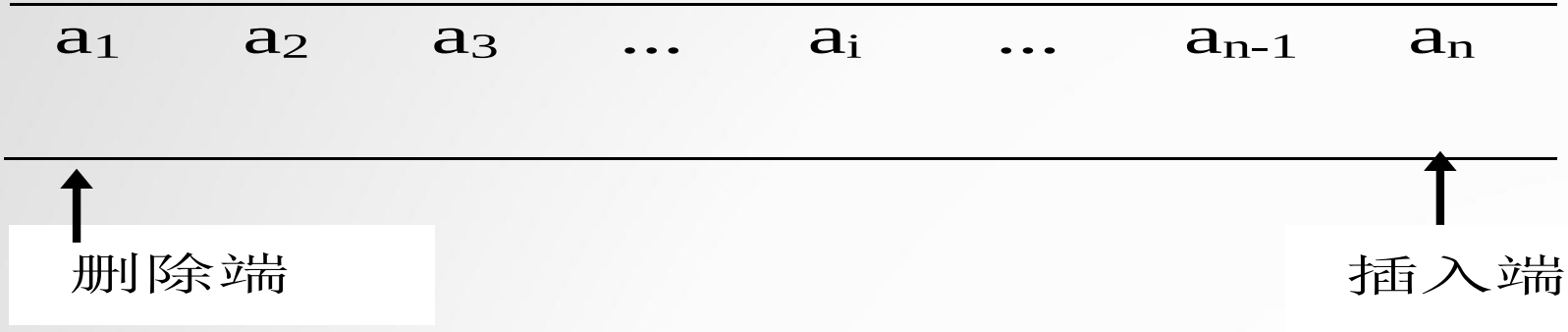

线性表
栈
队列

什么是队列？

教材P83：

队列也是一种特殊的线性表。其特殊性在于限定插入在线性表的一端（即表尾）进行，删除在线性表的另外一端（即表头）进行。如下所示：



队尾（rear）：允许插入的一端
队头（front）：允许删除的一端

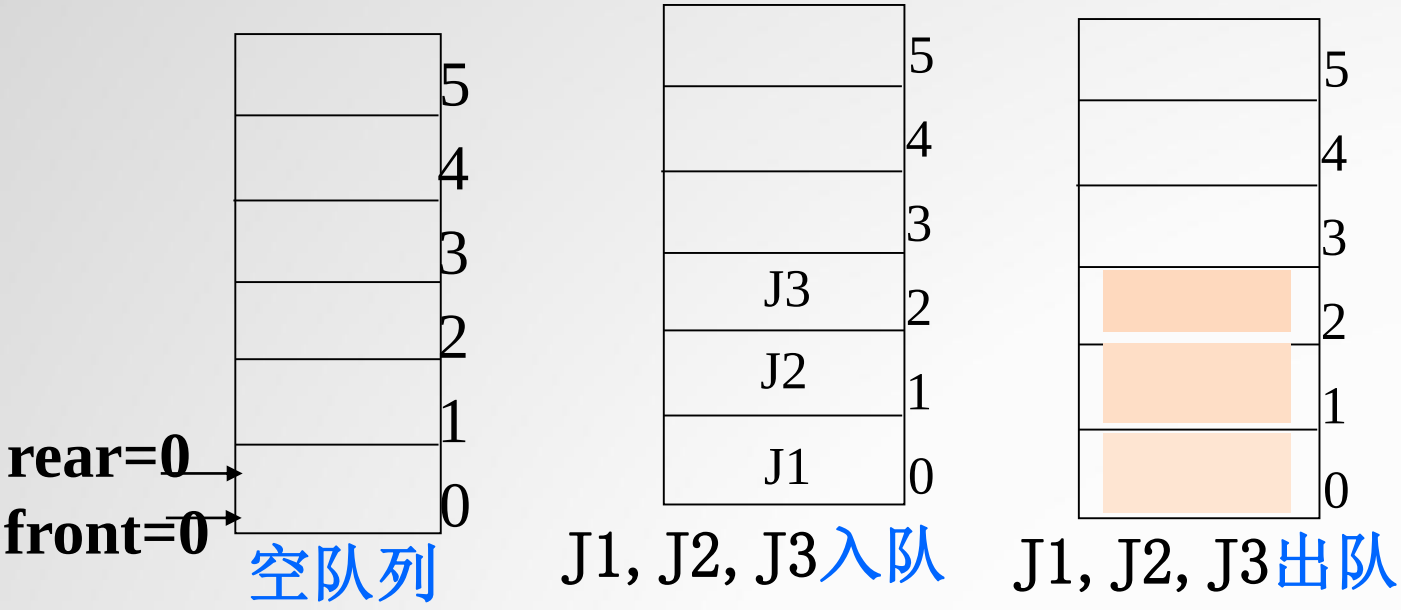
特点：先进先出（FIFO）

【Project 6】银行业务队列简单模拟

设某银行有A、B两个业务窗口，且处理业务的速度不一样，其中A窗口处理速度是B窗口的两倍—即当A窗口处理完2个顾客时，B窗口处理完1个顾客。给定到达银行的顾客序列，请按业务完成的顺序输出顾客序列。假定不考虑顾客先后到达的时间间隔，并且当不同窗口同时处理完2个顾客时，A窗口顾客优先输出。

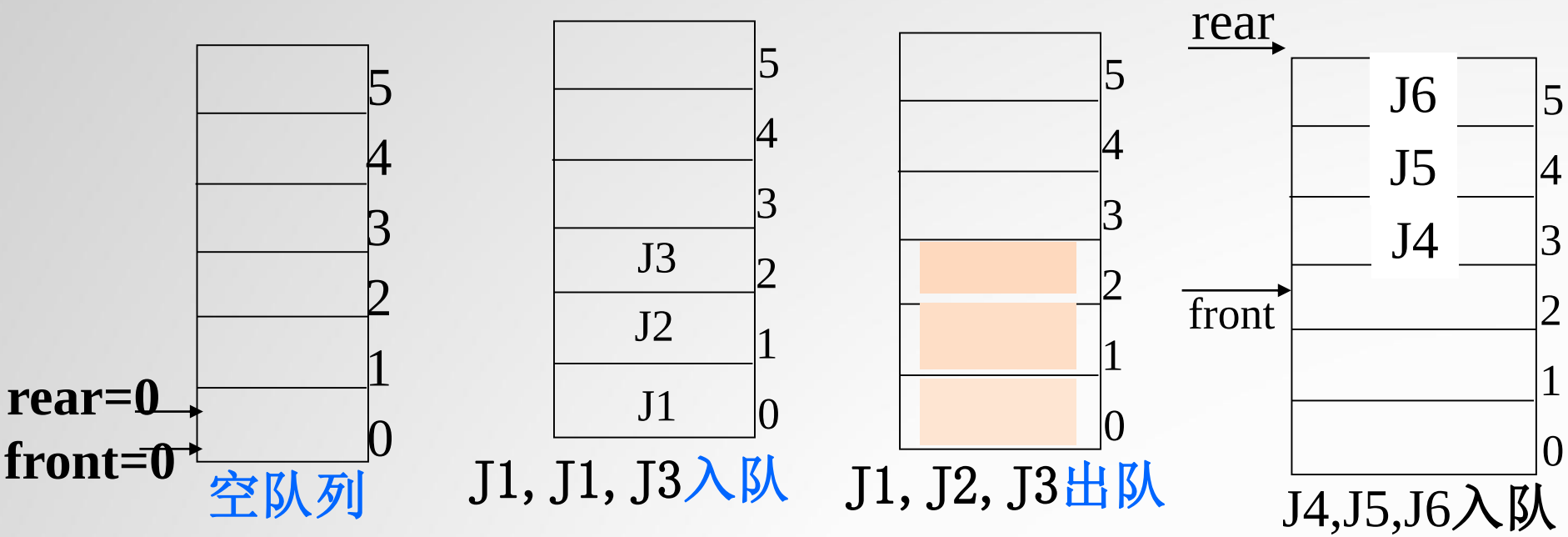
(1)队列的顺序存储结构（教材P85）

```
#define MAXSIZE 100
typedef struct
{
   ElemType elem[MAXSIZE];
   int front;  //队头，若队列不空“指向”队列头元素
   int rear; //队尾，若队列不空“指向”队列尾元素的下一个位置
}SqQueue;
```



【Question】如果空队列开始时front和rear值都是0，当插入4个元素并删除2个元素后，front和rear值分别是多少?

2，4



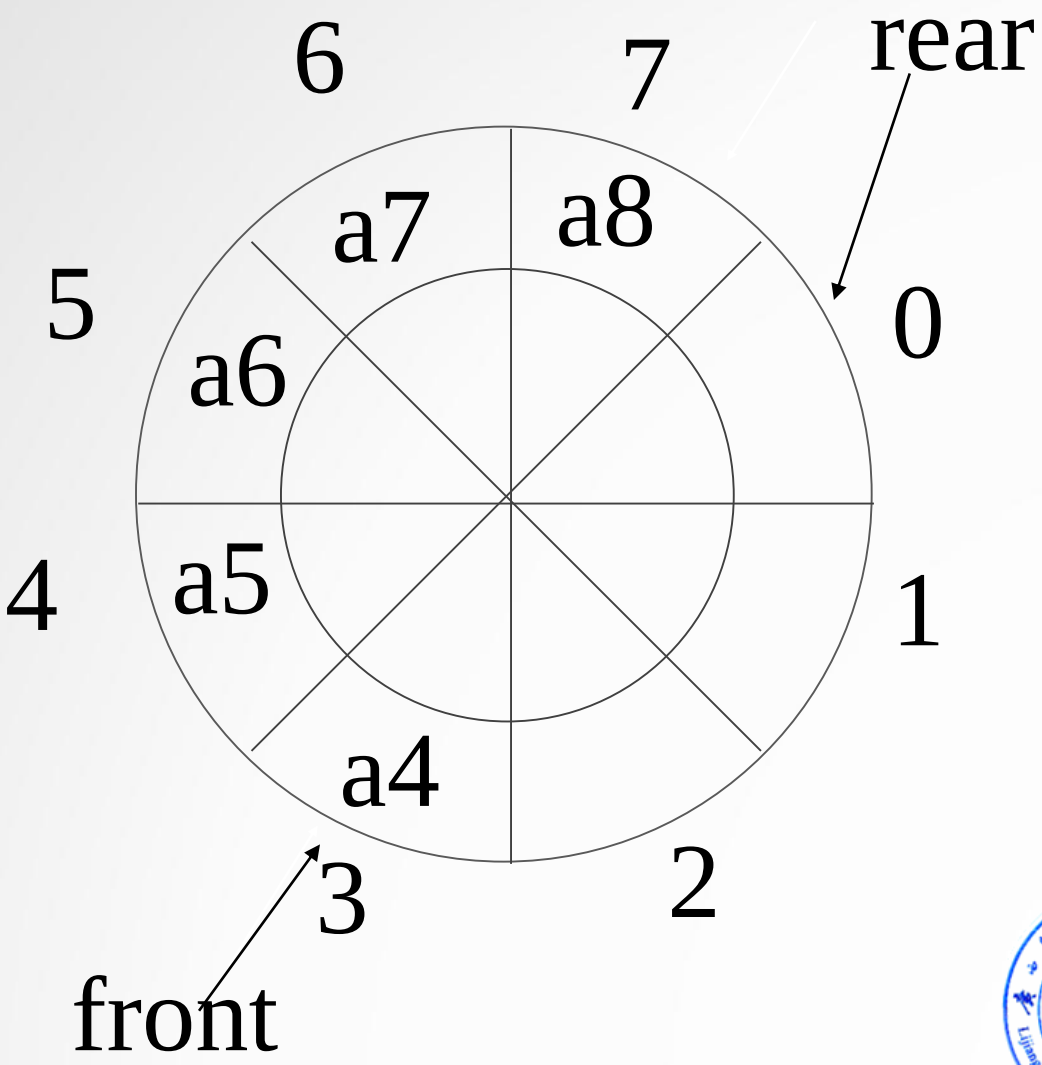
问题：

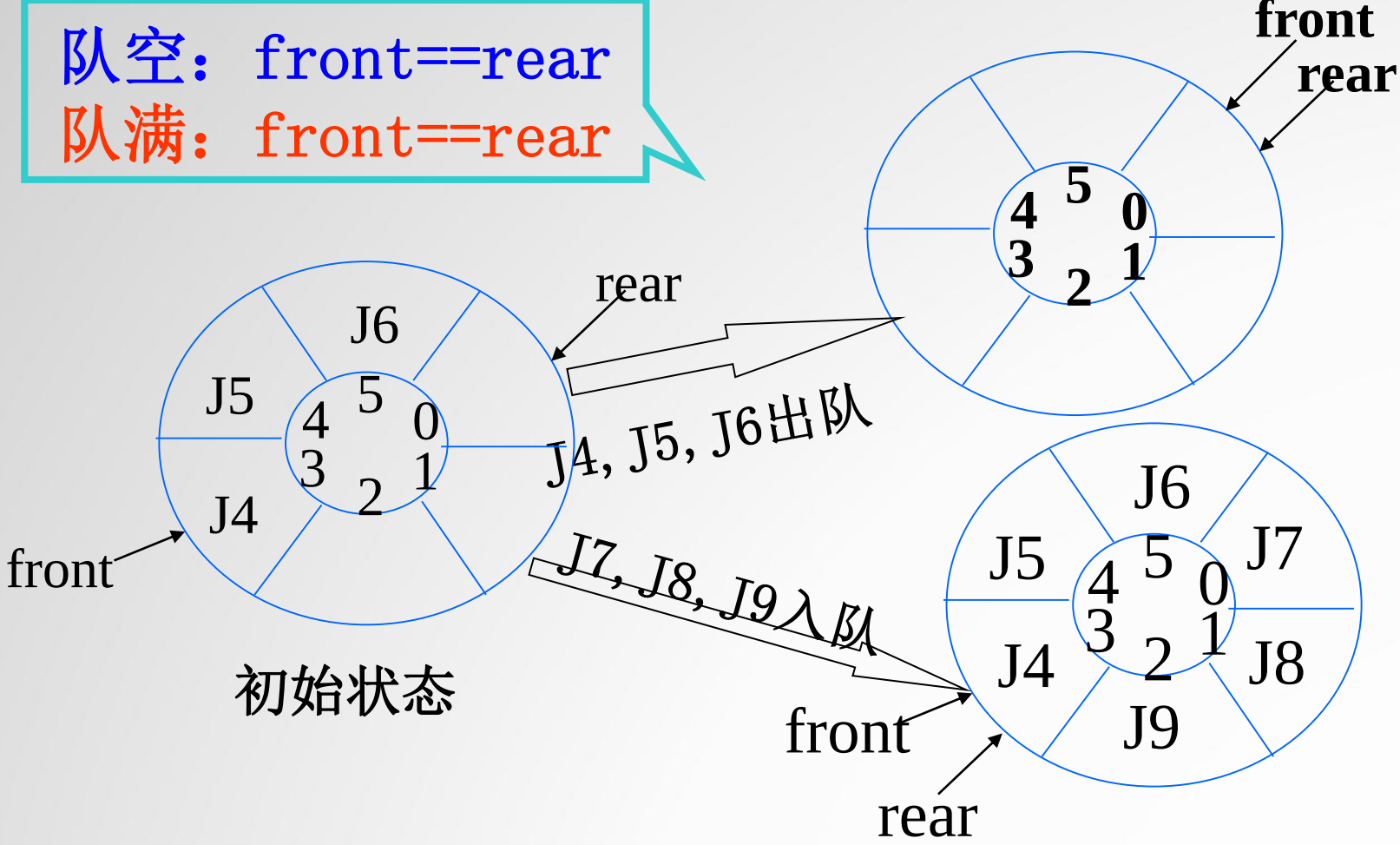
当front≠0,rear=M时再有元素入队发生溢出——假溢出

解决方法：循环队列

循环队列

将存储队列元素的一维数组首尾相接，形成一个环状。





问题：当**front==rear**时，如何判断此时的队列是满还是空？

解决方案：少用一个元素空间：

队空：front==rear　　队满：(rear+1) % MAXSIZE==front

--循环队列的基本操作算法

（1）初始化队列Q

```
void initQueue(SqQueue *Q)
{
      Q->front=0;
      Q->rear=0;
 }
```

（2）求循环队列Q的长度

```
int getSqLength(SqQueue Q)
{
  return (Q.rear-Q.front+MAXSIZE)%MAXSIZE;
 }
```

（3）入队(*Q,e)

```
void enQueue (SqQueue *Q, ElemType e)
{
  if ((Q->rear+1) % MAXSIZE ==Q->front)//队列满
     return ;
  Q->elem[Q->rear] = e; //元素e赋给队尾
  Q->rear = (Q->rear+1) % MAXSIZE;
}
```

（4）出队(*Q,*e)

```
void deQueue (SqQueue *Q, ElemType *e)
{
 if (Q->front ==Q->rear) //队列空
     return ;
 *e=Q->elem[Q->front];  //将队头元素赋给e
 Q->front = (Q->front+1) % MAXSIZE;
}
```