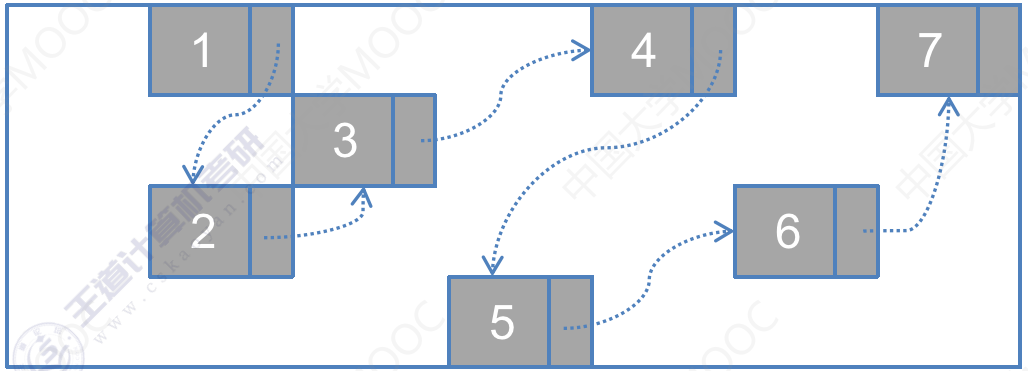
线性表的链式表示

顺序表

- 插入和删除操作移动大量元素。
- 数组的大小不好确定。
- 占用一大段连续的存储空间，造成很多碎片。

单链表

结点结构

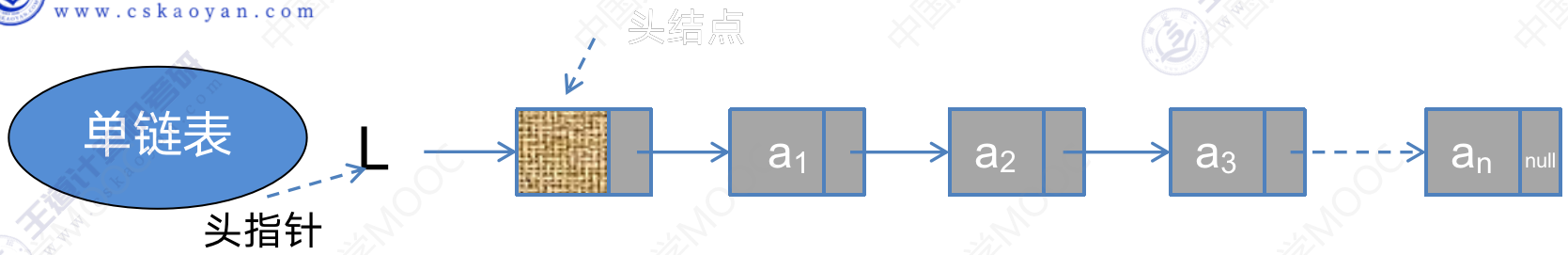


存储块

逻辑上相邻的两个元素在物理位置上不相邻

单链表结点的定义：

```
typedef struct LNode{       //单链表结点类型
    ElemType data;          //数据域
    struct LNode *next;     //指针域
}LNode, *LinkList;
```



头指针：链表中第一个结点的存储位置，用来标识单链表。
头结点：在单链表第一个结点之前附加的一个结点，为了操作上的方便。

若链表有头结点，则头指针永远指向头结点，不论链表是否为空，头指针均不为空，头指针是链表的必须元素，他标识一个链表。
头结点是为了操作的方便而设立的，其数据域一般为空，或者存放链表的长度。有头结点后，对在第一结点前插入和删除第一结点的操作就统一了，不需要频繁重置头指针。但头结点不是必须的。

优缺点

优点

- 插入和删除操作不需要移动元素，只需要修改指针。
- 不需要大量的连续存储空间。

缺点

- 单链表附加指针域，也存在浪费存储空间的缺点。
- 查找操作时需要从表头开始遍历，依次查找，不能随机存取。

插入操作

创建新结点代码：

```
q=(LNode*)malloc(sizeof(LNode))
q->data=x;
```

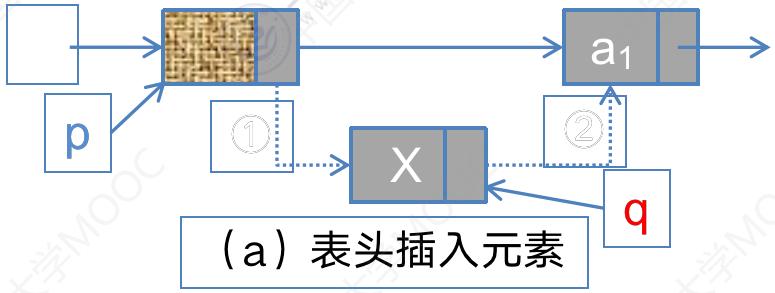
(a) (b)操作的代码：

```
q->next=p->next;
p->next=q;
```

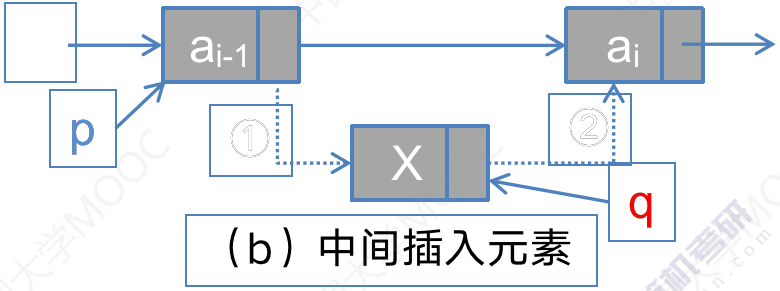
(c)操作的代码：

```
p->next=q;
q->next=NULL;
```

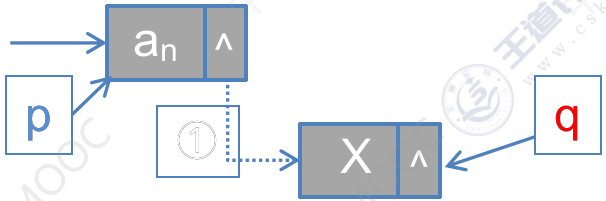
//请分别用头插法和尾插法创建一个单链表



（a）表头插入元素



（b）中间插入元素



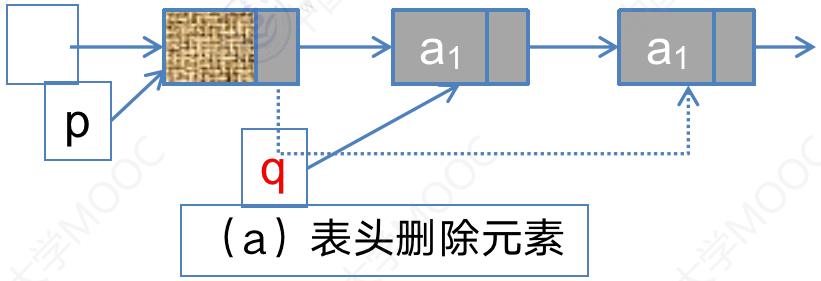
（c）表尾插入元素

删除操作

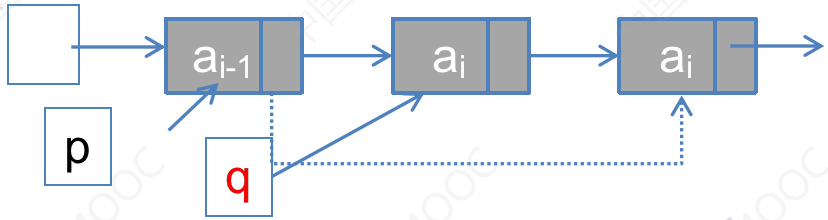
(a)(b) (c)操作的代码：

p=GetElem(L,i-1);//查找删除位置的前驱节点

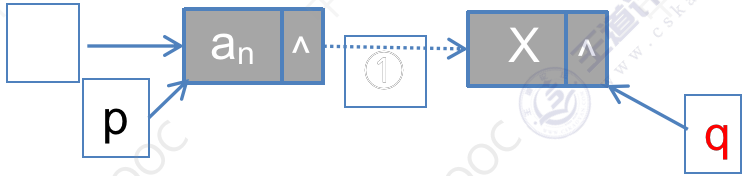
```
q=p->next;
P->next=q->next;
free(q);
```



（a）表头删除元素



（b）中间删除元素

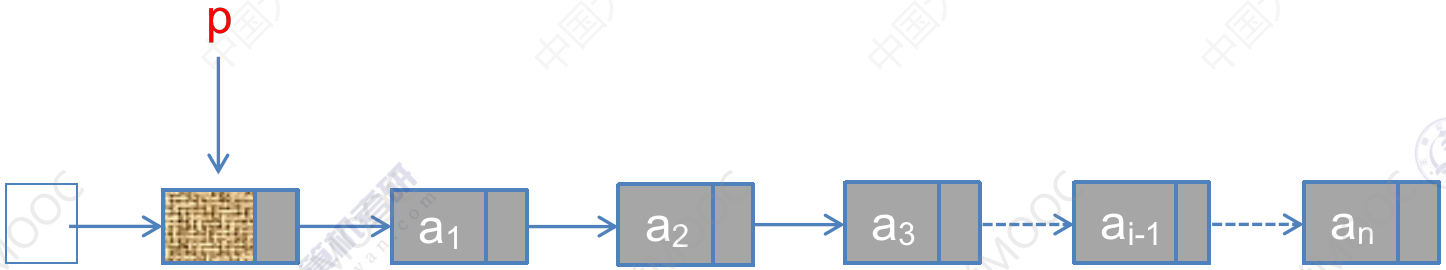


（c）表尾删除元素

查找操作

按序号查找结点值的算法如下：

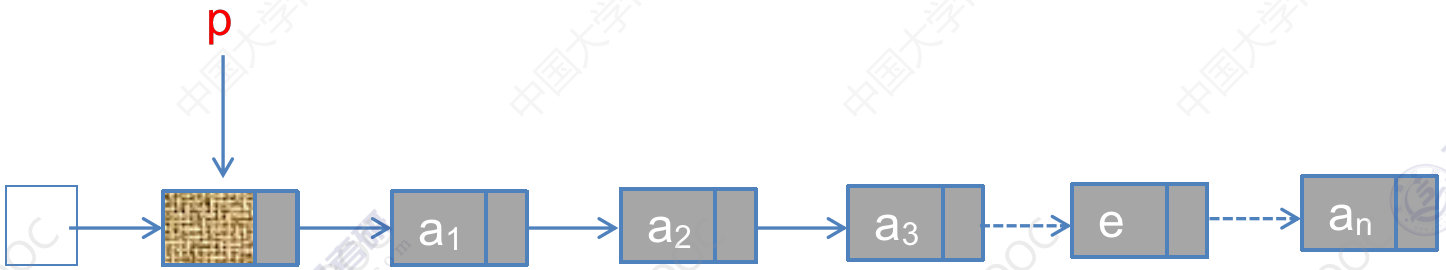
```
int j=1;
while(p&&j<i){
    p=p->next;
    j++;
  }
return p;
```



查找操作

按值查找结点的算法如下：

```
LNode *p=L->next;
while(p!=NULL&&p->data!=e){
      p=p->next;
}
return p;
```

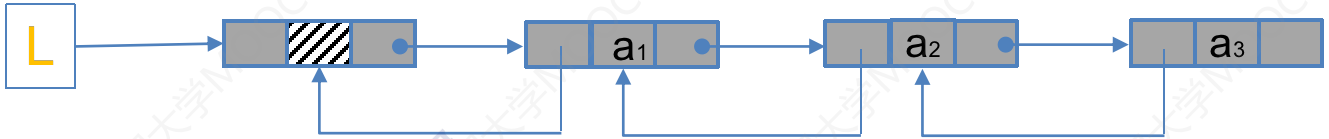


双链表

prior | data | next

结点结构

双链表示意图



逻辑上相邻的两个元素在物理位置上不相邻

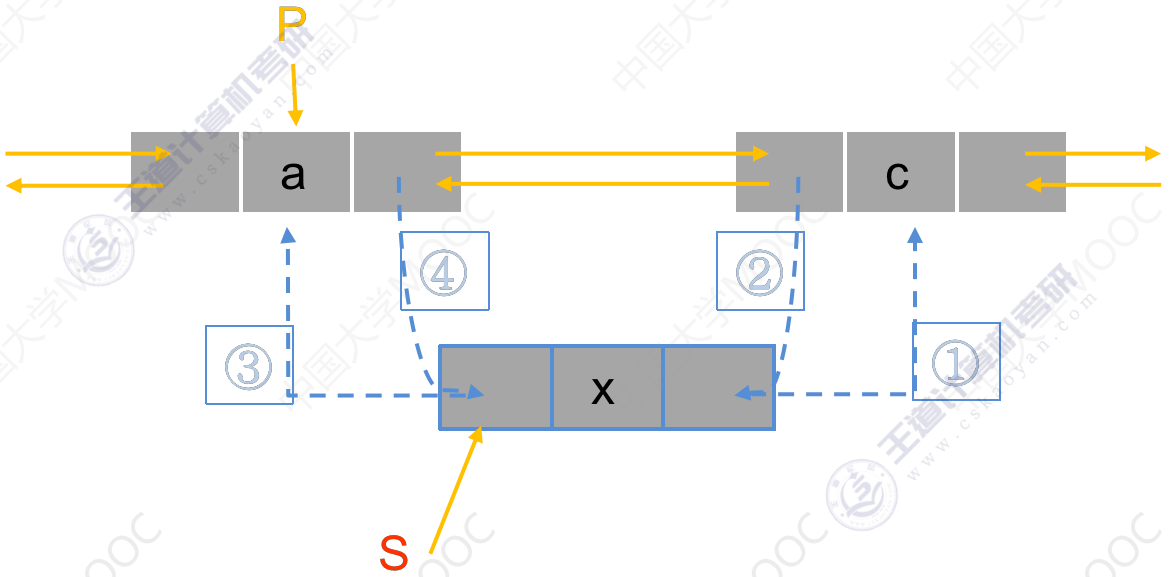
双链表结点的定义：

```
typedef struct DNode{          //双链表结点类型
    ElemType data;             //数据域
    struct DNode *prior;       //前驱指针
    struct DNode *next;        //后继指针
}DNode, *DinkList;
```

双链表的插入操作

在双链表中p所指的结点之后插入结点*s

1、s->next=p->next
2、p->next->prior=s;
3、s->prior=p;
4、p->next=s;



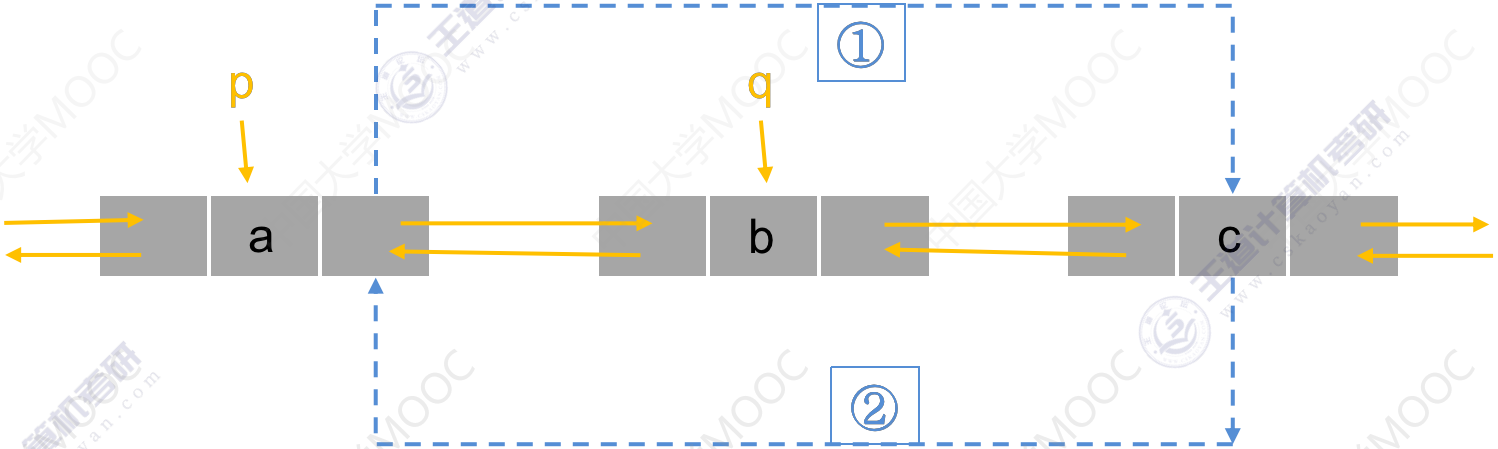
双链表的删除操作

删除双链表中结点*p的后继结点*q

1、p->next=q->next

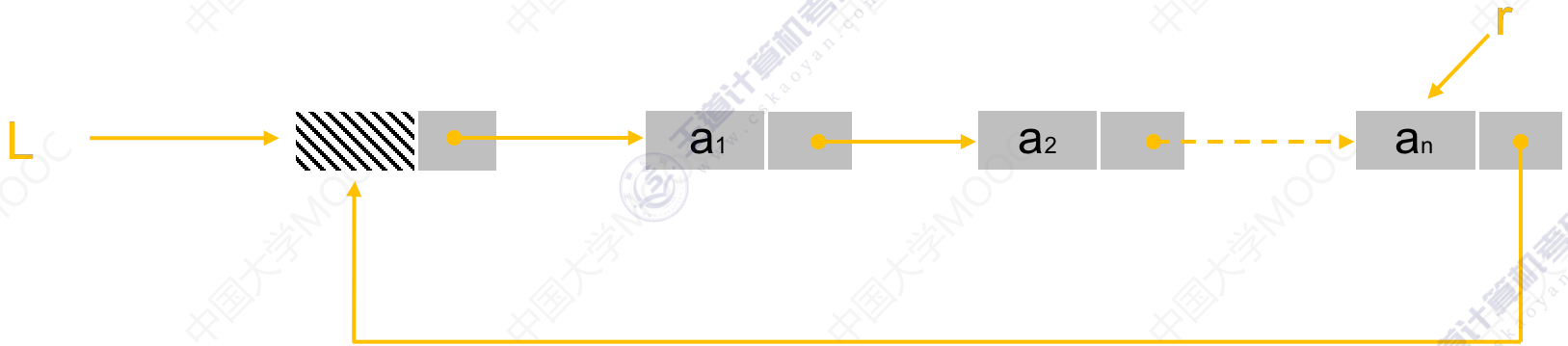
2、q->next->prior=p;

3、free(q);



循环单链表

循环单链表与单链表的区别在于，表中最后一个结点的next指针不是NULL，而是指向头结点L，从而整个链表形成一个环



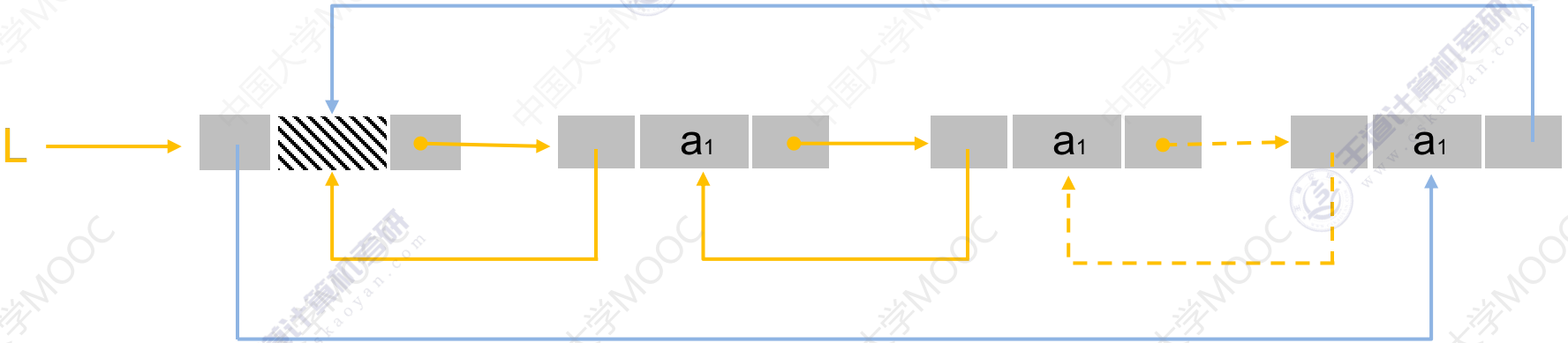
r->next=L;

while(pcur->next!=L)

循环双链表

循环双链表与双链表的区别在于，表中最后一个结点的next指针不是NULL，而是指向头结点L，同时L->prior指向尾节点。

当循环双链表为空时，其头结点的prior域和next域都等于L。



静态链表

静态链表是借助数组来描述线性表的链式存储结构，结构类型如下

```
#define Maxsize 50
typedef struct{
        ElemType data;
        int next;
}SLinkList[Maxsize];
```

0		2
1	b	6
2	a	1
3	d	-1
4		
5		
6	c	3