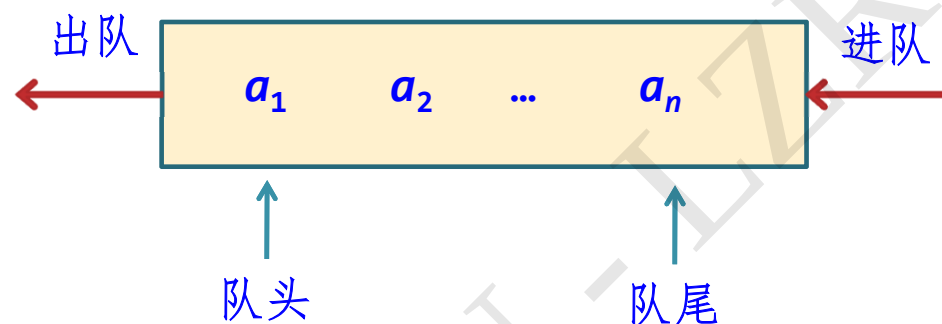


3.4 队 列

3.4.1 队列的基本概念

- 队列（简称队）也是一种运算受限的线性表，在这种线性表上，插入限定在表的某一端进行，删除限定在表的另一端进行。
- 队列的插入操作称为进队，删除操作称为出队。
- 允许插入的一端称为队尾，允许删除的一端称为队头。
- 新插入的元素只能添加到队尾，被删除的只能是排在队头的元素。

队列的模型示意图



- 是一种先进先出(First In First Out , 简称**FIFO**)的线性表。

队列的基本运算

- 初始化队列 **InitQueue(Q)**。建立一个空队Q。
- 销毁队 **DestroyQueue(Q)**。释放队列Q占用的内存空间。
- 进队 **EnQueue(Q, x)**。将x插入到队列Q的队尾。
- 出队 **DeQueue(Q, x)**。将队列Q的队头元素出队并赋给x。
- 取队头元素 **GetHead(Q, x)**。取出队列Q的队头元素并赋给x，但该元素不出队。
- 判断队空 **QueueEmpty(Q)**。判断队列Q是否为空。

【示例-1】以下属于队列的基本运算的是（ ）。

- A. 对队列中的元素排序
- B. 取出最近进队的元素
- C. 在队列中某元素之前插入元素
- D. 删除队头元素

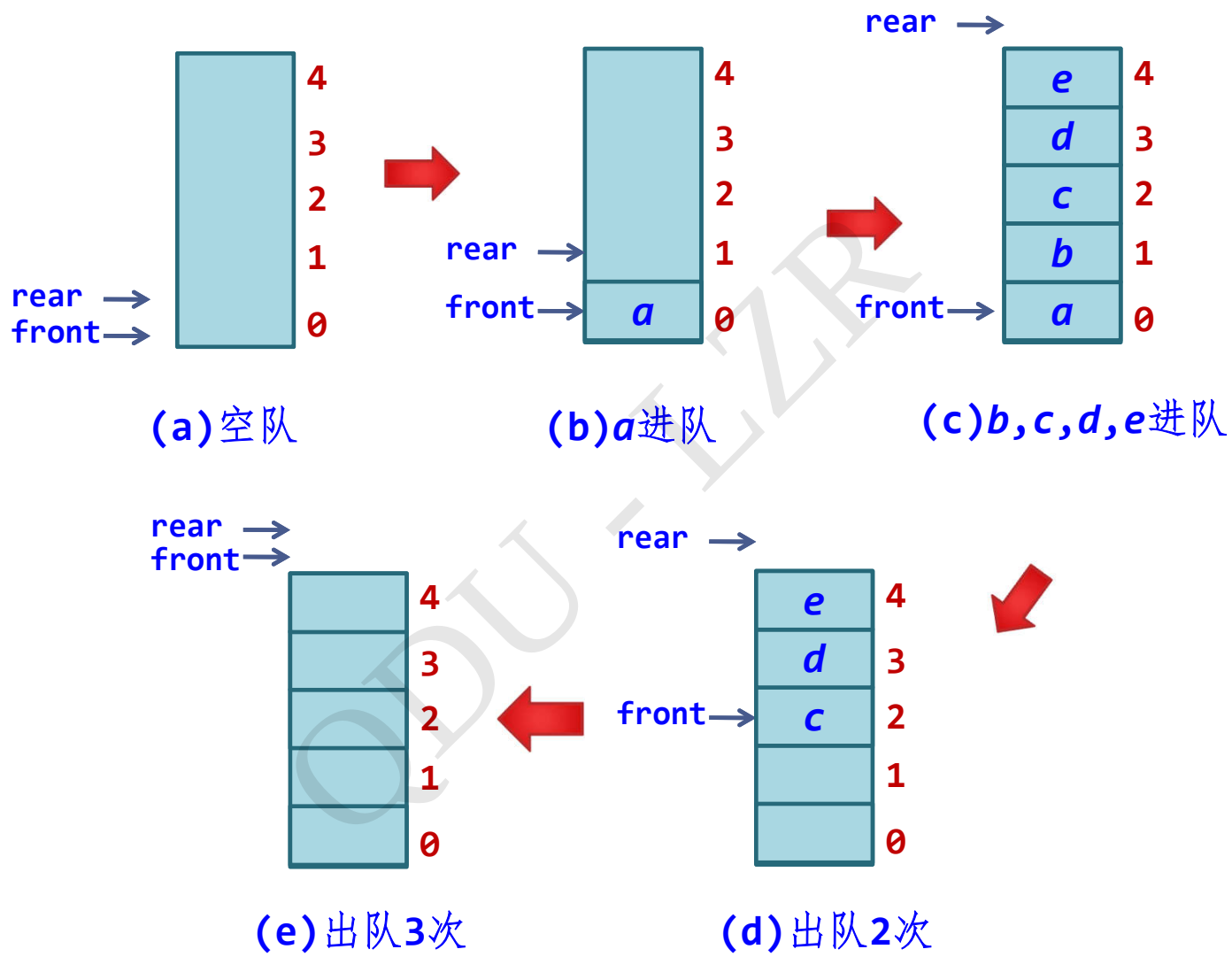
选择的答案是：D

3.4.2 循环队列 — 队列的顺序表示和实现

- 队列通常有两种存储结构，即顺序存储结构和链式存储结构。
- 队列的顺序存储结构简称为顺序队列，它由一个一维数组（用于存储队列中元素）及两个分别指示队头和队尾的变量组成，这两个变量分别称为“队头指针”和“队尾指针”。
- 通常约定：初始化建空队列时，令 $\text{front}=\text{rear}=0$ 。在非空队列中，头指针始终指向队列头元素，而尾指针始终指向rear队列尾元素的下一个位置。

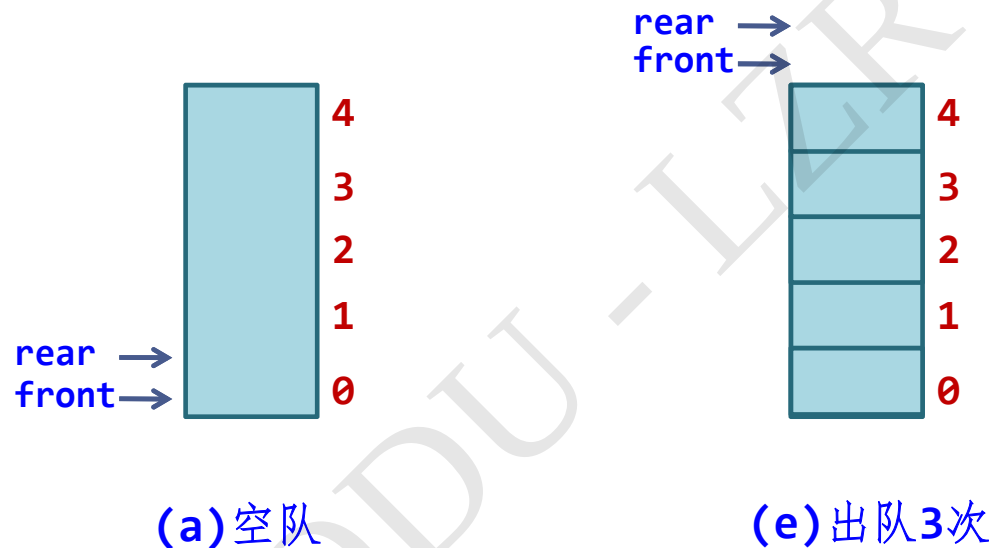
◆ 顺序队列的类型声明如下：

```
//-----队列的静态存储结构-----  
#define MaxSize 20          //指定队列的容量  
typedef struct  
{  
    ElemType data[MaxSize]; //保存队中元素  
    int front, rear;        //队头和队尾指针  
} SqQueue;
```



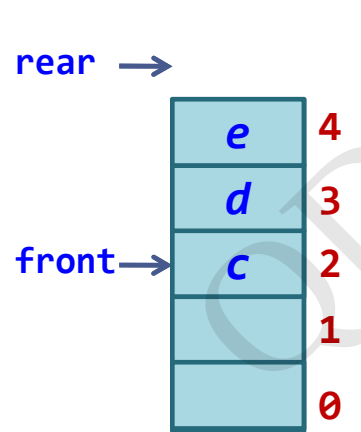
顺序队的几种状态

从中可以看到，图 (a)和(e)都是队空的情况，均满足 $\text{front}==\text{rear}$ 的条件，所以可以将 $\text{front}==\text{rear}$ 作为队空的条件。

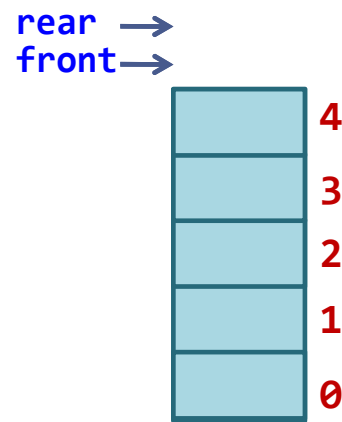


那么队满的条件如何设置呢？受顺序栈的启发，似乎很容易得到队满的条件为 $\text{rear}==\text{MAXSIZE}$ 。？

- 显然这里有问题，因为图 (d)和(e)都满足这个“队满”的条件，而实际上队列并没有满。
- 这种因为队满条件设置不合理而导致的“溢出”称为**假溢出**，也就是说这种“溢出”并不是真正的溢出，尽管队满条件成立了，但队列中还有多个存放元素的空位置。

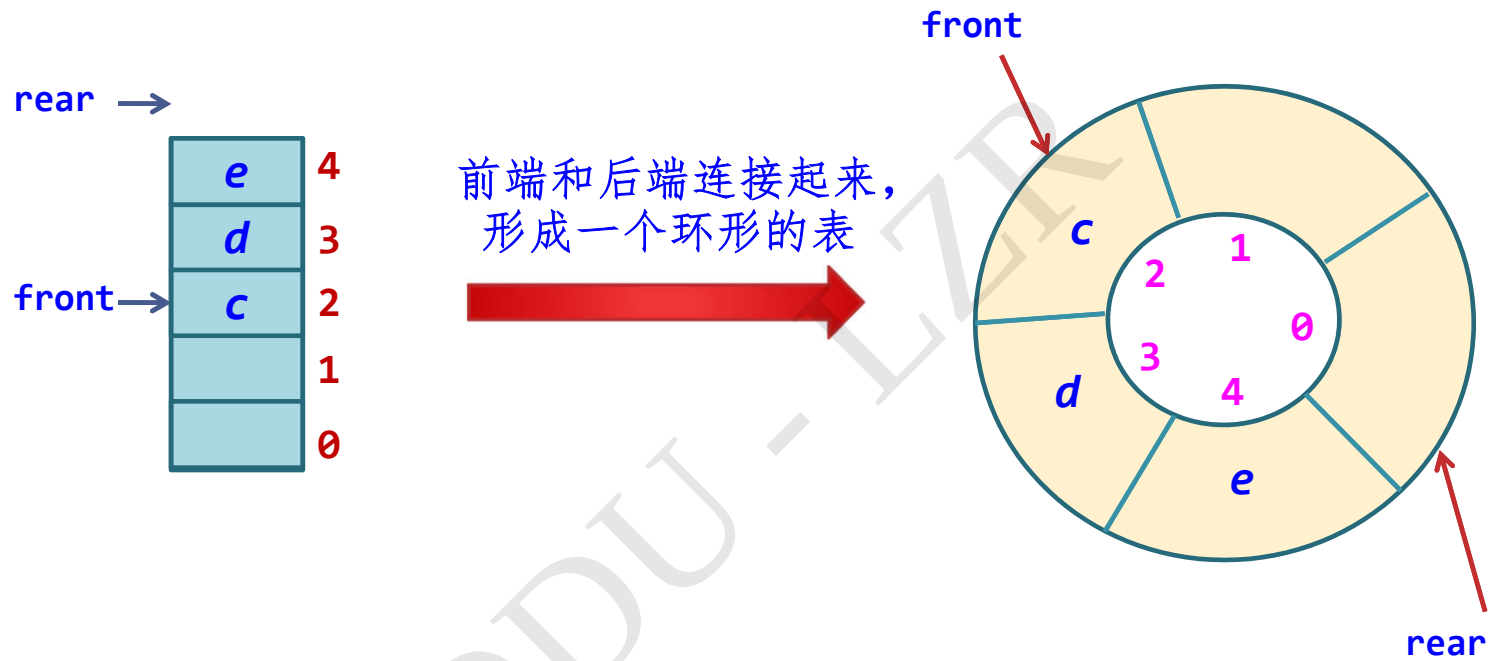


(d) 出队2次



(e) 出队3次

- 为了能够充分地使用数组中的存储空间，可以把数组的前端和后端“**假想地**”连接起来，形成一个环形的表，即把存储队列元素的表从逻辑上看成一个环。
- 这个环形的表叫做**循环队列**或**环形队列**。



队头队尾指针加1的操作为：

- 队头指针进1: $\text{front} = (\text{front} + 1) \text{ MOD MAXSIZE}$
- 队尾指针进1: $\text{rear} = (\text{rear} + 1) \text{ MOD MAXSIZE}$

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/936042052213010100>