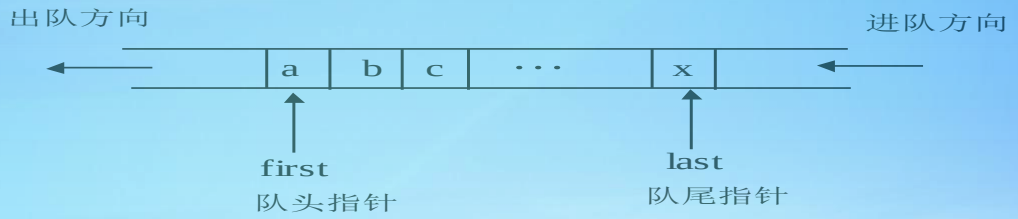
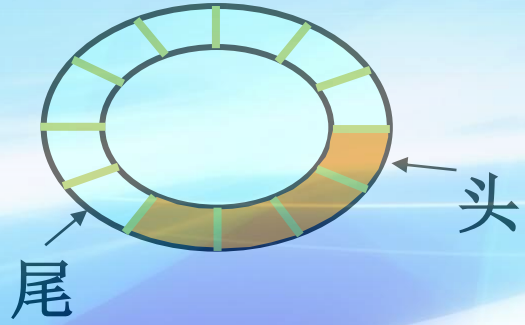
队列

《数据结构》

网络工程教研中心　陈卫卫





学习目标和要求

1.能够准确描述队列的特点；

2.能够写出顺序队的入队和出队算法；

1. 队列的概念

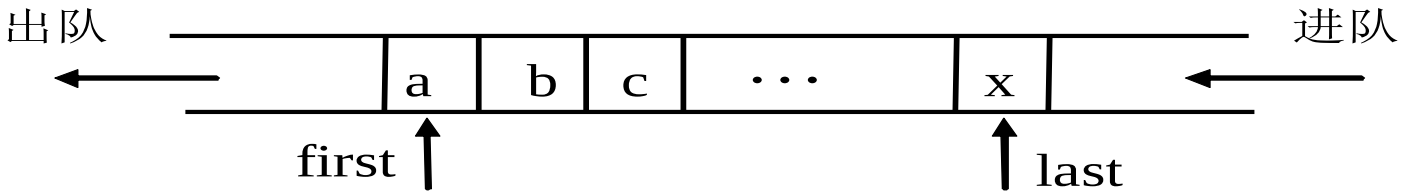
队的术语和图示：

插入端，队尾（rear）

删除端，队头（front）

first和last：分别指向队头元素和队尾元素

进队和出队



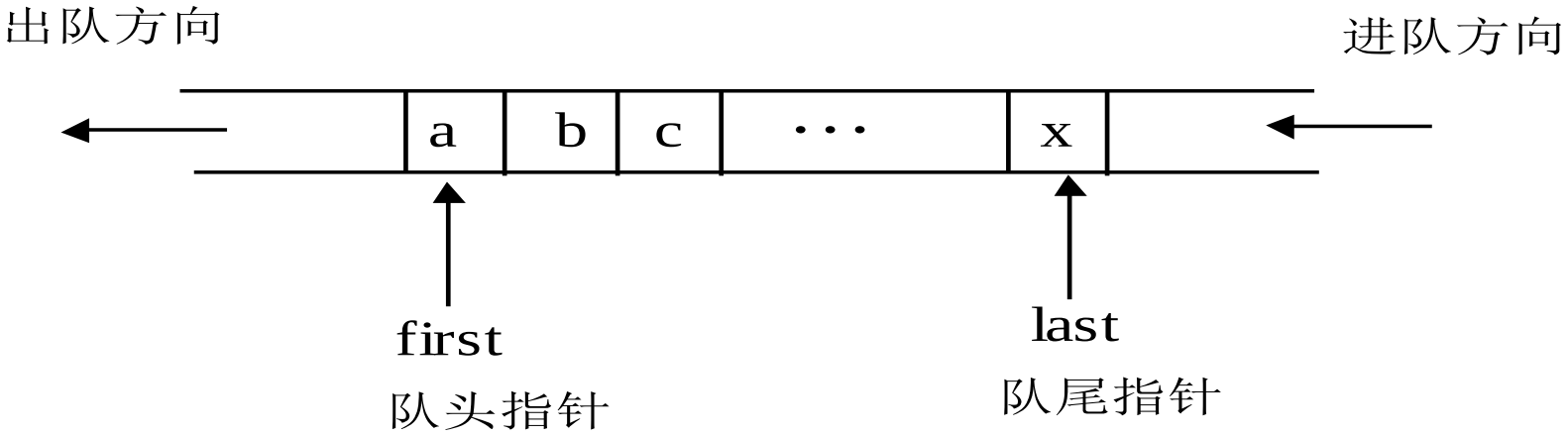
2. 队列的特点

队结构——管道

一端进入，另一端退出

先进先出表

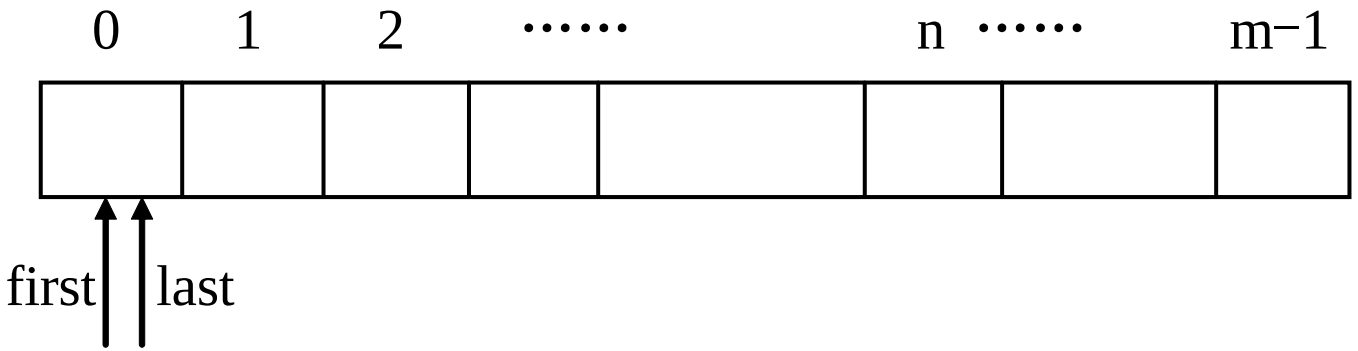
FIFO表



3. 顺序队

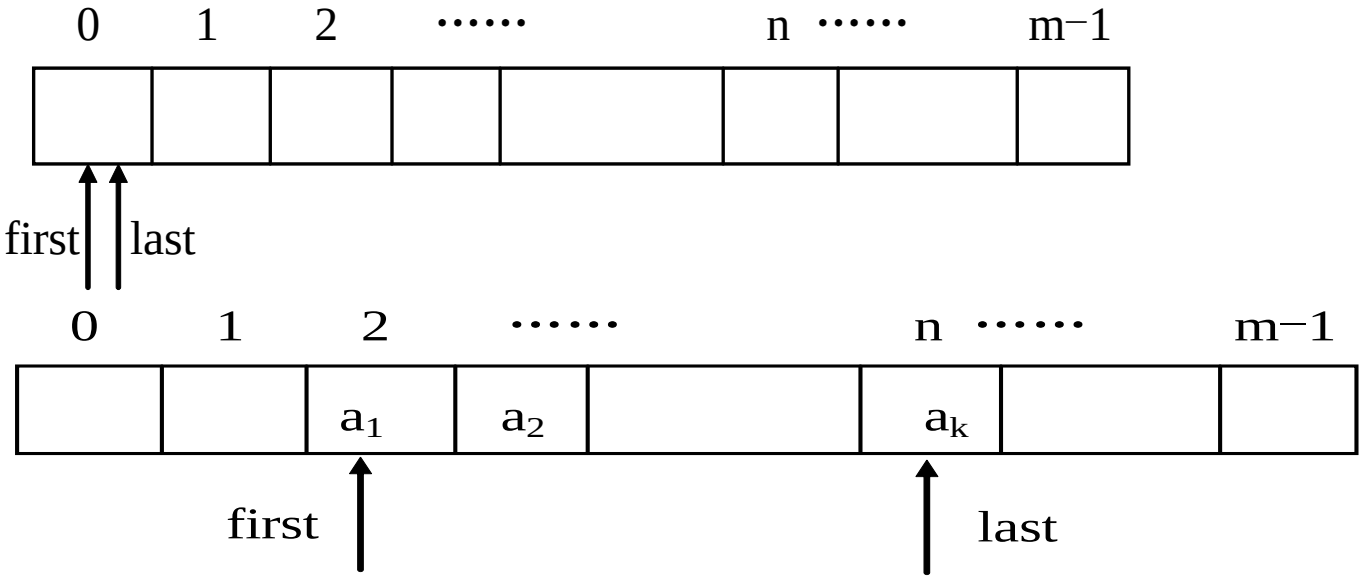
顺序队的基本用法：　首尾指针用法

数组q[m]:



顺序队的基本用法： 首尾指针用法

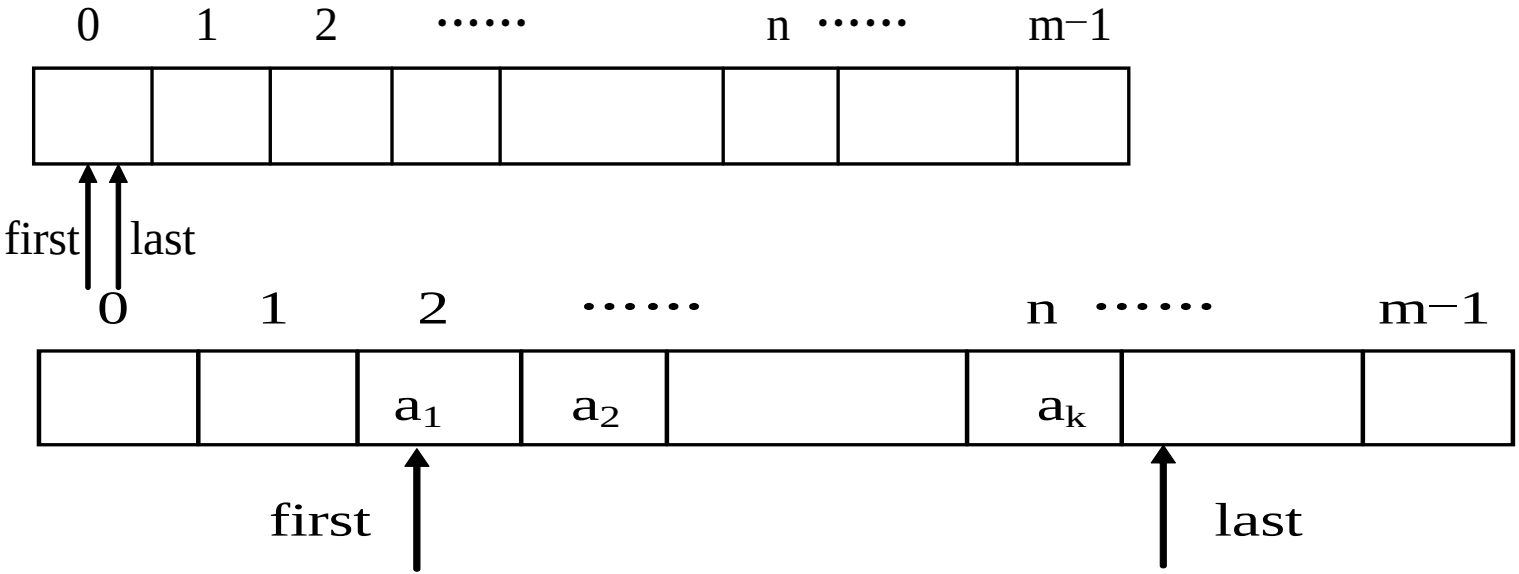
[1] last指向当前尾，first指向当前头（麻烦）



3. 顺序队

顺序队的基本用法：　首尾指针用法

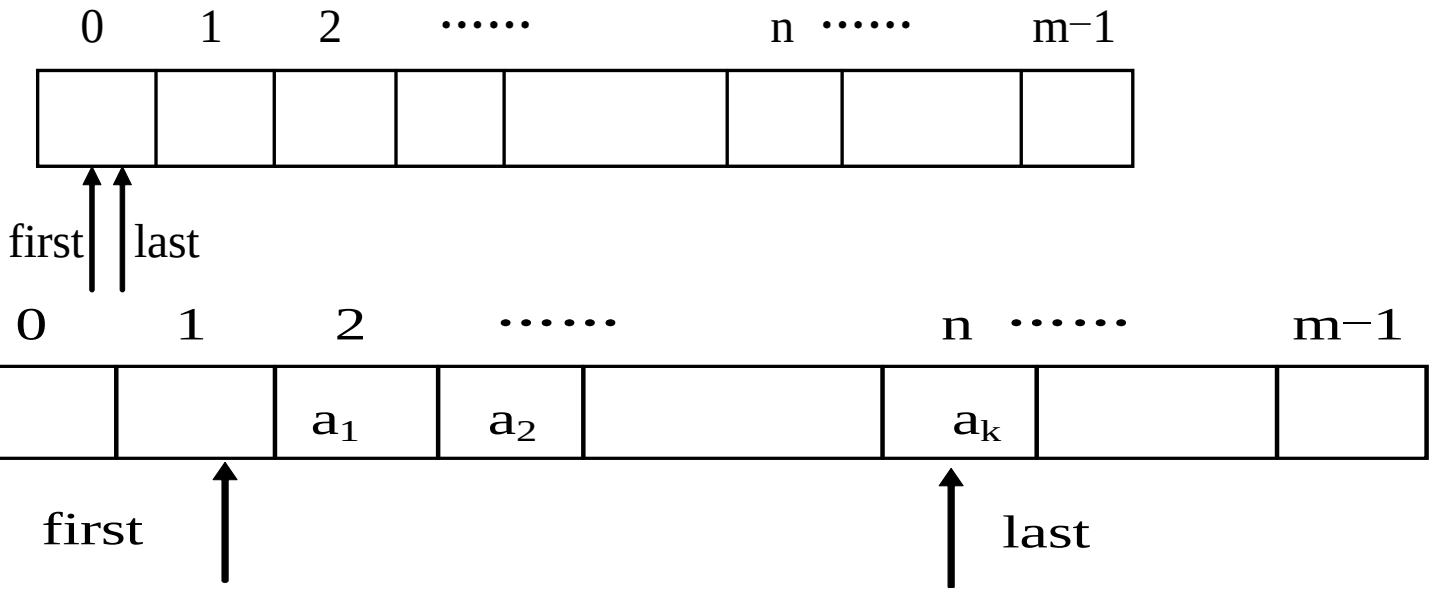
[2] 尾指针last前置



3. 顺序队

顺序队的基本用法：　首尾指针用法

[3] 首指针first后置



3. 顺序队

[1] last指向当前尾，first指向当前头（麻烦）

[2] 尾指针last前置

[3] 首指针first后置

- 如何判断队空？

  当first=last=i（i是0~m-1之间的任一值），都表示队空。

3. 顺序队

如何判断队满？

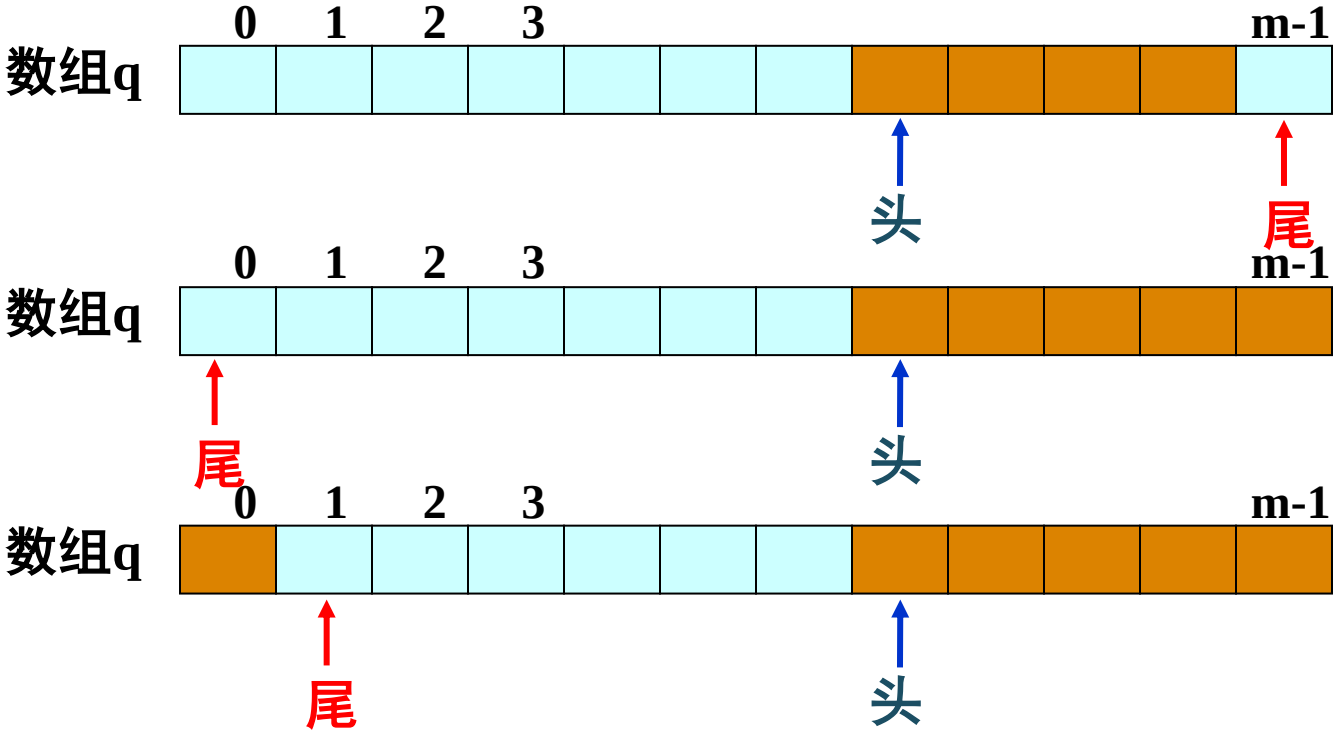
情况1：在程序执行期间，如果要求进队的元素总量不超过数组长度m，不会出现队满情况

如何判断队满？

情况2：在程序执行期间，需要进队处理的元素总量多于m个，但任何时刻，队中的元素数目始终小于m，仍可使用长度为m的数组存储（不会发生队满）

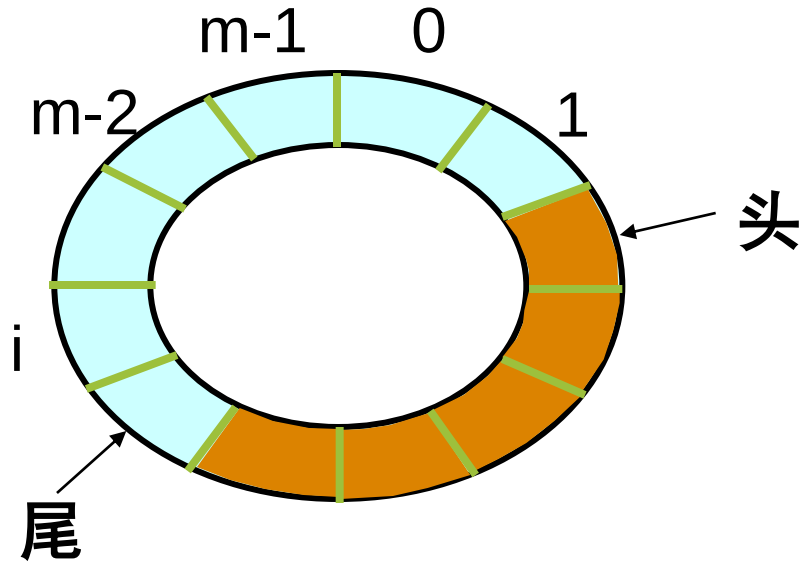
如何判断队满？

特点：重复使用已退队元素所占存储单元！



4. 循环队（环形队，cyclic queue）

特点：重复使用已退队元素所占存储单元！



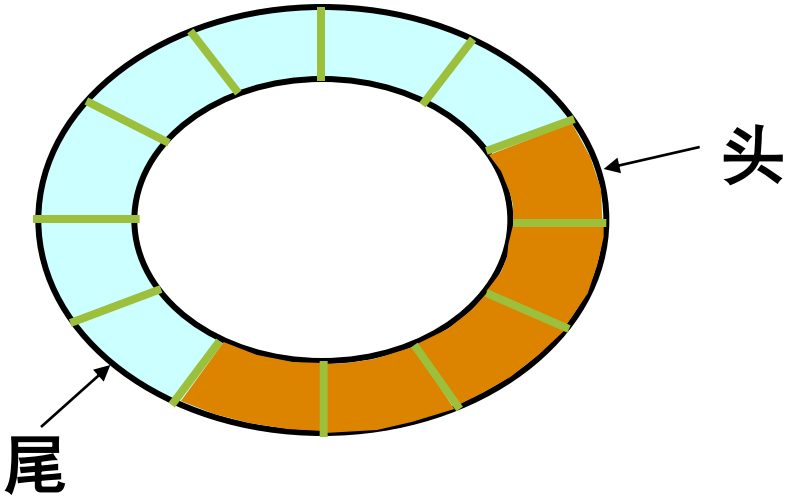
一般状态

first=(first+1)%m;

last=(last+1)%m;

4. 循环队（环形队，cyclic queue）

特点：重复使用已退队元素所占存储单元！



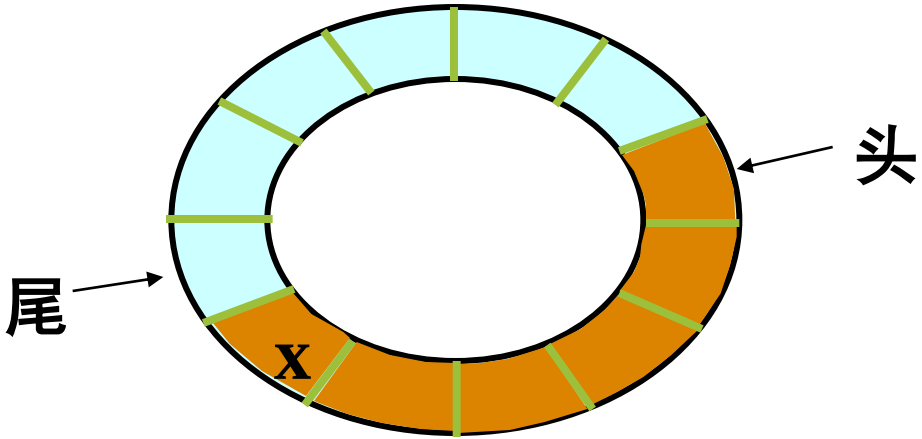
一般状态

元素x进队：

```
q[last]=x;
last=(last+1)%m;
```

4. 循环队（环形队，cyclic queue）

特点：重复使用已退队元素所占存储单元！



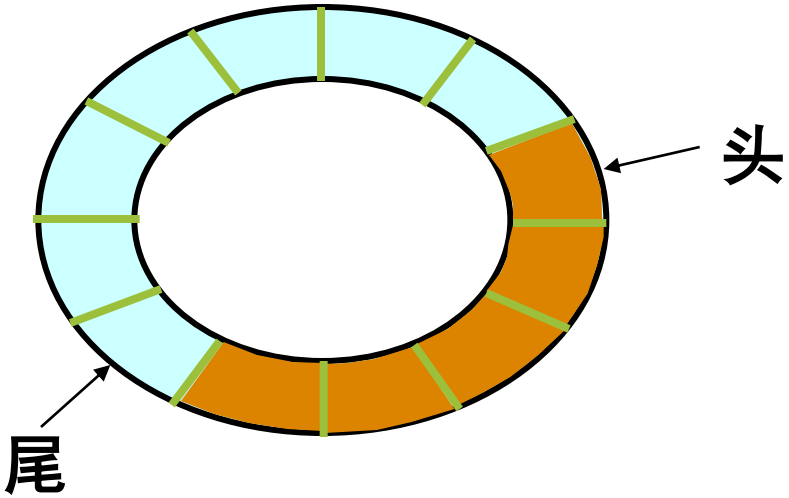
一般状态

元素x进队：

```
q[last]=x;
last=(last+1)%m;
```

4. 循环队（环形队，cyclic queue）

特点：重复使用已退队元素所占存储单元！



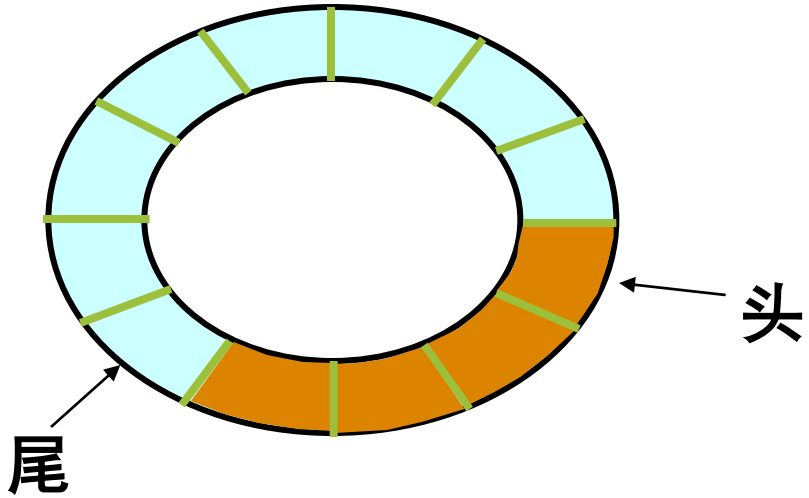
一般状态

元素x出队：

```
x=q[first];
first=(first+1)%m;
```

4. 循环队（环形队，cyclic queue）

特点：重复使用已退队元素所占存储单元！



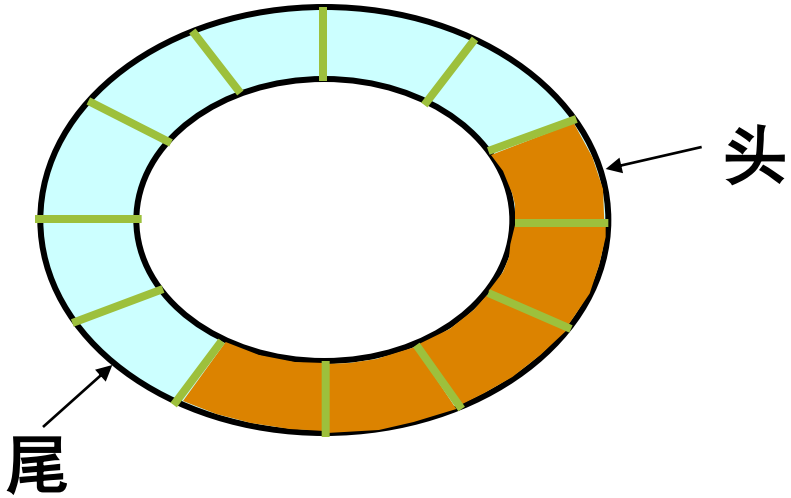
一般状态

元素x出队：

```
x=q[first];
first=(first+1)%m;
```

4. 循环队（环形队，cyclic queue）

特点：重复使用已退队元素所占存储单元！



一般状态

元素x进队：

```
q[last]=x;
last=(last+1)%m;
```

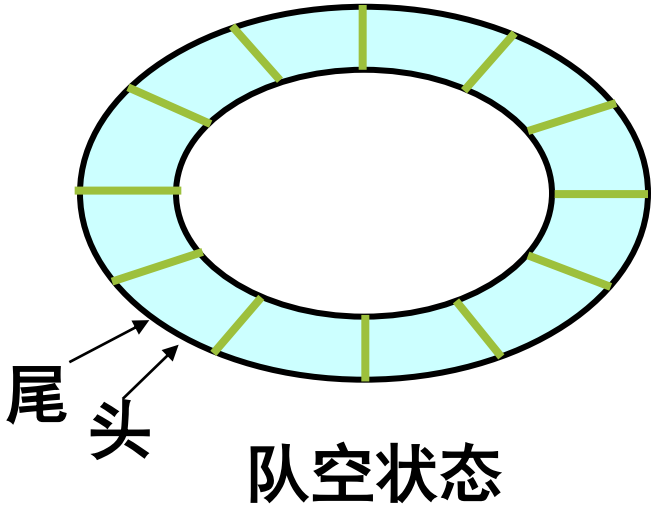
元素x出队：

```
x=q[first];
first=(first+1)%m;
```

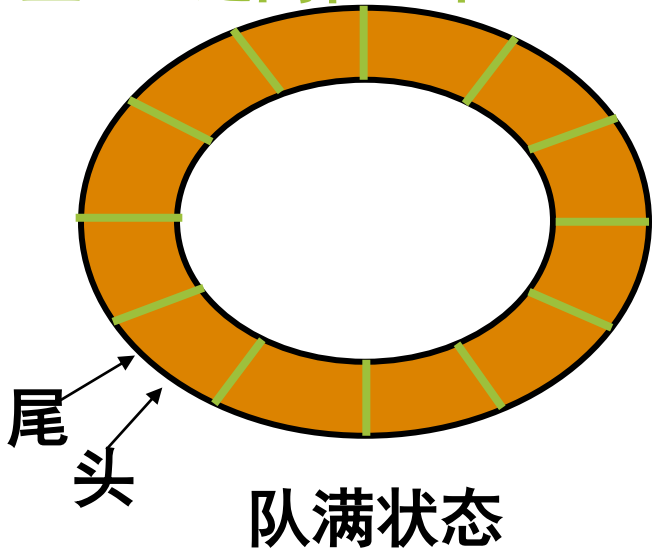
4. 循环队（环形队，cyclic queue）

队空状态：头尾指针相等，等于0至m-1之间任一值

队满状态：头尾指针相等，等于0至m-1之间任一值



队空状态



队满状态

不能靠头尾指针是否相等判断空？/ 满？

4. 循环队（环形队，cyclic queue）

开始时，队空，指针first=last=0

若一段时间内，只进不出，last赶上first，满

若一段时间内，只出不进， first赶上last，空

当first等于last时，无所适从

少用一个单元！

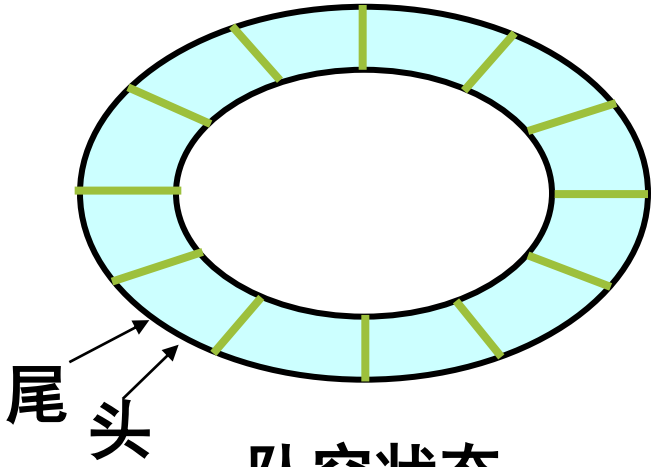
因此通用的计算队列长度公式为:

(last- first + m) %m　　m为表长

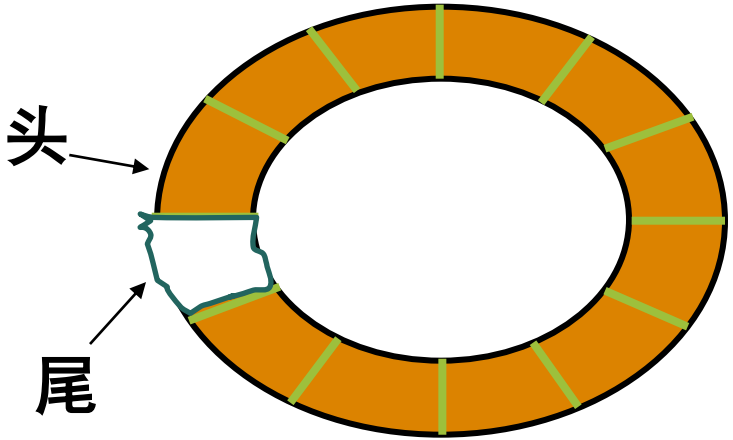
4. 循环队（环形队，cyclic queue）

队空状态：first=last

队满状态：(last+1)%m==first



队空状态



队满状态

5. 循环队的进队算法

```
int  addq(int q[ ], int first,  int &last, int x )
{
        if((last+1)%m==first)  return 0; //队满
        q[last]=x;
        last=(last+1)%m;
        return   1;  //进队成功
}
```

6. 循环队的出队算法

```
int delq(int q[  ], int &first,  int last,int &x )
{
        if(first==last)   return 0; //队空
        x=q[first];
        first=(first+1)%m;
        return    1; //出队成功
}
```

学习目标和要求

1.能够写出链式队列的入队和出队算法；

7. 链队的进出队算法

链式队可采用单向加头链表结构

first为首指针，last为尾指针

进队：

将新结点插在表尾处，last移向新结点

出队：

将头结点后的第一个元素出队，删除头结点

(头结点不固定)保证尾结点不被删除

first和last同时指向头结点作为队空的判断条件

7. 链队的进队算法

```
 int  saddq(ptr &last, int x)
{ ptr p;
1.   p=(ptr)malloc(sizeof(snode));
2.   if(p==NULL)return  0; //空间分配失败
3.   p->data=x;
4.   last->next=p; //表尾插入法
5.   last=p;
6.   last->next=NULL;
7.   return  1; //进队成功
 }
```

8. 链队的出队算法

```
int  sdelq(ptr &first,ptr last, int &x)
{ ptr  p;
1.  if(first==last)return 0; //队空
2.  p=first; //表头删除法
3.  first=p->next;
4.  x=first->data; //x出队
5.  free(p);
6.  return  1; //出队成功
 }
```

注意：与以前的表头删除法不同，因为当表删空时，last就会丢失，因此采用删监督元的做法，将要删除的元素作为新的监督元。

小结

- 队列的特点：先进先出
- 队列通常采用循环队列形式