i个元素
  DuLinkList p;
  if (!(p = GetElemP_DuL(L, i)))  // 在L中确定第i个元素的位置指针p
    return ERROR;                 // 即第i个元素不存在
  e = p->data;
  p->prior->next = p->next;
  p->next->prior = p->prior;
  free(p);
  return OK;
} // ListDelete_DuL
```

2.3 链表

七. 顺序表与链表的比较（空间）

◆ 存储分配的方式

顺序表的存储空间是静态分配的

链表的存储空间是动态分配的

存储密度 = 结点数据本身所占的存储量/结点结构所占的存储总量

顺序表的存储密度 = 1

链表的存储密度 < 1（要包含指针域）

2.3 链表

七. 顺序表与链表的比较（时间）

- 存取方式

  顺序表可以随机存取，也可以顺序存取

  链表必须顺序存取

- 插入/删除时移动元素个数

  顺序表平均需要移动近一半元素

  链表不需要移动元素，只需要修改指针

2.3 链表

七. 顺序表与链表的比较（应用）

- 如果线性表主要是存储大量的数据，并主要用于查找时，采用顺序表较好，如数据库
- 如果线性表存储的数据元素经常需要做插入与删除操作，则采用链表较好，如操作系统中进程控制块(PCB)的管理，内存空间的管理等

2.3 链表

八. 一元多项式的表示与运算

- 一元多项式： $f(x)=a_0+a_1x+\cdots+a_{n-1}x^{n-1}+a_nx^n$
  - 由n+1个系数唯一确定，在计算机中可用线性表(p_0，p_1，p_2，…，p_n)表示。
  - 主要运算：多项式相加、相减、相乘等

不失一般性，设有两个一元多项式：

$P(x) = p_0+p_1x+p_2x^2+ \ldots +p_nx^n$，

$Q(x) = q_0+q_1x+q_2x^2+ \ldots +q_mx^m \quad (m<n)$

$R(x) = P(x) + Q(x)$

R(x)由线性表R((p_0+q_0)，(p_1+q_1)，(p_2+q_2)，…，(p_m+q_m)，…，p_n)唯一表示。

2.3 链表

八．一元多项式的表示与运算

【分析】多项式的关键数据是：多项式项数**n** 、每一项的系数a_i（及相应指数 **i**）。有**3**种不同的表示方法。

方法1：采用顺序存储结构直接表示

例如：　$f(x)=4x^5-3x^2+1$

表示成：

下标 i	0	1	2	3	4	5	……
a[i]	1	0	–3	0	0	4	……

2.3 链表

八. 一元多项式的表示与运算

◆ 顺序存储表示的相加

- 用顺序表示的相加非常简单。访问第5项可直接访问：**L.a[4].coef，L.a[4].expn**
- 两个多项式相加就是在两个顺序表中寻找指数**expn**相同的元素把两者的系数**coef**相加
- 例如 **$f(x)=5+x+2x^2+3x^3$**， **$p(x)= -5-x+6x^2+4x^4$**，两者相加 **$R(x)=f(x)+p(x)$**

下标	0	1	2	3	4
f(x)	5	1	2	3	0
p(x)	-5	-1	6	0	4
R(x)	0	0	8	3	4

系数为0的部分不显示， 最终结果： **$R(x)=8x^2+3x^3+4x^4$**

方法2：采用顺序存储结构表示多项式的非零项。

每个非零项 $a_i x^i$ 涉及两个信息：指数 i 和系数 a_i ，可以将一个多项式看成是一个 (a_i, i) 二元组的集合。

```
typedef struct
    {
        float  coef;   /*系数部分*/
        int    expn;   /*指数部分*/
    } ElemType ;
```

方法2：采用顺序存储结构表示多项式的非零项。

例如：$P_1(x)=9x^{12}+15x^8+3x^2$ 和 $P_2(x)=26x^{19}-4x^8-13x^6+82$

❖ 相加过程：

- 比较（9, 12）和（26，19），将（26, 19）移到结果多项式；
- 继续比较（9, 12）和（–4，8），将（9, 12）移到结果多项式；
- 比较（15, 8）和（–4，8），15+（–4）=11，不为0，将新的一项（11,8）增加到结果多项式；
- 比较（3,2）和（–13,6）, 将（–13,6）移到结果多项式；
- 比较（3,2）和（82,0）, 将（3,2）移到结果多项式；
- 将（82,0）直接移到结果多项式。
- 最后得到的结果多项式是：((26,19), (9,12), (11,8), (–13,6), (3,2), (82,0))
- $$P_2(x)=26x^{19}+9x^{12}+11x^8-13x^6+3x^2+82$$

下标i	0	1	2
系数	9	15	3
指数	12	8	2

$P_1(x)$

下标i	0	1	2	3
系数	26	–4	–13	82
指数	19	8	6	0

$P_2(x)$

方法3：采用链表结构来存储多项式的非零项。

每个链表结点存储多项式中的一个非零项，包括系数和指数两个数据域以及一个指针域，表示为：

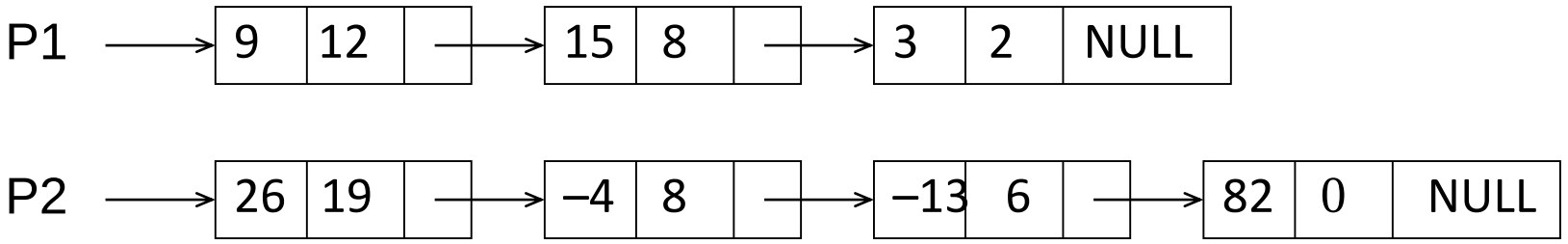

```
typedef struct PolyNode *Polynomial;
typedef struct PolyNode {
        int coef;
        int expon;
        Polynomial link;
}
```

例如：

$$P_1(x) = 9x^{12} + 15x^8 + 3x^2$$

$$P_2(x) = 26x^{19} - 4x^8 - 13x^6 + 82$$

链表存储形式为：



2.3 链表

八．一元多项式的表示与运算

- 链式存储表示的相加
  - 当采用链式存储表示时，根据结点类型定义，凡是系数为0的项不在链表中出现，从而可以大大减少链表的长度。
  - 相加的实质是：
  - ✓ 指数不同： 是链表的合并
  - ✓ 指数相同： 系数相加，和为0，去掉结点；和不为0，修改结点的系数域
- 程序实现的操作包括：
  - 多项式链表创建、相加、输出
  - 项插入、删除、查找

2.3 链表

八．一元多项式的表示与运算

多项式链表相加的实现

```
Ploy  add_ploy(ploy  *La，  ploy  *Lb)  //La，Lb为头指针，结果保存在La中
{
        Lc=pc=La ;   pa=La->next ;     pb=Lb->next ;
         while (  pa!=NULL && pb!=NULL )
         {
              if  (  pa->expn  <  pb->expn)
              {  pc->next = pa ; pc = pa ;  pa = pa->next ;    }
              else if  (  pa->expn  >  pb->expn)
              {  pc->next=pb ; pc=pb ;  pb=pb->next ;    }
```

(未完见下页)

2.3 链表

八．一元多项式的表示与运算

◆ 多项式链表相加的实现（续）

```
        else
        {   x = pa->coef + pb->coef ;
            if  ( abs(x)<=1.0e-6 )   //如果系数和为0，删除两个结点
            {  ptr = pa ; pa = pa->next ;  free(ptr) ;
               ptr = pb ; pb = pb->next ;  free(ptr) ;
            }
            else     //  如果系数和不为0，修改其中一个结点的系数域，删除另一个结点
            {  pc->next = pa ; pa->coef = x ;
              pc=pa ; pa=pa->next ;
                        ptr=pb ; pb=pb->next ; free(pb) ;
            }
        }//end else
  } // end while
  if  (pa==NULL)  pc->next = pb ;
  else  pc->next = pa ;
  return (Lc) ;
}
```

第2章总结

- 线性表是n个数据元素的有限序列，数据同一性、数据顺序性
- 顺序表是用一组地址连续的存储单元依次存储线性表的数据元素
  - 采用一维数组表示顺序表
  - 顺序表的创建需要为数组分配空间
  - 顺序表的数据结构包括*elem、length、listsize
    - 顺序表的插入：n-i+1个元素往后移动，时间复杂度O(*n*)
    - 顺序表的删除：n-i个元素往前移动，时间复杂度O(*n*)
- 链表是线性表的链式存储表示，逻辑上相邻的元素不一定在存储位置上相连
  - 链表由结点组成，每个结点包含数据域Data和指针Next
  - 带头结点的单链表
    - 链表的查找只有从头结点开始，顺链一步步查找，时间复杂度O(*n*)
    - 链表插入和删除，对指针的修改，时间复杂度O(*n*)
- 静态链表、循环链表、双向链表

练习

习题1. 已知非空循环链表，设h是指向头结点的指针，p是辅助指针。执行以下程序段的作用是什么？

```
p=h;
while ( p->next->next != h )
    p=p->next;
p->next=h;
```

习题2. 双向链表中前驱指针为prior，后继指针为next，在指针P所指结点前插入指针S所指的结点，请写出要执行的四行语句？