

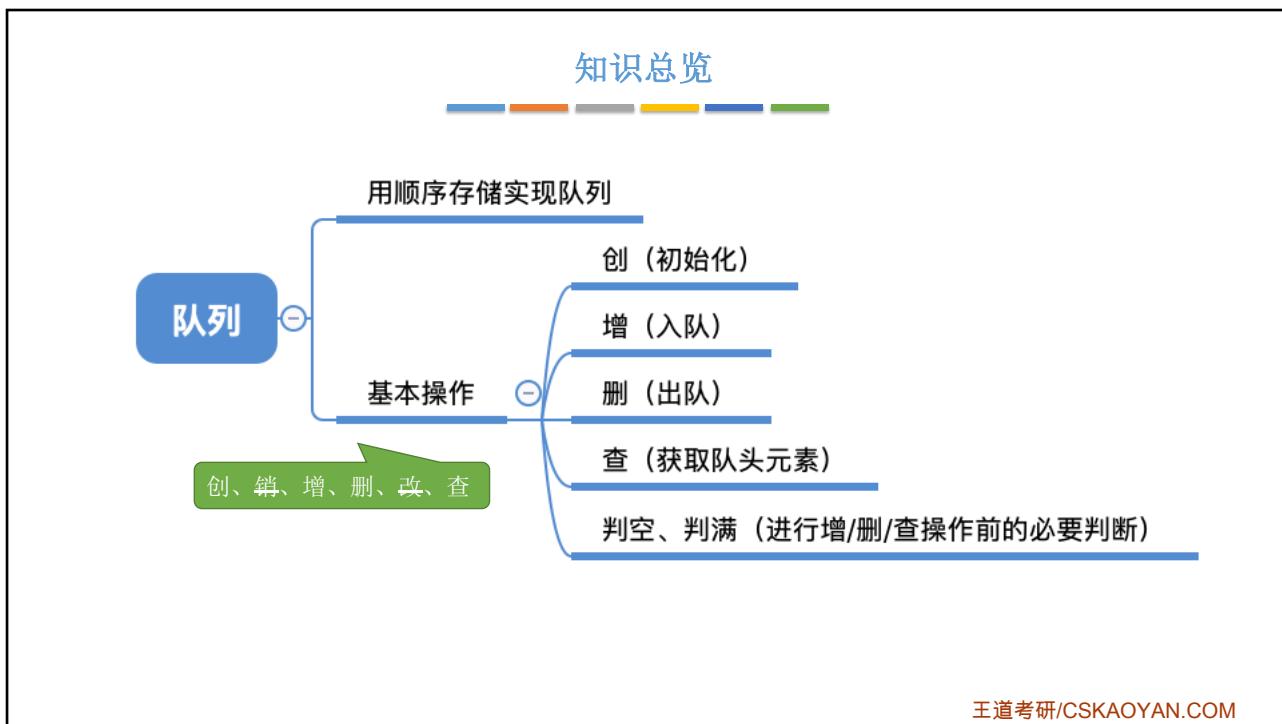
本节内容

队列

顺序实现

王道考研/CSKAOYAN.COM

1



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{           // 用静态数组存放队列元素
    ELEMTYPE data[MaxSize]; // 队头指针和队尾指针
    int front,rear;        // 连续的存储空间，大小 MaxSize*sizeof(ELEMTYPE)
} SqQueue;

```

Sq: sequence —— 顺序

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

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

front

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

内存
0 front
5 rear
data[9]
data[8]
data[7]
data[6]
data[5]
e
d
c
b
a

王道考研/CSKAOYAN.COM

3

初始化操作

```

#define MaxSize 10           // 定义队列中元素的最大个数
typedef struct{           // 用静态数组存放队列元素
    ELEMTYPE data[MaxSize]; // 队头指针和队尾指针
    int front,rear;        // 连续的存储空间，大小 MaxSize*sizeof(ELEMTYPE)
} 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...
}

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

内存
0 front
1 rear
data[9]
data[8]
data[7]
data[6]
data[5]
data[4]
data[3]
data[2]
data[1]
a

王道考研/CSKAOYAN.COM

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{           // 用静态数组存放队列元素
    ElemType data[MaxSize]; // 队头指针和队尾指针
    int front,rear;        // 队列已满的条件:
} SqQueue;                // rear==MaxSize ???
```

内存

3 10

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{           // 用静态数组存放队列元素
    ElemType data[MaxSize]; // 队头指针和队尾指针
    int front,rear;        // 队列已满的条件:
} 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)%MaxSize; // 队尾指针加1取模
    return true;
}
```

将存储空间在逻辑上变成了“环状”

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 $a \text{ MOD } b$

内存

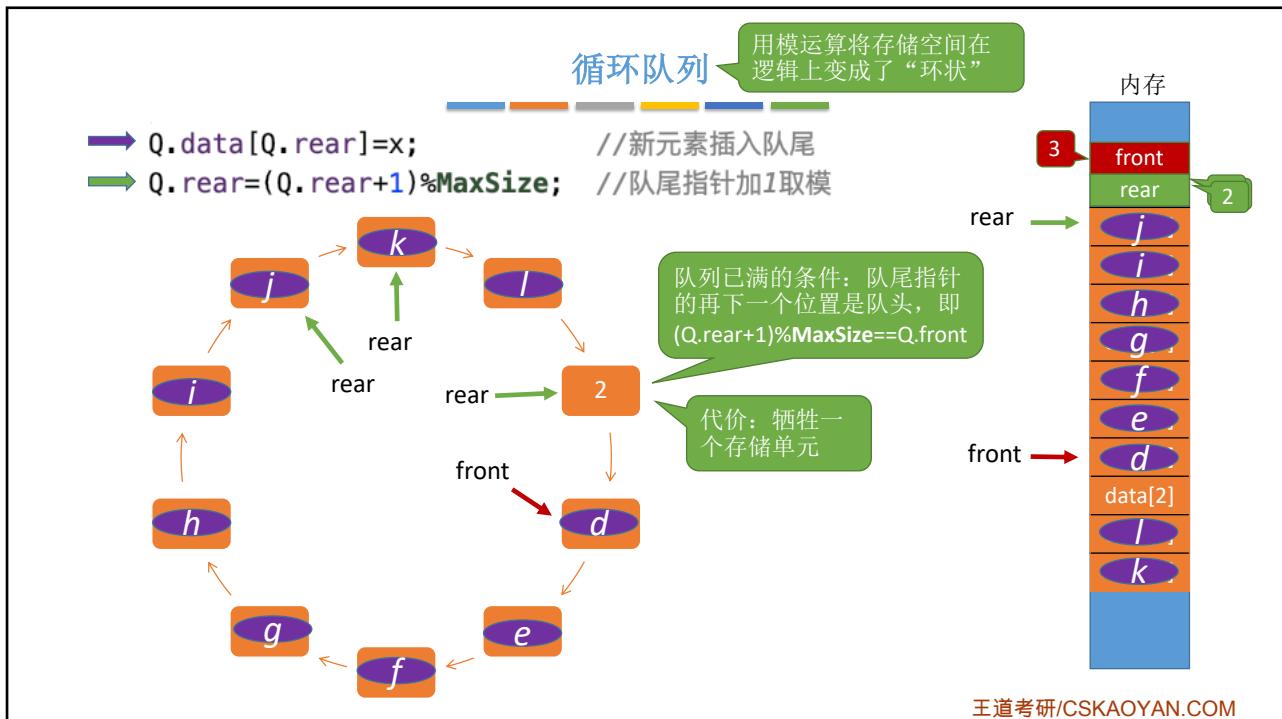
3 0

front

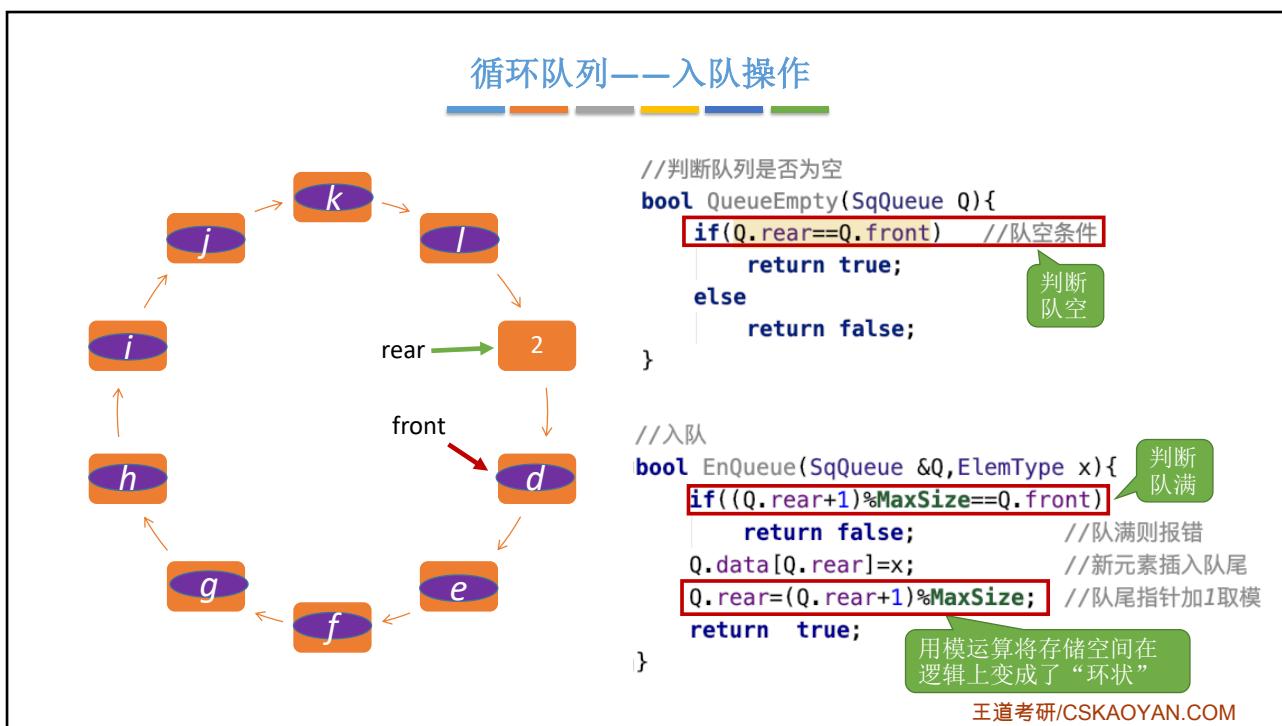
rear

王道考研/CSKAOYAN.COM

6



7



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]; // 赋值
        Q.front=(Q.front+1)%MaxSize; // 队头指针后移
        return true;
    }

    // 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;
    }

```

王道考研/CSKAOYAN.COM

9

方案一：判断队列已满/已空

队列元素个数：
 $(\text{rear}+\text{MaxSize}-\text{front})\% \text{MaxSize}$

出题老师：不准浪费这可怜孤独的存储空间！！！

```

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

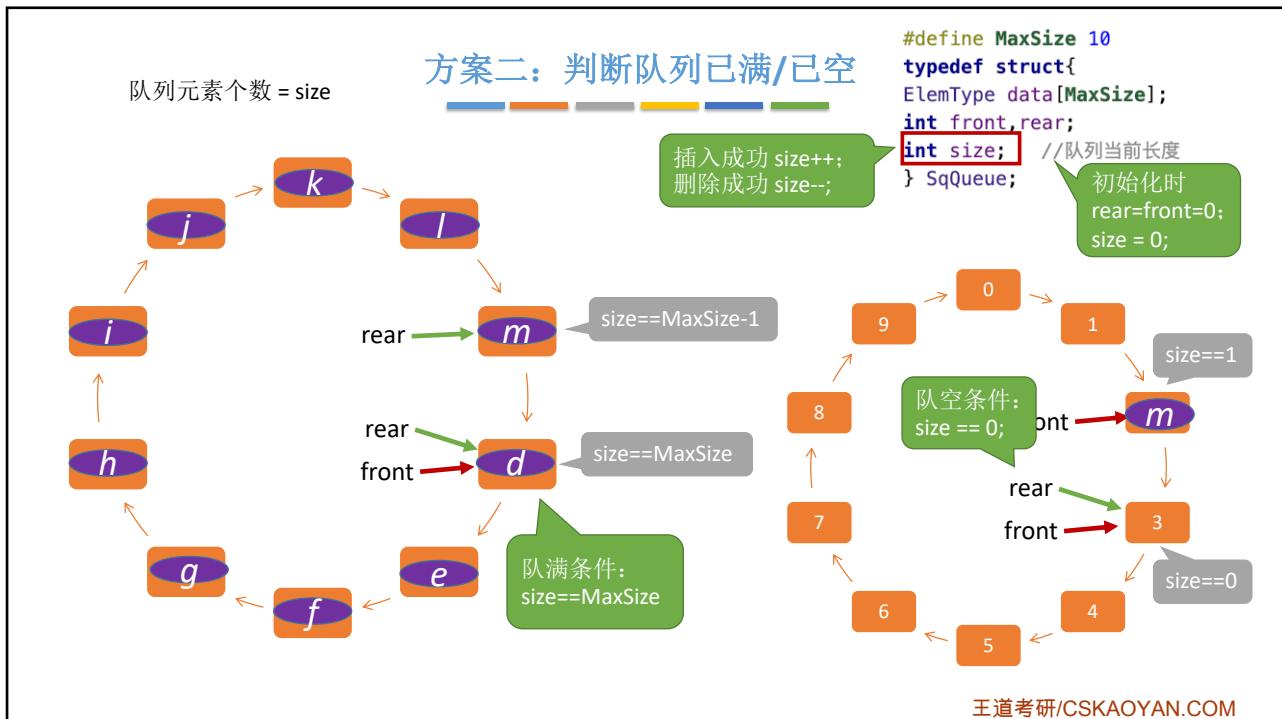
```

初始化时
rear=front=0

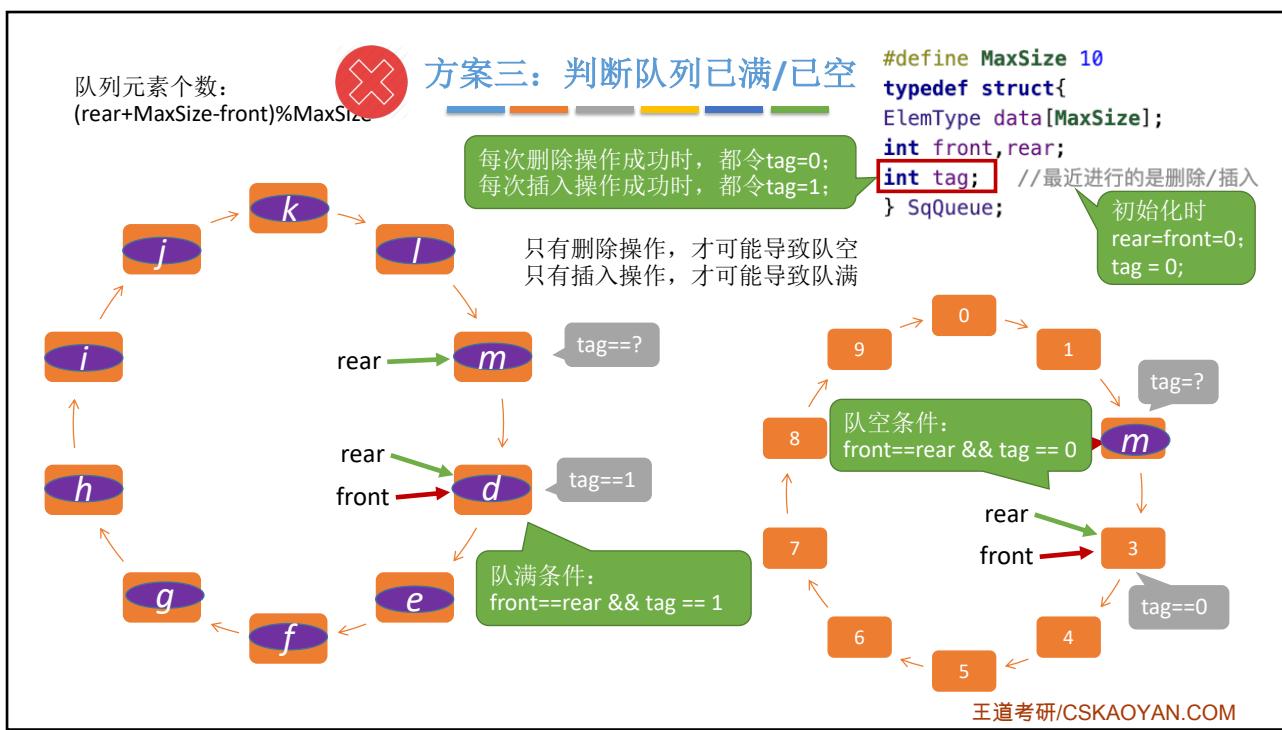
队空条件：
 $\text{Q}.rear == \text{Q}.front$

王道考研/CSKAOYAN.COM

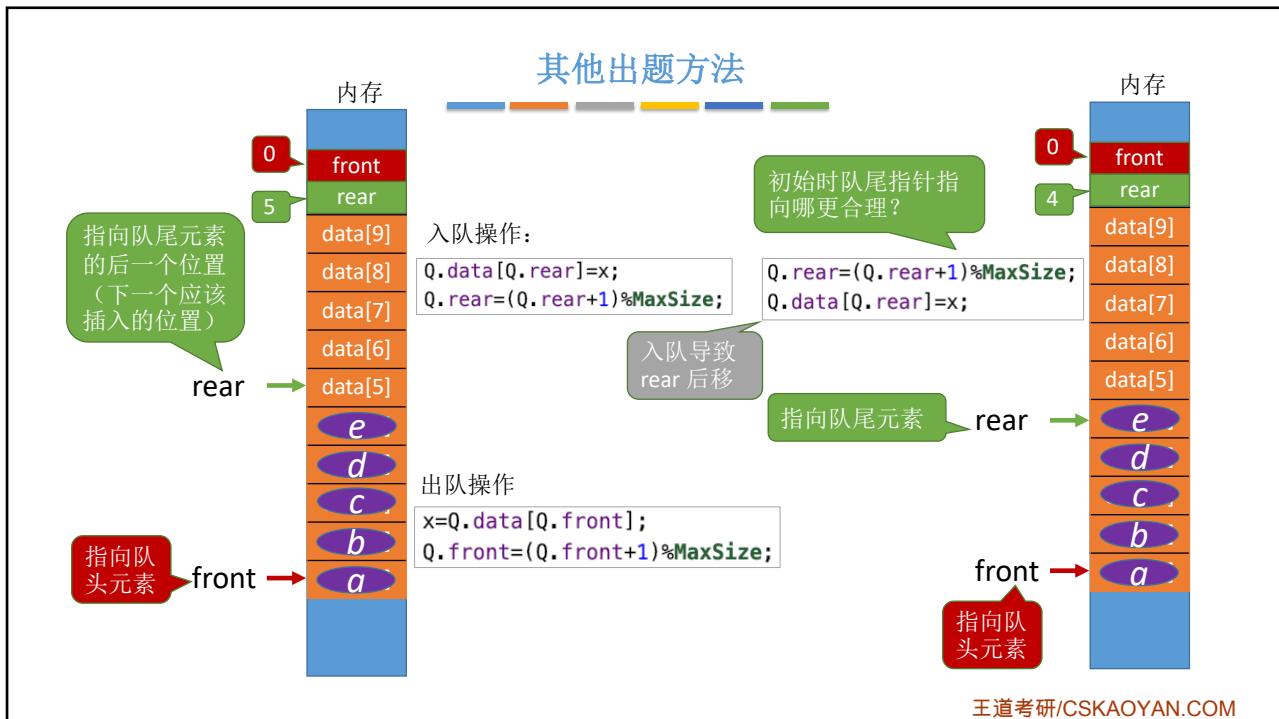
10



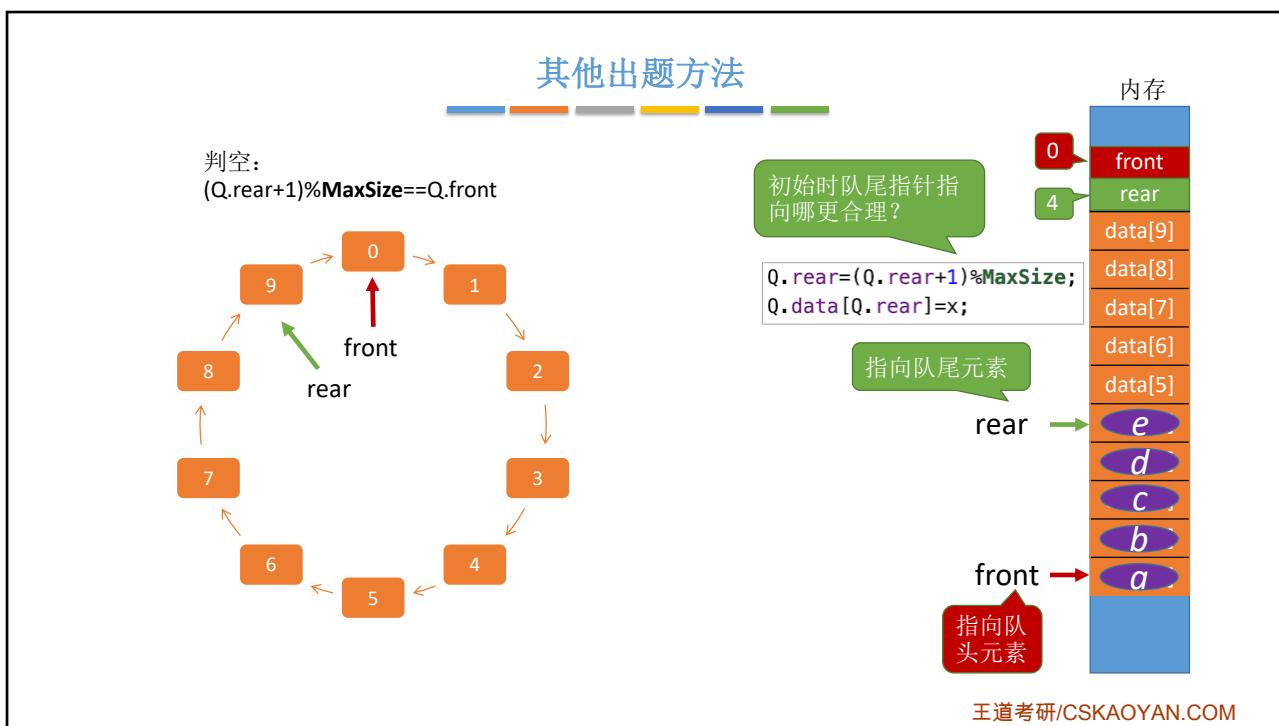
11



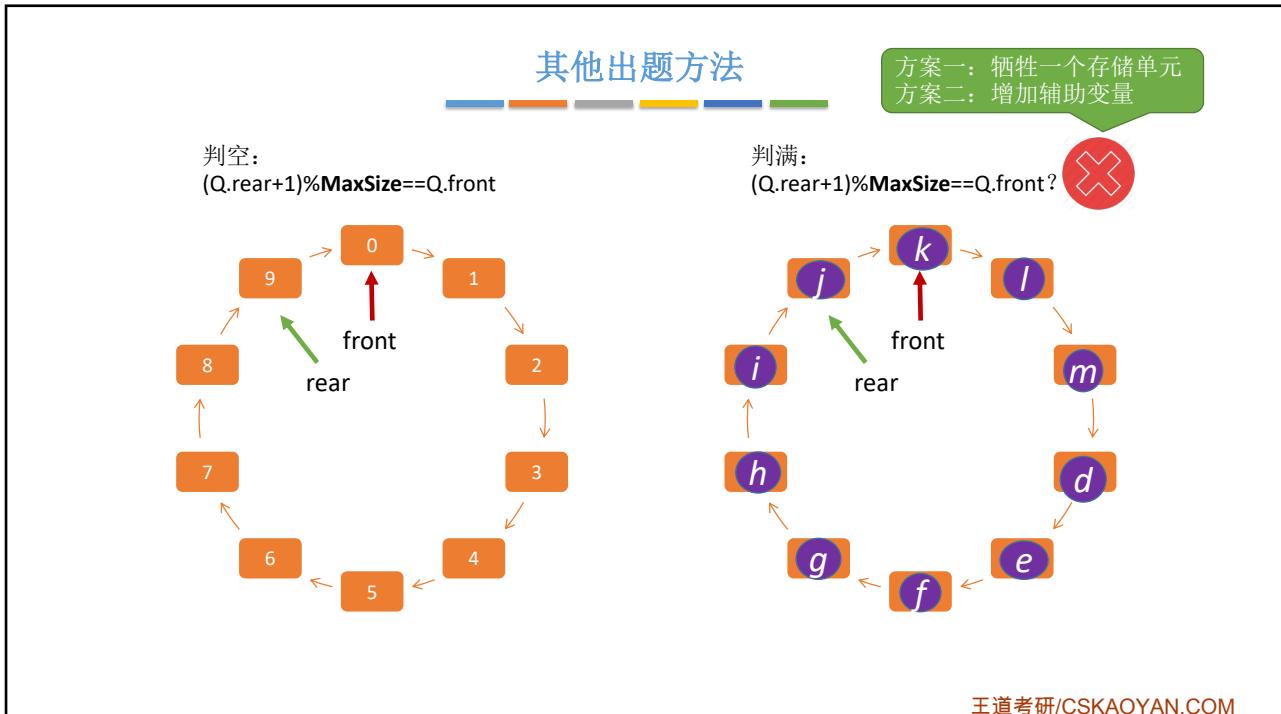
12



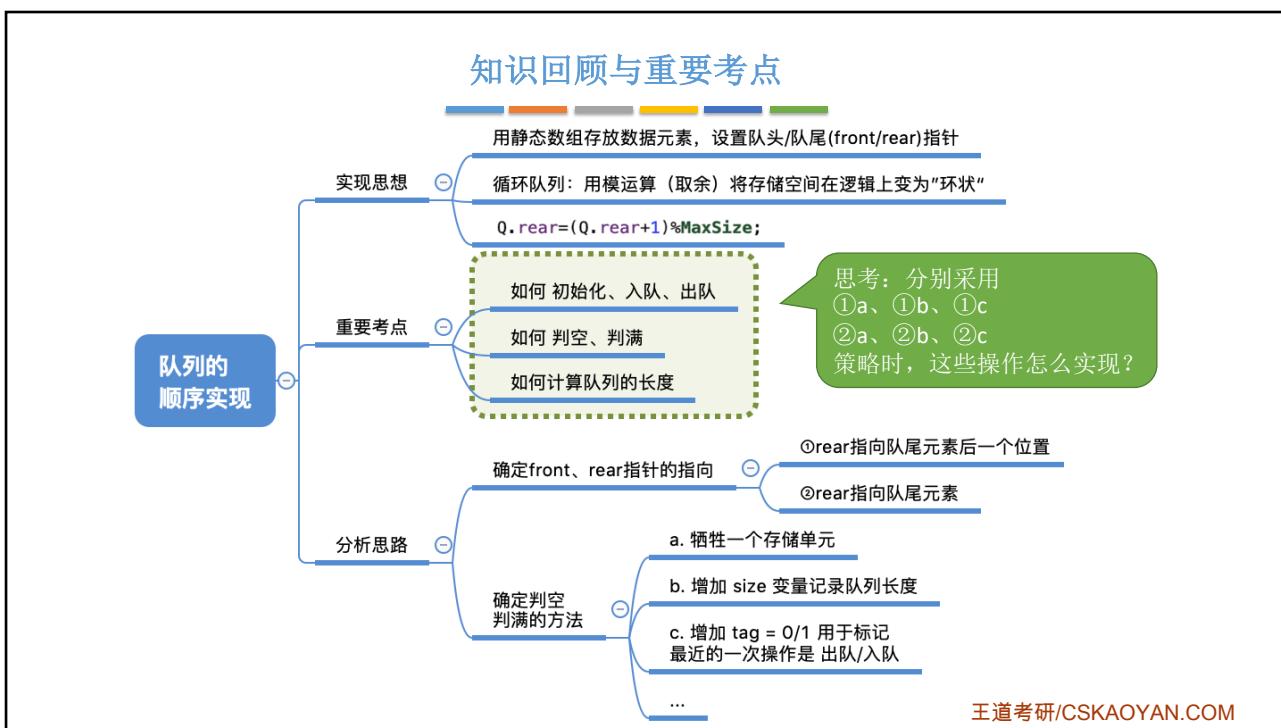
13



14



15



16