顺序队列内部采用的是一维数组进行存储。

因为出队只能在队首进行，入队只能在队尾进行，所以需要两个指针一个指向队首、一个指向队尾。

顺序队列的内部结构为：一维数组+队首指针+队尾指针

Object[] queueElem;

int front; //指向当前队首的指针

int rear; //指向当前队尾的指针

队列初始化的时候

front=rear=0；

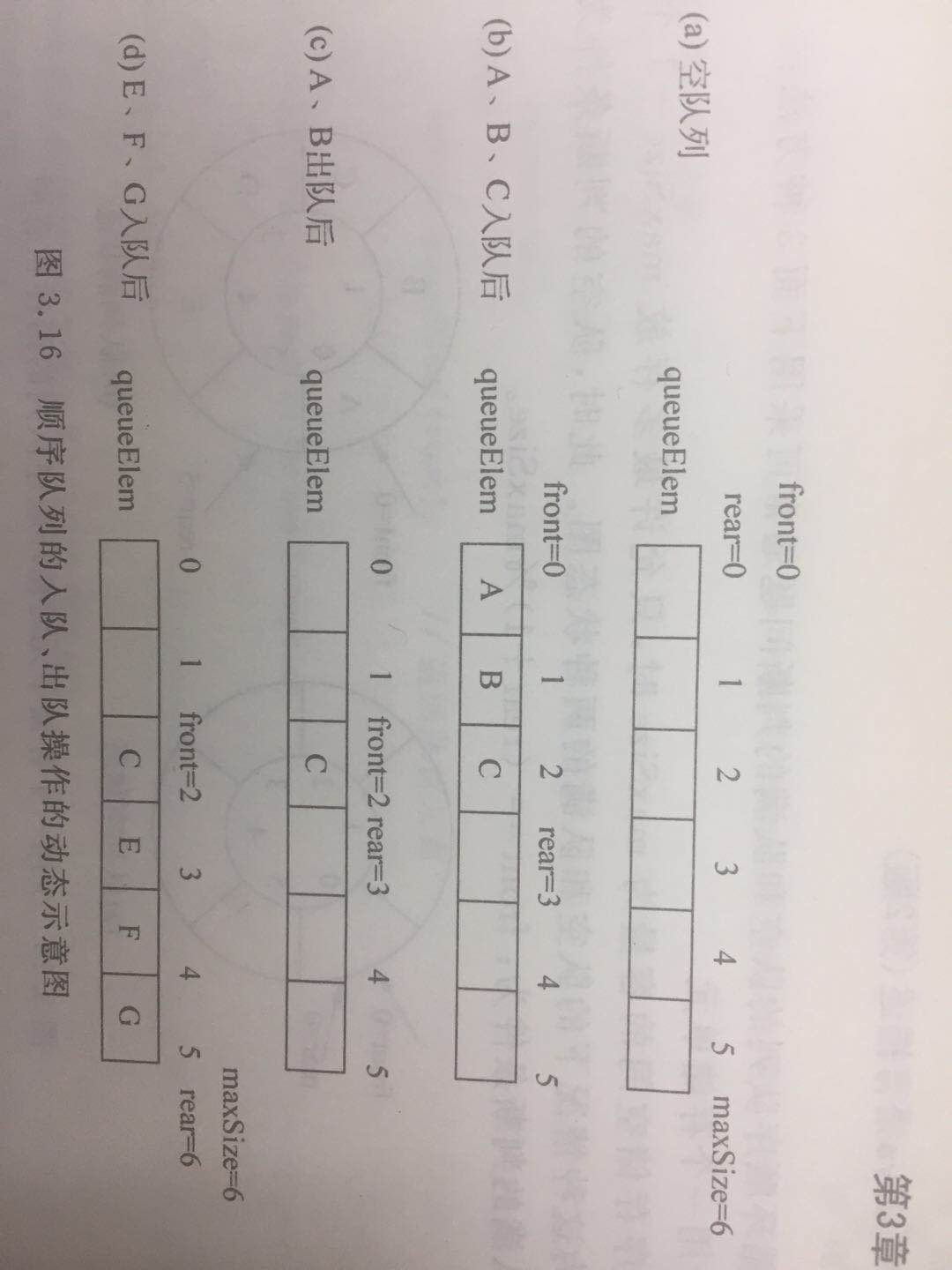
入队

在rear指向的单元存入数据，然后rear=rear+1

出队

取出front指向的单元数据，然后front=front+1

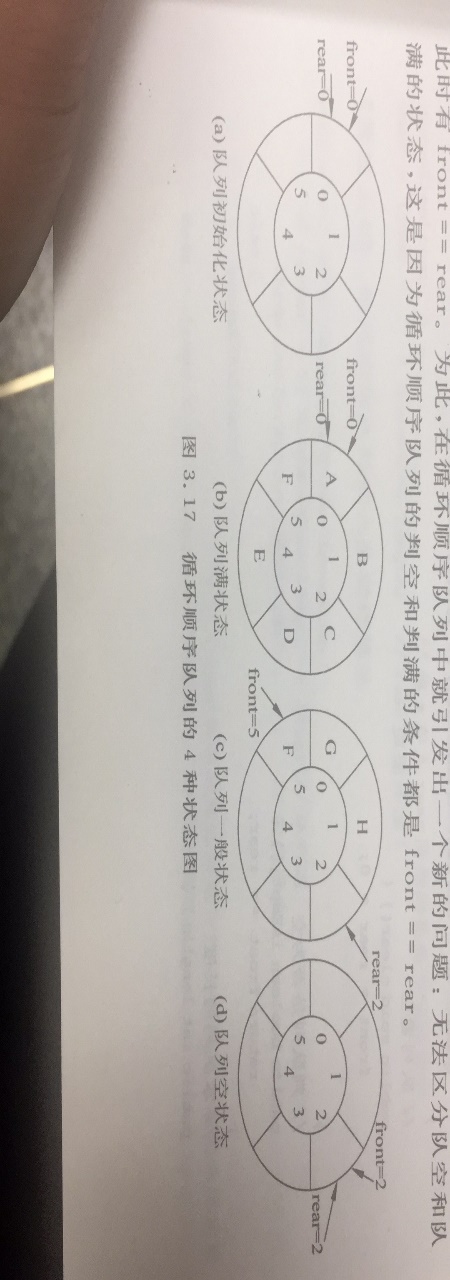
示例



如图所示，当d后，我们还想放入H入队，在队尾已经没有地方可以放入了。但是队头还有两个位置没有存入数据，这种提示数组溢出，但不是真正的空间不够的溢出，这种溢出叫做“假溢出“。

要解决假溢出的问题，最好的办法就是把顺序队列所使用的空间看作是一种逻辑上的首尾相连的循环队列。当front和rear指针到达maxSize-1的位置时候，再+1，就自动到了0.

例如上图当rear指针指到了maxSize=6-1=5的位置的时候，我们存入G元素后rear=rear+1后，rear指向0.这样就不会出现假溢出现象了。那么如何做到呢？每次增加后求与maxSize的余数就能准确找到逻辑上的循环位置，也就是rear=（rear+1）%maxSize。



如图所示（a）（b）所示，对空的时候front=rear=0，队满的时候也是front=rear=0。我们如何判断队空还是队满呢？

1.我们可以在队尾少使用一个单元，

队空：front==rear

队满；front==(rear+1)%maxSize

2.设置一个出入队的标志位

在内部结构中加一个标志位flag

初始flag=0，每次入队flag=+1，每次出队flag=-1.

队空：front==rear&&（flag==0||flag==-1）

队满：front==rear&&flag=+1

3.设置一个计数器

在内部结构加入一个计数器counter

初始为0，每次入队+1，出队-1。

队空：counter==0

队满：counter>0&&front==rear

我采用的是第二种方法：

功能实现：

1.public void clear() { //清空队列

front=0;

rear=0;

flag=0;

}

2. public boolean isEmpty() { //当入队状态下的front==rear，为队列满

return (front==rear)&&(flag==0||flag==-1);

}

3. public int length() { //因为是循环队列

return (rear-front+queueElem.length)%queueElem.length;

}

4. public Object peek() { //取队首元素

return queueElem[front];

}

5. public void offer(Object x) { //先进队,队尾指针+1,因为当前的队尾指针指向的是空单元

if((rear)%queueElem.length==front&&flag==+1) //判断是否队满

System.out.println("队列已满");

else { //队不满才可以入队

queueElem[rear] = x;//先入队

rear = (rear + 1) % queueElem.length;//因为是循环队列，需要求余

flag = +1;//入队，修改当前的标志位

}

}

6. public Object poll() { //出队，直接出队，然后修改队头指针

if(this.isEmpty()){

System.out.println("队列为空");

return null;

}

else {

Object temp = queueElem[front];

front=(front+1)%queueElem.length; //直接修改指针,不用真正意义上清空

flag=-1;//修改进入队标志位

return temp;//返回刚才出队的元素

}

}

7. public void display(){ //遍历循环队列

for(int i=front;i<=rear;i=(i+1)%queueElem.length)

System.out.print(queueElem[i]+" ");

System.out.println();

}