4.4 高级链表 Advanced Linked Lists

郝家胜

hao@uestc.edu.cn

自动化工程学院

内容回顾

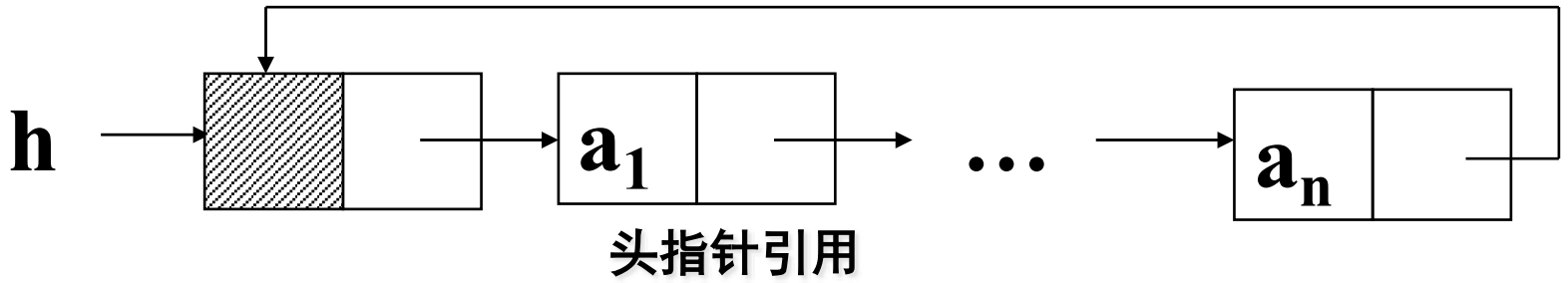
- 线性表的概念
- 线性表的抽象数据类型
- 线性表抽象数据类型的运算示例

内容提要

- 循环链表
  - 循环链表的表示
  - 循环链表的操作
- 双链表
  - 双链表的表示
  - 双链表的操作

循环链表

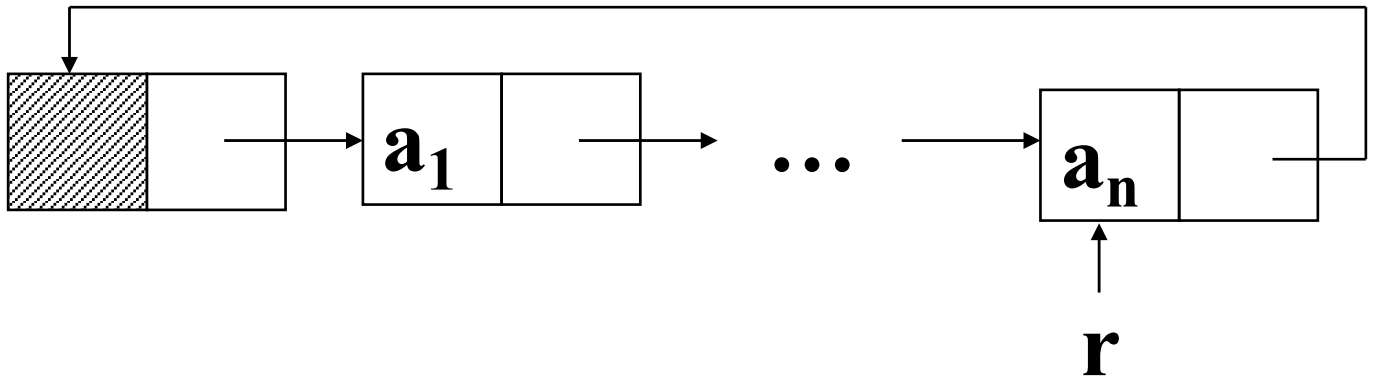
- 一种首尾相接的链表
- 将单链表的最后一个结点的指针域指向头结点（或不带头结点链表的第一个元素结点）



- 不增加存储量
- 从任意结点开始，可以遍历所有结点

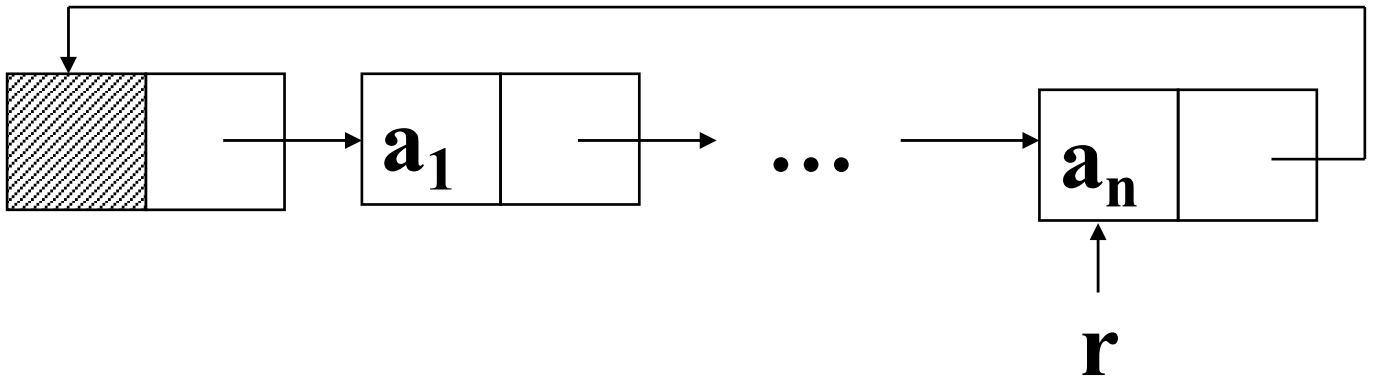
尾指针引用

- 循环单链表常用尾指针r来命名，这样查找链表开始结点a1和终端结点an都很方便，
- 它们的存储位置分别是r->next->next 和 r



循环链表的特征

结点结构：首尾相连　　next(rear(L)) == head(L)

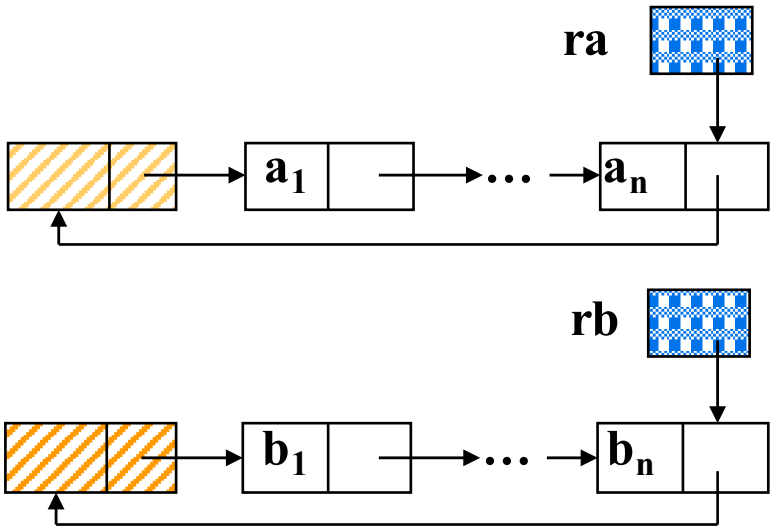


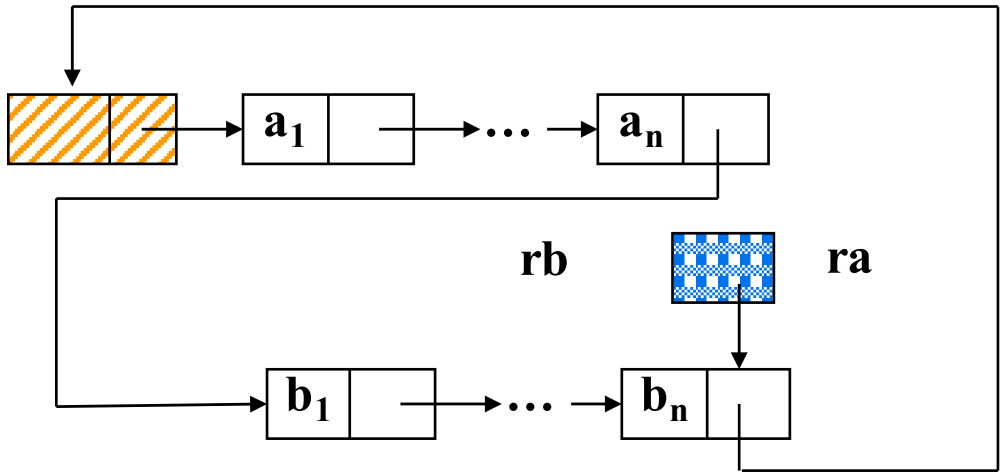
空表判断　　：　next(head(L)) == head(L)
最后一个节点：　next(x) == head(L)
第一个节点：　　prev(x) == head(L)
访问第一个节点：　next(head(L))

循环链表的操作

- 获取链表长度
  - LENGTH(L)
- 获取指定结点
  - GET(L, i)
- 在指定位置插入新结点
  - INSERT(L, i, x)
- 删除指定位置结点
  - DELETE(L, i)

两个循环链表的合并





- 使用头指针表示的单链表和循环单链表来实现
  - 需要遍历第一个链表，找到结点a_n，
  - 再将第二个链表的第一结点b_1链接到a_n后。

使用尾指针表示的循环单链表来实现

- 只需修改一些指针域
- 不需要遍历第一个链表

循环链表合并算法伪代码

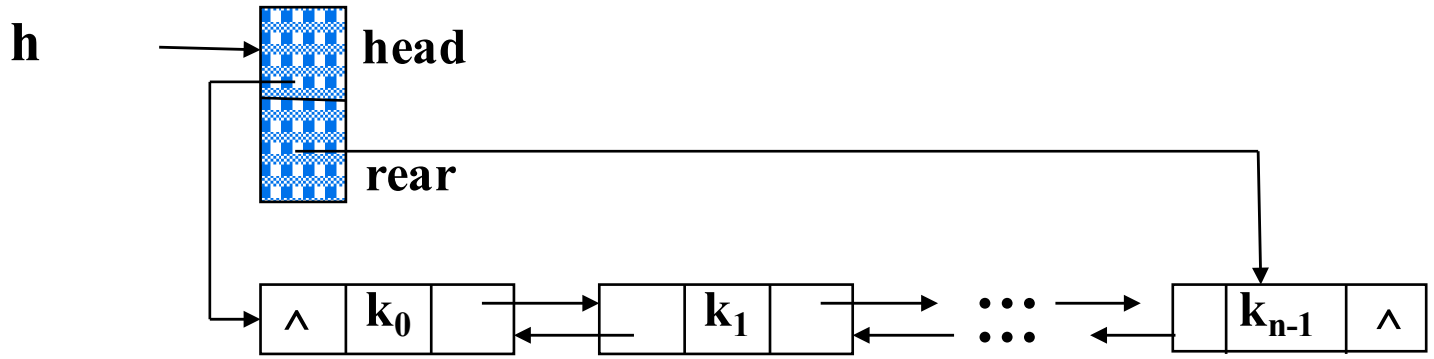
```
ha ← next(rear(a))
hb ← next(rear(b))

next(rear(a))← next(hb)
next(rear(b))← ha
rear(a) ← rear(b)

LENGTH(a)← LENGTH(a) + LENGTH(b)
delete hb
```

双链表

单链表缺点：找后继容易，找前驱必须从头开始查找。

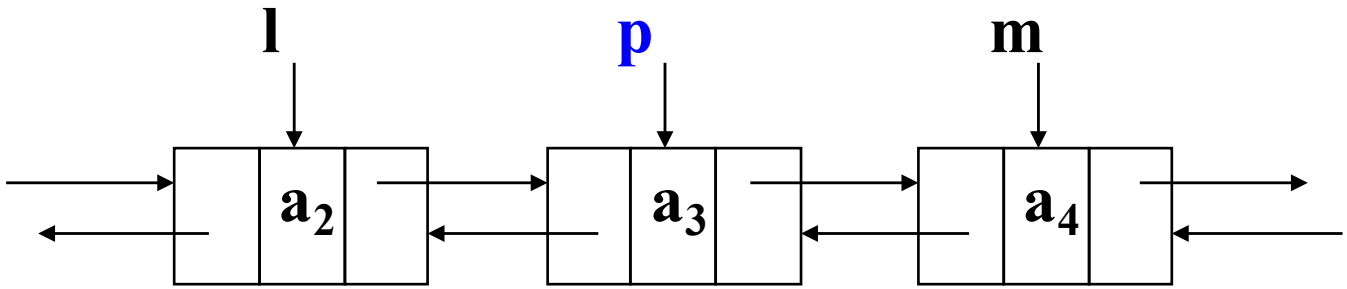
双向链表：　　既可以找前驱，也可以找后继。

结点结构：



双链表

- 双链表中，若p是指向结点的指针

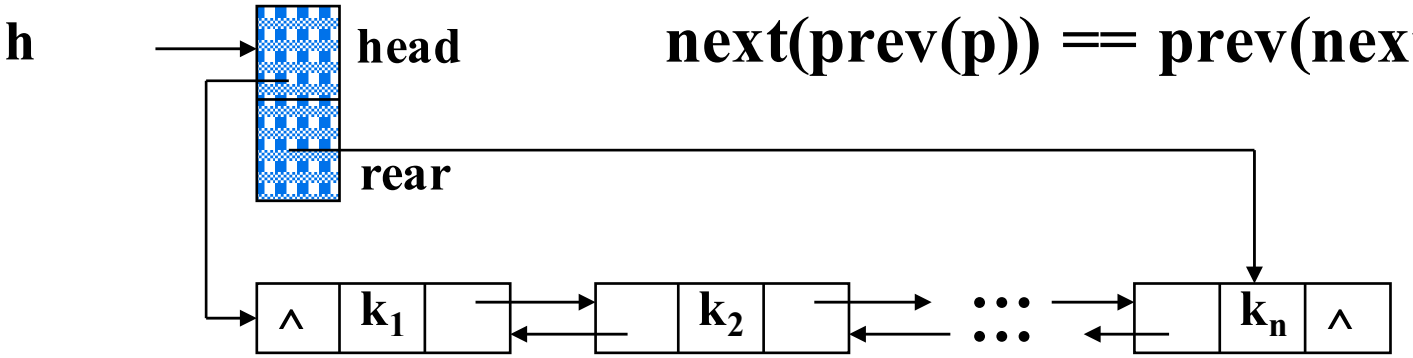


$$next(prev(p)) == p$$

$$prev(next(p)) == p$$

双向链表的特征

结点结构：



next(prev(p)) == prev(next(p)) == p

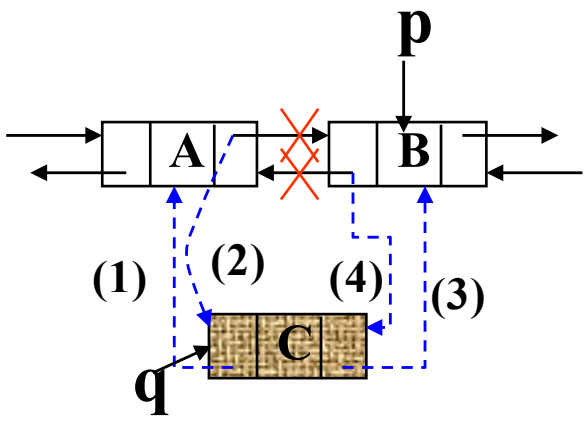
空表判断　　：　next(head(L)) == NULL

最后一个节点：　next(x) == NULL

第一个节点：　prev(x) == head(L)

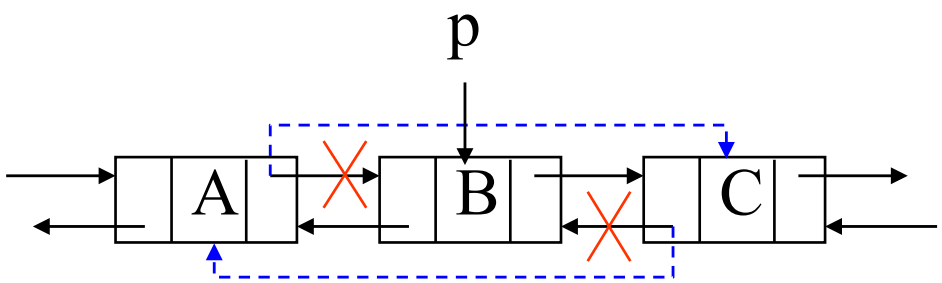
访问第一个节点：　next(head(L))

双向链表结点插入



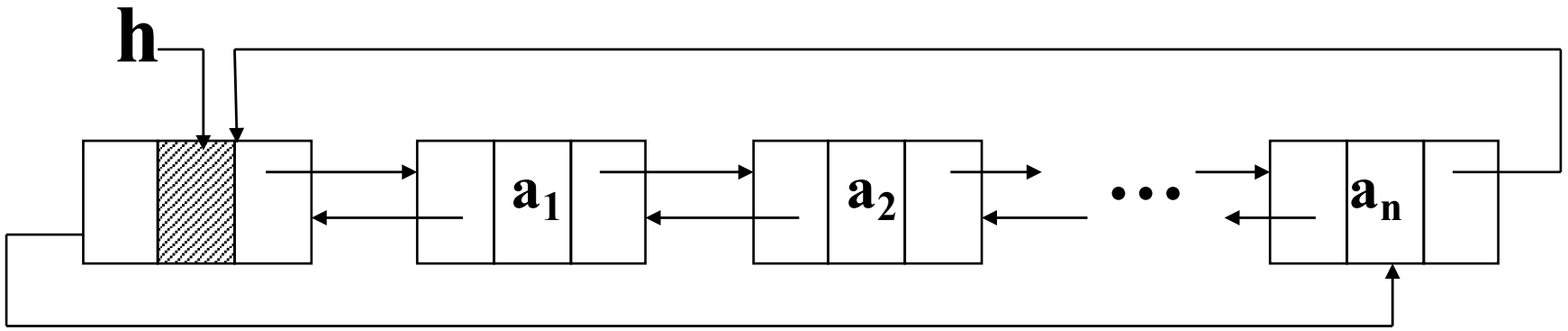

```
(1)  prev(q)  ←  prev(p)
(2)  next(prev(p))←  q
(3)  next(q)  ←  p
(4)  prev(p)←  q
```

双向链表结点删除




```
next(prev(p)) ← next(p)
prev(next(p)) ← prev(p)
delete(p)
```

双向循环链表



空表判断　　：prev(head(L)) == next(head(L)) == head(L)

最后结点判断：p->rlink=pdlist->head 或

p =pdlist->head->llink

第一个结点　：prev(x) == head(L)

最后结点　　：next(x) == head(L)

应用举例（1）： Josephus问题

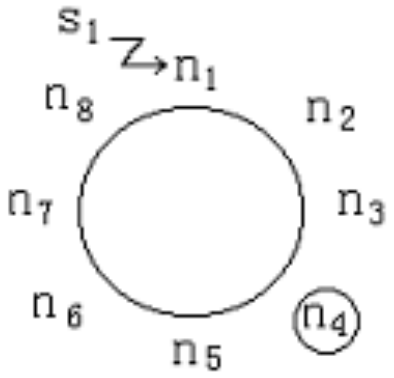
设有n个人围坐在一个圆桌周围，现从第s个人开始报数，数到第m的人出列，然后从出列的下一个人重新开始报数，数到第m的人又出列，…，如此反复直到所有的人全部出列为止。

Josephus问题是：

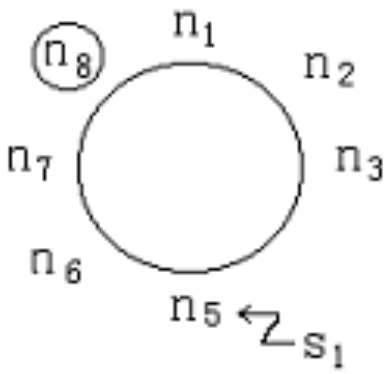
对于任意给定的n,s和m，求出按出列次序得到的n个人员的序列。

Josephus问题

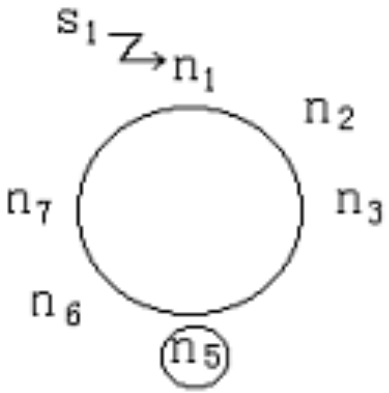
以n=8，s=1，m=4为例



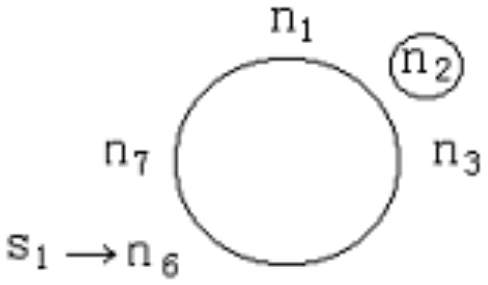
(a) n_4



(b) n_4 n_8



(c) n_4 n_8 n_5



(d) n_4 n_8 n_5 n_2

Josephus问题

- 求解Josephus问题的一般步骤为：

（1）首先利用线性表的一些运算如创建空线性表、插入元素等构造Josephus表；

（2）从Josephus表中的第s个结点开始寻找、输出和删除表中的第m个结点，然后再从该结点后的下一结点开始寻找、输出和删除表中的第m个结点，重复此过程，直到Josephus表中的所有元素都删除。

应用举例（2）：一元多项式表示和运算

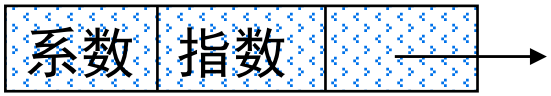
一元多项式：$P_n(x) = p_0+p_1x+p_2x^2+\cdots+p_n\ x^n$

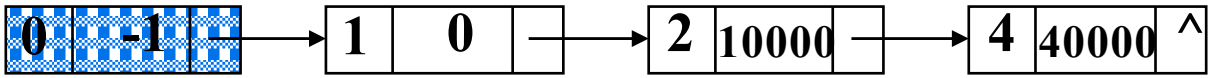
线性表表示：$P=(p_0,\ p_1,\ p_2,\ \cdots,\ p_n)$

顺序表表示：只存系数（第i个元素存x^i的系数）

特殊问题：$p(x) = 1+\ 2x^{10000}+\ 4x^{40000}$

链表表示：　　每个结点结构





一元多项式的求和运算

- 数据：

  $f(x)=6x^3+12x^2+10x+11$

  $f(x)=3x^2+5x+9;$

- 运算：f(x)+g(x)　f(x)-g(x)　f(x)*g(x)　f(x)/g(x)
- 用线性表表示多项式；用节点表示项；用节点之间的关系表示项之间的降幂关系。

线性表小结

- 线性表的逻辑结构
- 线性表的顺序存储结构，要求能够灵活应用
  - 顺序表上的插入、删除操作及其平均时间性能分析
  - 利用顺序表设计算法解决简单的应用问题
- ADT的基本概念及基本实现方法
- 线性表的链式存储结构的单链表
  - 单链表如何表示线性表中元素之间的逻辑关系
  - 单链表中头指针和头结点的使用
  - 单链表上实现的建表、查找、插入和删除等基本算法，并分析其时间复杂度
  - 利用单链表设计算法解决简单的应用问题
- 线性表的链式存储结构的循环链表和双链表