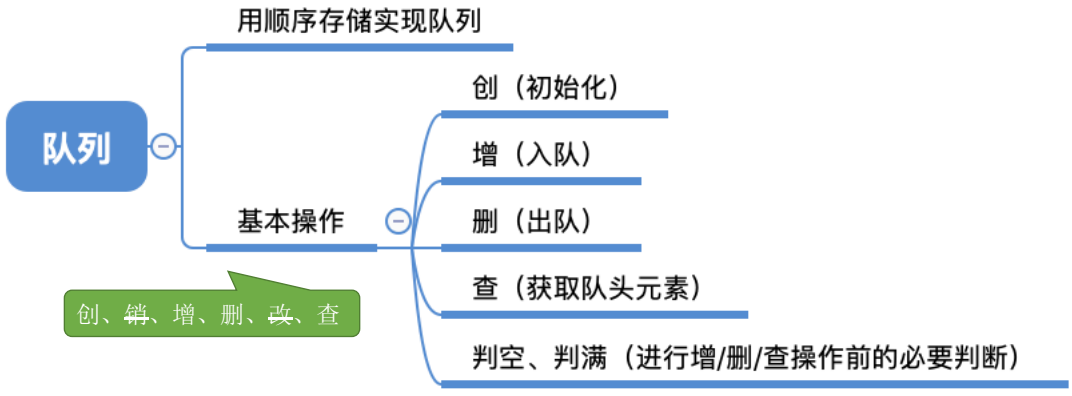
本节内容

队列

顺序实现

1

知识总览



2

队列的顺序实现

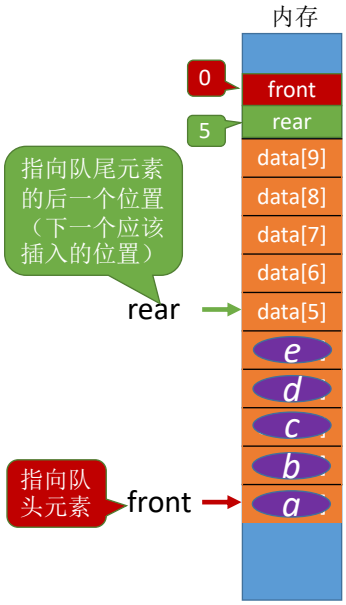
```
#define MaxSize 10              //定义队列中元素的最大个数
typedef struct{
ElemType data[MaxSize];         //用静态数组存放队列元素
int front,rear;                 //队头指针和队尾指针
} SqQueue;

void testQueue(){
    SqQueue Q;          //声明一个队列（顺序存储）
    //...后续操作...
}
```

Sq：sequence —— 顺序

连续的存储空间，大小 MaxSize*sizeof(ElemType)

rear
英 [nə(r)] 美 [nɪr]
n. 后面；后方部队；屁股
adj. 后方的，后面的；背面的

front
英 [frʌnt] 美 [frʌnt]
n. 前面；正面；前线
vt. 面对；朝向；对付
vi. 朝向
adj. 前面的；正面的

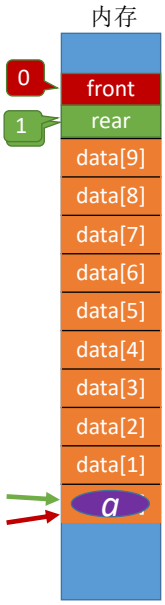


3

初始化操作

```
#define MaxSize 10              //定义队列中元素的最大个数
typedef struct{
ElemType data[MaxSize];         //用静态数组存放队列元素
int front,rear;                 //队头指针和队尾指针
} SqQueue;

//初始化队列
void InitQueue(SqQueue &Q){
    //初始时 队头、队尾指针指向0
    Q.rear=Q.front=0;
}

void testQueue(){
    //声明一个队列（顺序存储）
    SqQueue Q;
    InitQueue(Q);
    //...后续操作...
}

//判断队列是否为空
bool QueueEmpty(SqQueue Q){
    if(Q.rear==Q.front)    //队空条件
        return true;
    else
        return false;
}
```

增删改查



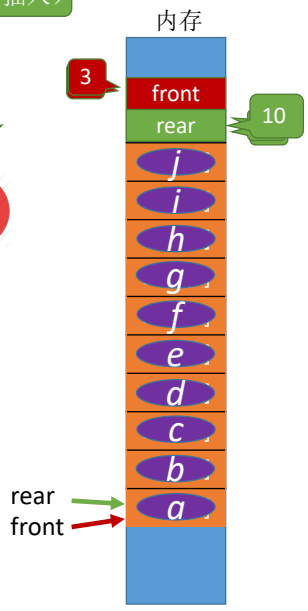
4

入队操作

只能从队尾入队（插入）

```
#define MaxSize 10              //定义队列中元素的最大个数
typedef struct{
ElemType data[MaxSize];         //用静态数组存放队列元
int front,rear;                 //队头指针和队尾指针
} SqQueue;

//入队
bool EnQueue(SqQueue &Q,ElemType x){
    if(队列已满)
        return false;           //队满则报错
    Q.data[Q.rear]=x;           //将x插入队尾
    Q.rear=Q.rear+1;            //队尾指针后移
    return true;
}
```

队列已满的条件：rear==MaxSize ???



王道考研/CSKAOYAN.COM

5

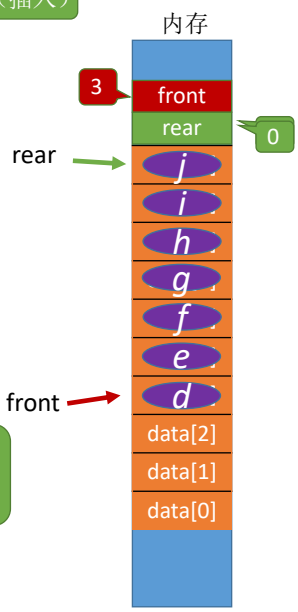
入队操作

只能从队尾入队（插入）

```
#define MaxSize 10              //定义队列中元素的最大个数
typedef struct{
ElemType data[MaxSize];         //用静态数组存放队列元素
int front,rear;                 //队头指针和队尾指针
} SqQueue;

//入队
bool EnQueue(SqQueue &Q,ElemType x){
    if(队列已满)
        return false;           //队满则报错
    Q.data[Q.rear]=x;           //新元素插入队尾
    Q.rear=(Q.rear+1)%MaxSize;  //队尾指针加1取模
    return true;
}
```

{0, 1, 2, ..., MaxSize-1} 将存储空间在逻辑上变成了“环状”

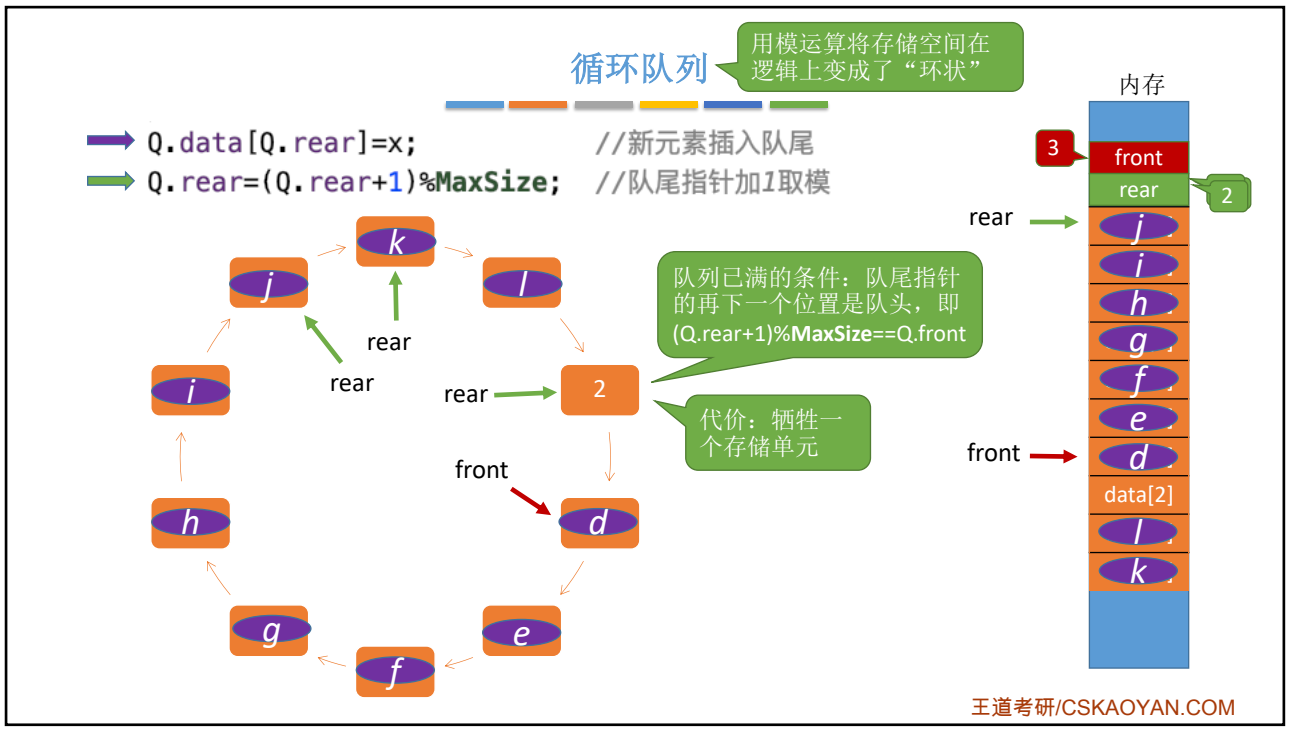
模运算将无限的整数域映射到有限的整数集合 {0, 1, 2, ..., b-1} 上



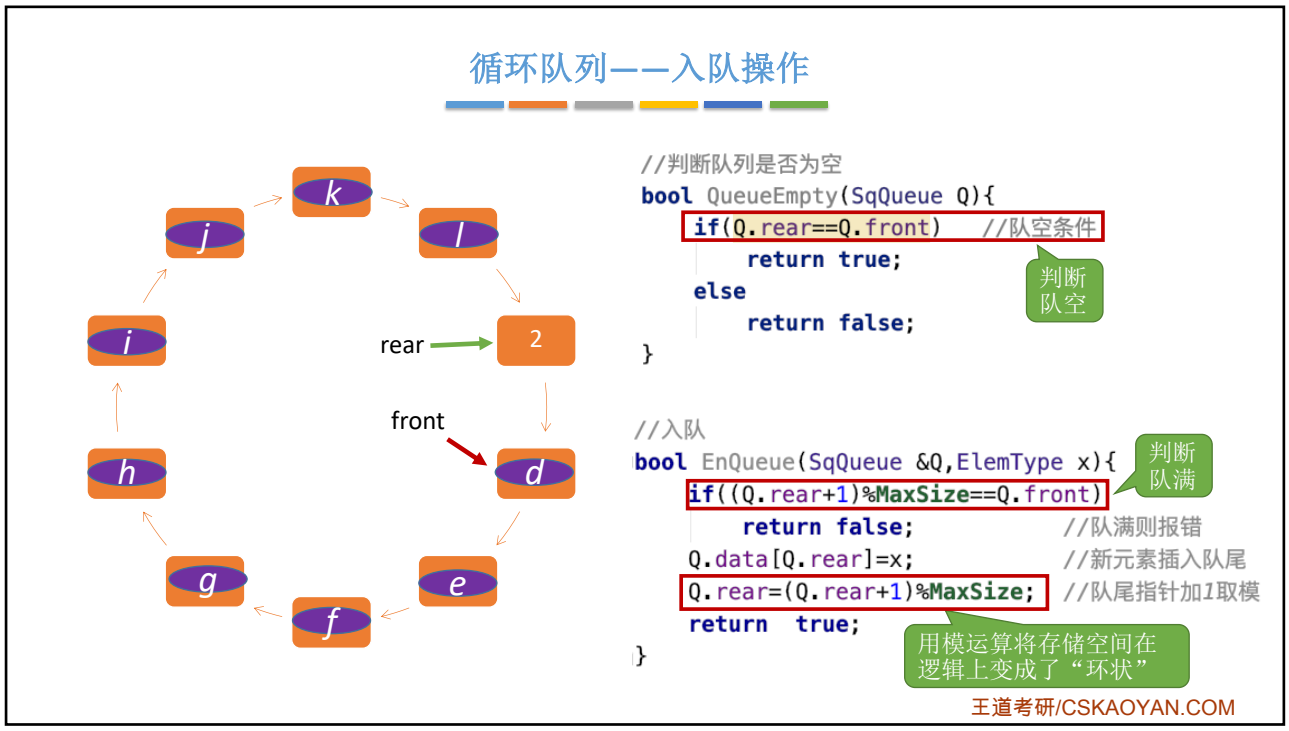
跨考Tips：取模运算，即取余运算。两个整数 a,b，a%b == a除以b的余数
在《数论》中，通常表示为 a MOD b

王道考研/CSKAOYAN.COM

6


循环队列
用模运算将存储空间在逻辑上变成了“环状”
Q.data[Q.rear]=x; //新元素插入队尾
Q.rear=(Q.rear+1)%MaxSize; //队尾指针加1取模
队列已满的条件：队尾指针的再下一个位置是队头，即(Q.rear+1)%MaxSize==Q.front
代价：牺牲一个存储单元
rear
front
内存
3
front
rear
2
data[2]
王道考研/CSKAOYAN.COM

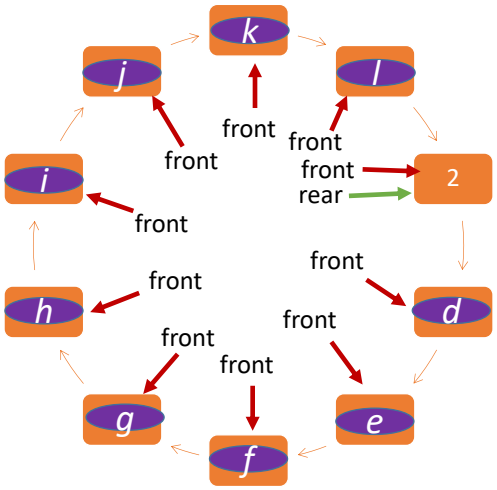
7


循环队列——入队操作
//判断队列是否为空
bool QueueEmpty(SqQueue Q){
if(Q.rear==Q.front) //队空条件
return true;
else
return false;
}
判断队空
//入队
bool EnQueue(SqQueue &Q,ElemType x){
if((Q.rear+1)%MaxSize==Q.front)
return false; //队满则报错
Q.data[Q.rear]=x; //新元素插入队尾
Q.rear=(Q.rear+1)%MaxSize; //队尾指针加1取模
return true;
}
判断队满
用模运算将存储空间在逻辑上变成了“环状”
rear
front
王道考研/CSKAOYAN.COM

8

循环队列——出队操作

只能让队头元素出队




```
//出队（删除一个队头元素，并用x返回）
bool DeQueue(SqQueue &Q,ElemType &x){
    if(Q.rear==Q.front)          判断队空
        return false;    //队空则报错
    x=Q.data[Q.front];
    Q.front=(Q.front+1)%MaxSize;
    return true;
}                                队头指针后移

    //获得队头元素的值，用x返回
    bool GetHead(SqQueue Q,ElemType &x){
        if(Q.rear==Q.front)
            return false;    //队空则报错
        x=Q.data[Q.front];
        return true;
    }
```

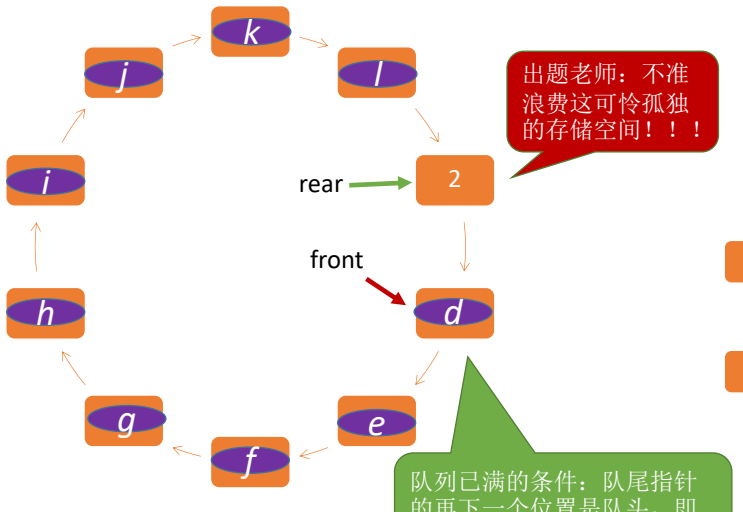

9

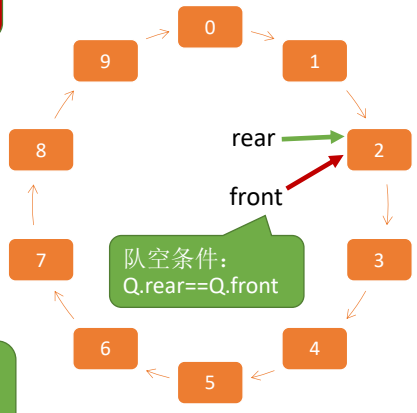
方案一：判断队列已满/已空

队列元素个数：
(rear+MaxSize-front)%MaxSize

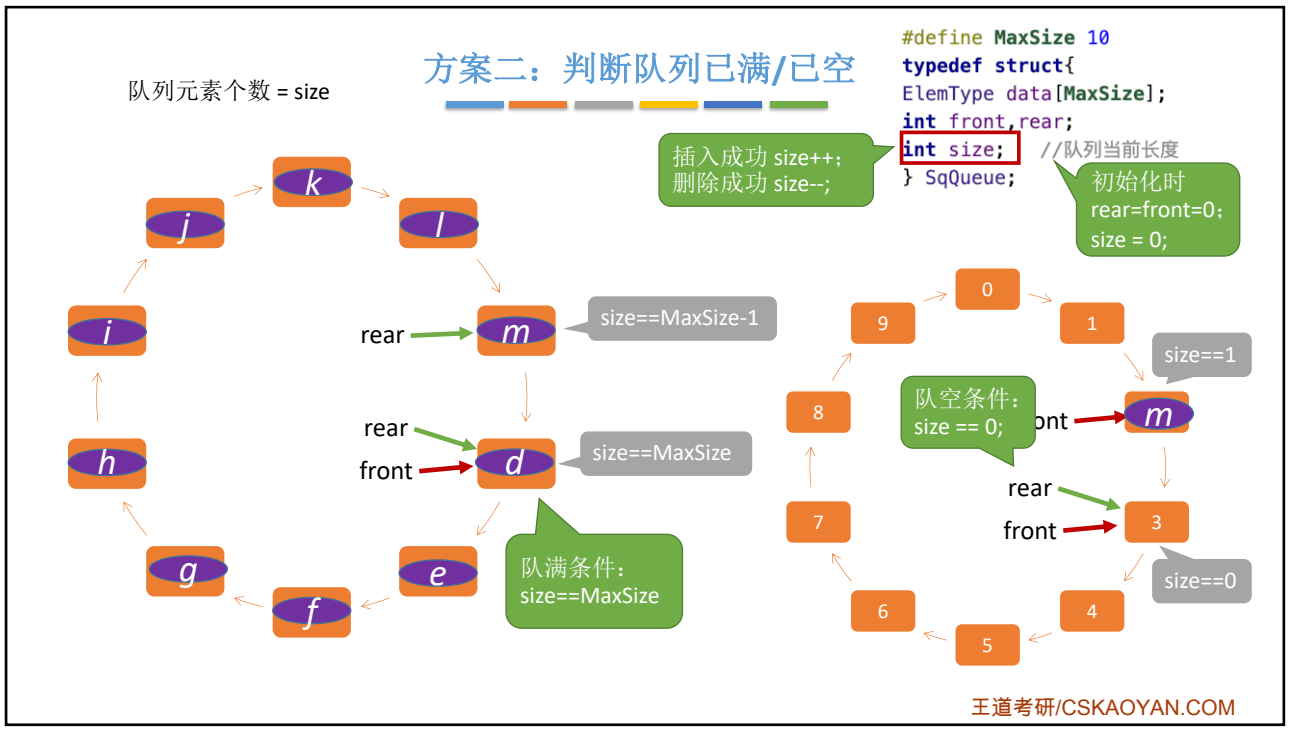
```
#define MaxSize 10
typedef struct{
ElemType data[MaxSize];
int front,rear;
} SqQueue;
```

初始化时
rear=front=0

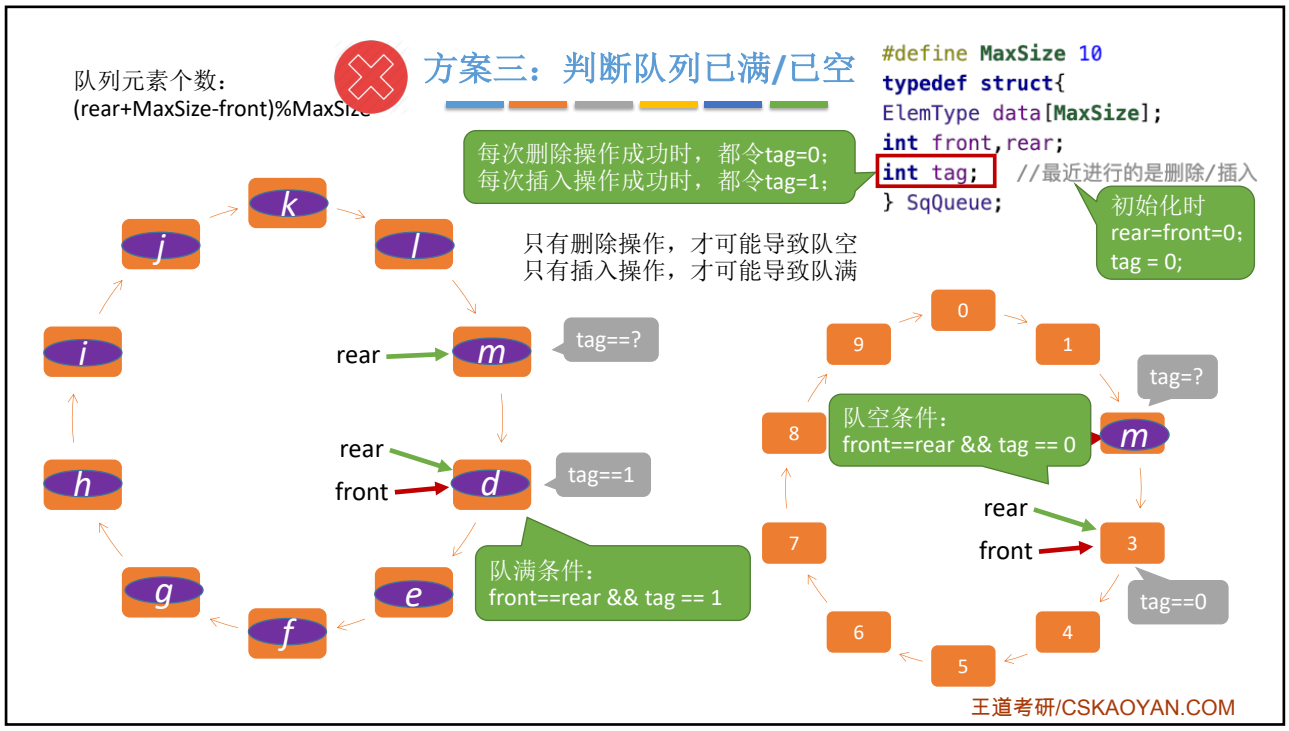




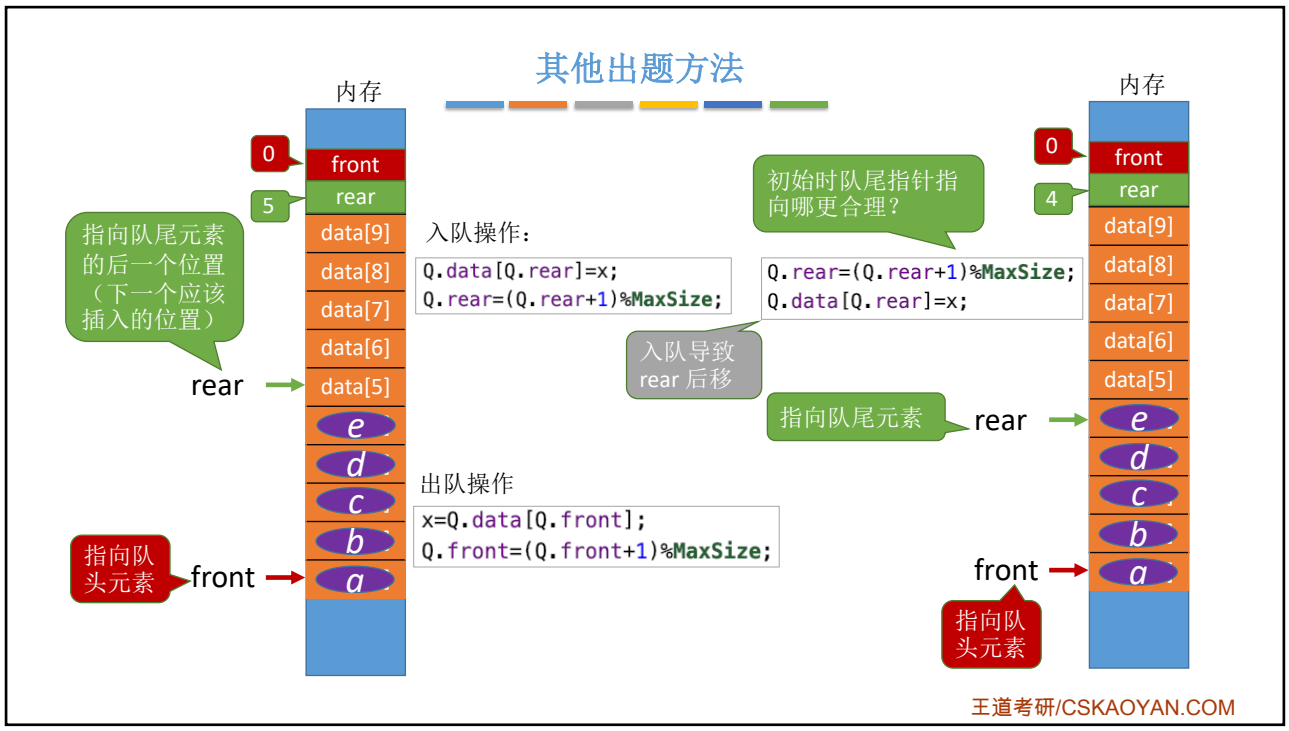
10


方案二：判断队列已满/已空
队列元素个数 = size
#define MaxSize 10
typedef struct{
ElemType data[MaxSize];
int front,rear;
int size; //队列当前长度
} SqQueue;
插入成功 size++;
删除成功 size--;
初始化时
rear=front=0;
size = 0;
rear
size==MaxSize-1
rear
front
size==MaxSize
队满条件：
size==MaxSize
size==1
队空条件：
size == 0;
rear
front
size==0
王道考研/CSKAOYAN.COM

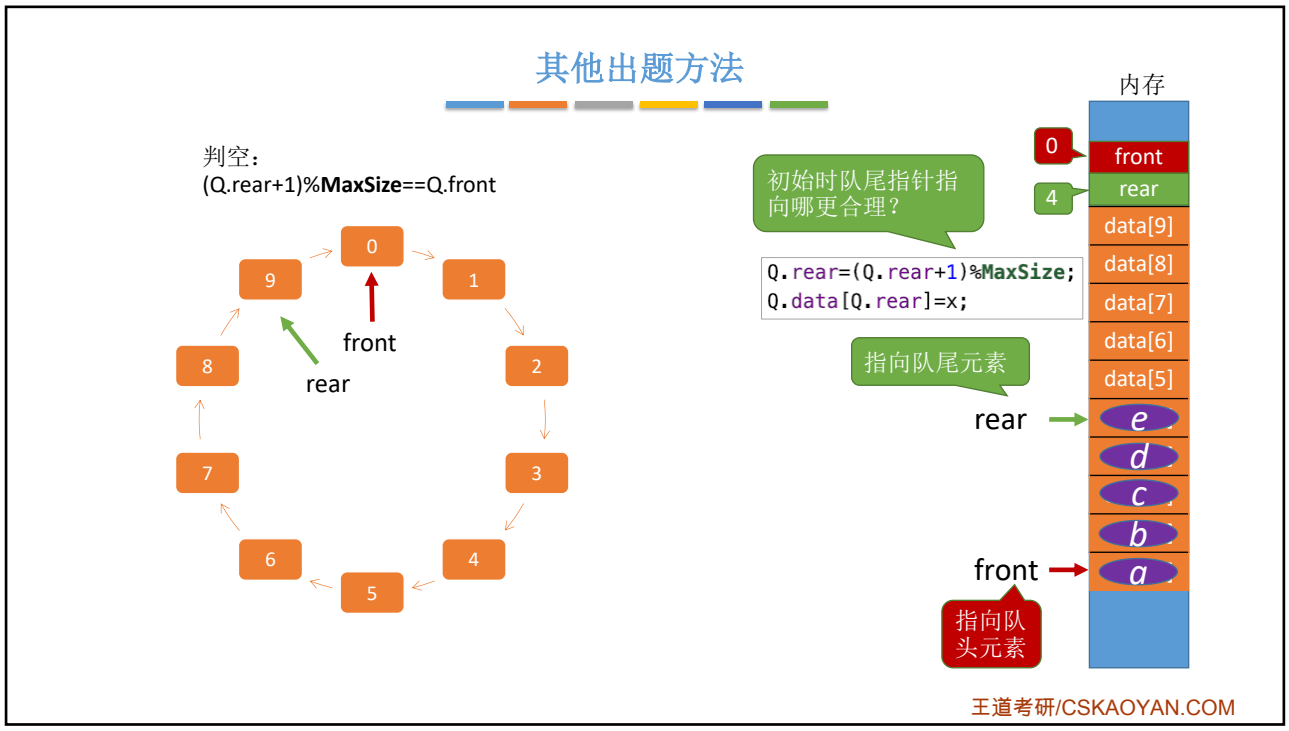
11


方案三：判断队列已满/已空
队列元素个数：
(rear+MaxSize-front)%MaxSize
#define MaxSize 10
typedef struct{
ElemType data[MaxSize];
int front,rear;
int tag; //最近进行的是删除/插入
} SqQueue;
每次删除操作成功时，都令tag=0;
每次插入操作成功时，都令tag=1;
只有删除操作，才可能导致队空
只有插入操作，才可能导致队满
初始化时
rear=front=0;
tag = 0;
rear
tag==?
rear
front
tag==1
队满条件：
front==rear && tag == 1
tag=?
队空条件：
front==rear && tag == 0
rear
front
tag==0
王道考研/CSKAOYAN.COM

12


其他出题方法
内存
0
front
rear
5
data[9]
data[8]
data[7]
data[6]
data[5]
e
d
c
b
a
指向队尾元素的后一个位置（下一个应该插入的位置）
rear
指向队头元素
front
入队操作：
Q.data[Q.rear]=x;
Q.rear=(Q.rear+1)%MaxSize;
出队操作
x=Q.data[Q.front];
Q.front=(Q.front+1)%MaxSize;
初始时队尾指针指向哪更合理？
Q.rear=(Q.rear+1)%MaxSize;
Q.data[Q.rear]=x;
入队导致rear后移
指向队尾元素
内存
0
front
4
rear
data[9]
data[8]
data[7]
data[6]
data[5]
e
d
c
b
a
rear
front
指向队头元素
王道考研/CSKAOYAN.COM

13


其他出题方法
判空：
(Q.rear+1)%MaxSize==Q.front
0
1
2
3
4
5
6
7
8
9
front
rear
初始时队尾指针指向哪更合理？
Q.rear=(Q.rear+1)%MaxSize;
Q.data[Q.rear]=x;
指向队尾元素
rear
front
指向队头元素
内存
0
front
4
rear
data[9]
data[8]
data[7]
data[6]
data[5]
e
d
c
b
a
王道考研/CSKAOYAN.COM

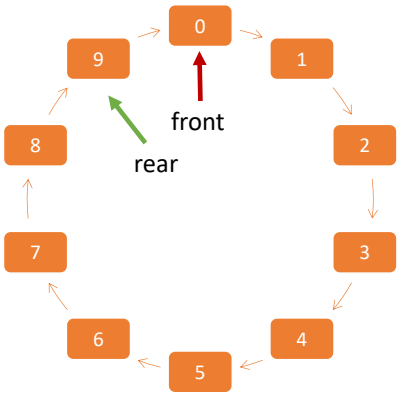
14

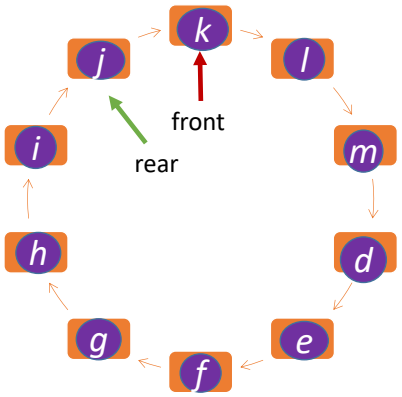
其他出题方法

方案一：牺牲一个存储单元
方案二：增加辅助变量

判空：
(Q.rear+1)%**MaxSize**==Q.front

判满：
(Q.rear+1)%**MaxSize**==Q.front？

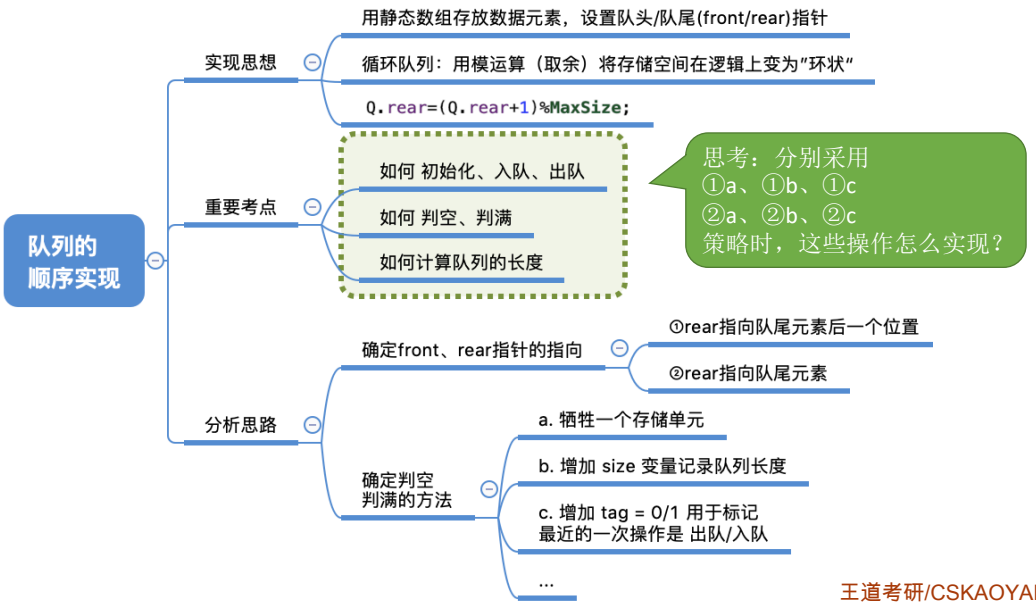




王道考研/CSKAOYAN.COM

15

知识回顾与重要考点



王道考研/CSKAOYAN.COM

16