2.3 Queues

Main contents

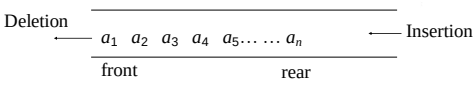
- Definition and operations
- Implementation
- Applications

2.3.1 Definition

A queue is a list. With a queues, insertions is done at one end (known ad rear) whereas deletion is performed at the other end (known as front) .

–First in first out(FIFO)

- 队列：只允许在一端进行插入操作，而另一端进行删除操作的线性表。
- 空队列：不含任何数据元素的队列。
- 队尾和队首：允许插入（也称入队、进队）的一端称为队尾，允许删除（也称出队）的一端称为队首。



Operations

- 初始化：InitQueue
- 判断是否为空：IsEmpty
- 入队列：EnQueue
- 出队列：DlQueue
- 取队列头：GetHead
- 清空队列：Clear
- 得到队列长度：GetSize

ADT of queue

```
template <class T>  class Queue
{  // 队列的元素类型为T，它们是按先后次序的
   //线性表结构
   // 一般使用front和rear指示队列的前端和尾端
   // 用curr_len 存储当时的队列长度
   //栈的运算集为：
   Queue(int s);   //创建队列实例，最大长度为s
   ~ Queue();      //该实例消亡,释放全部空间
```

```
void  EnQueue(T  item);    //item进入队列前端
//返回队列的前端元素内容，并从队列删去T
DeQueue();
 //返回队列的前端元素内容，但不从尾部删去T
GetFirst();
void MakeEmpty();    //变为空队列
int  IsEmpty();       //返回真，若队列已空
int  IsFull();          //返回真，若队列已满
 };
```

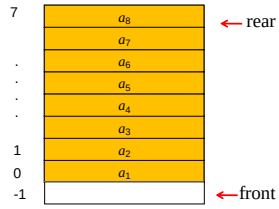
2.3.2 Implementation

✓ 存储方式：连续设计

队头指针总是指向队头元素的前一个位置。

初始：front=-1;rear=-1。

存储空间0..*m*-1



Basic operations

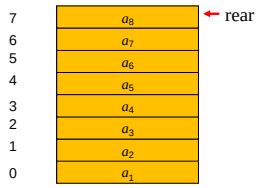
入队：rear=rear+1

出队：front=front+1

队空：rear=front

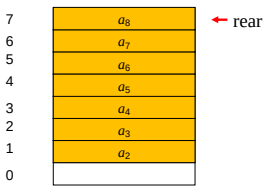
队满：rear-front=*m*

队列溢出现象



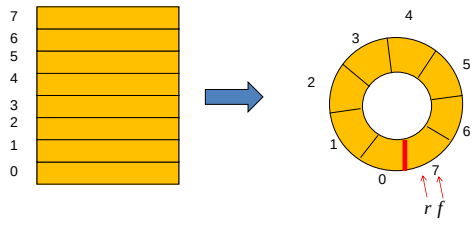
全部空间都已经使用，不能再加入元素，真正溢出。

队列溢出现象



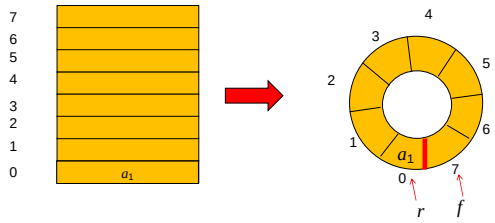
由于队尾指针已经指向最后位置，不能再加入元素。但是前面还有空余，假溢出。

Circular queue循环队列



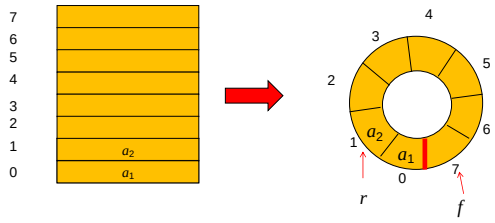
初始：rear=front

Circular queue



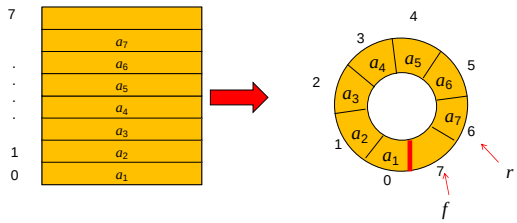
入队：rear=rear+1

Circular queue



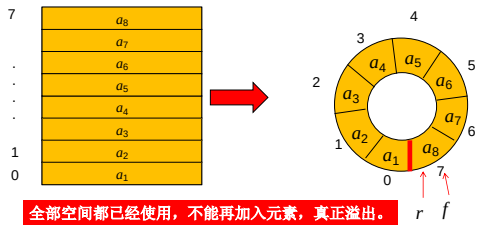
入队：rear=rear+1

Circular queue



入队：rear=rear+1

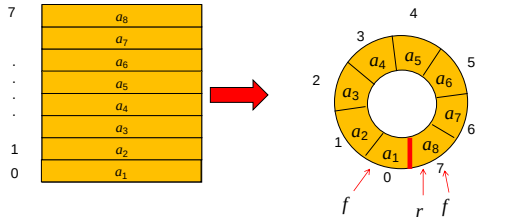
Circular queue



全部空间都已经使用，不能再加入元素，真正溢出。

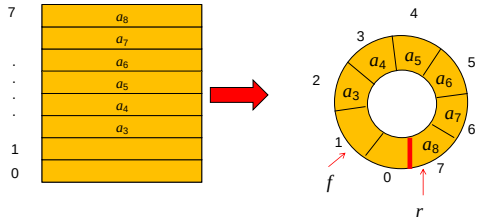
入队：rear=rear+1

Circular queue



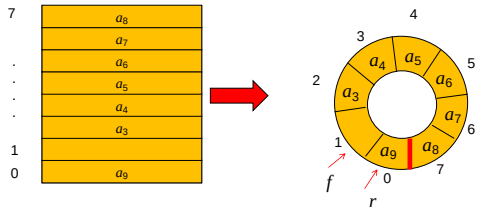
出队：front=front+1

Circular queue



出队：front=front+1

Circular queue



入队：rear=rear+1 ？

构建循环队列时，要解决：

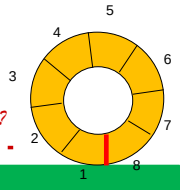
队空：rear=front　　队满：rear=front

- 方法一：增设一个存储队列中元素个数的计数器count，当front==rear 且count==0时，队空；当front==rear 且count==MaxSize时，队满；
- 方法二：设置标志flag，当front==rear且flag==0时为队空；当front==rear且flag==1时为队满。
- 方法三：保留队空的判定条件：front==rear；把队满判定条件修改为：（(front+1)%MaxSize==rear）。
  - 代价：浪费一个元素空间，队满时数组中有一个空闲单元；

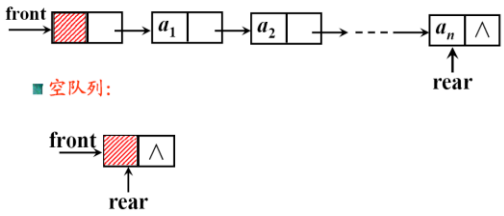
Circular queue

入队：rear=(rear+1)%m
出队：front=(front+1)%m
队空：rear=front
队满：(rear+1)%m=front

存储空间为1..m，上述的操作又如何？



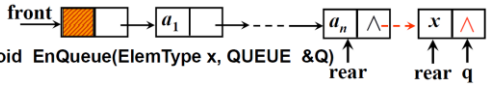
✓ 存储方式：链接设计



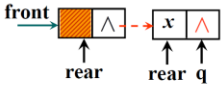
■ 空队列：

◆ 队列的链接存储结构及实现

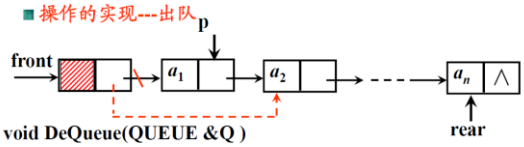
■ 操作的实现----入队



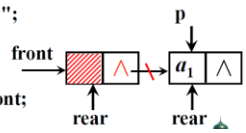

```
③ void  EnQueue(ElemType x, QUEUE  &Q)
{
    q=new cwlltype;
    q->data=x ;
    q->next=NULL;
    Q.rear->next=q;
    Q.rear=q;
}
```



■ 操作的实现---出队




```
void DeQueue(QUEUE &Q )
{  if (Q.rear==Q.front) cout<<"队空";
   p=Q.front->next;
   Q.front->next=p->next;
   if (p->next==NULL) Q.rear=Q.front;
   delete p;
}
```



2.3.3 离散事件的模拟

银行有四个窗口对外服务，从开门起不断有客户进入银行。每个窗口在某一时刻只能接待一位客户，因此在客户人数众多时，需要在每个窗口顺序排队。对于刚进入的客户，如果某个窗口正在空闲，则可上前办理业务，否则，排在人数最少的队列后面等待。编制程序，模拟银行的业务活动，并计算客户的平均逗留时间。

- 离散事件模拟
- 问题：
  - 一个银行，有N个窗口；
  - 每分钟来一个客户，客户业务处理时间为一个随机数M；
  - 每个客户总是排到最短的队上。
- 要求：
  - 模拟一段时间内的排队情况，并进行定量统计（平均逗留时间）

  若只有一个队列时，平均逗留时间又是多少？

分　析

事件：客户到达银行和离开银行时发生的事情。

　　事件的类型，事件发生的时刻。

事件的发生：

　　到达事件：客户的到来时形成（0）。

　　离开事件：由客户服务时间和等待时间决定。（1..4）

　　建立事件链表，记录模拟过程中发生的事件。按照事件发生的时刻的先后次序存储。

分　析

设立四个队列，存储客户到达的时刻和服务所需要的时间。队头元素为窗口正在服务的客户。每个队头客户都存在一个将要离开的事件。队列的结构如下：

到达的时刻	需要服务的时间
ArrivalTime	Duration

任意时刻发生的事件，事件结点结构如下：

新客户的到来

1号窗口客户离开

2号窗口客户离开

3号窗口客户离开

4号窗口客户离开

到达0
离开1,2,3,4

事件发生的时刻	事件类型

ev：事件链表；记录将要发生的事件（到达/离开）

　　如果是到达的类型，则找个队进行排队等待，

　　如果是离开的事件，则删除对应队列中的元素。

　　仿真器总是从事件链表中获得事件进行处理。

　　结构：事件发生时间(OccurTime)，事件类型(Ntype)。

en:　事件结点；记录要处理的事件信息，

　　结构：事件发生时间(OccurTime)，事件类型(Ntype) 。

q[i]：　队列；　客户排队等待

　　结构：到达时间(ArrivalTime)，服务时间(Duration)。

```
Void BankSimulation()
 {
   OpenForday();
   While ev非空/事件链表中有待处理的事件
    {
      DelList(ev, en);/获得要处理的事件
      if (en.NType==0) /处理事件
        CustomerArrived();
      else CustomerDeparture();
    }
   计算平均逗留时间；
 }
```

```
Void OpenForday()
{
  Totaltime=0;
   CustomerNum=0;
   InitList(ev);    //事件链表置空
   for (i=1;i<=4;i++)
     InitQueue(q[i]);  //队列置空
   en.OccurTime=0;
   en.NType=0;          //设定第一个事件（客户到达事件）；
   InsertList(ev,en)   //插入事件表中
  }
```

```
Void CustomerArrived()  //客户到达处理模块
{ /* en.NType=0，（处理当前事件，产生新的到达事件）
   CustomerNum++;
   Random(Durtime, intertime)  //（需要服务时间, 下一到达时刻间隔 ）
   t=en.OccurTime+intertime;    //下一个客户到达的时刻；
   if (t<Closetime)
        InsertList(ev, (t,0));      //插入到事件链表，
                                    表示将在t时刻有新客户到达
    i=Minimum(q)；  //为当前事件查找最短的队列
    Enqueue(q[i], (en.OccurTime, Durtime));  // 入队
    if (QueueLength(q[i])==1)
      InsertList(ev, (en.Occurtime+Durtime,i));
      //产生离开事件，插入到事件链表
}
```

```
Void CustomerDeparture()  //客户离开事件的处理模块
 {
   i=en.NType;
   DelQueue(q[i], Customer); //删除第i队头元素，
                                值存储在Customer中
   TotalTime=TotalTime+en.OccurTime-Customer.ArrivalTime;
   if (!QueueEmpty(q[i])
    //将第i个队列的队头元素作为离开事件插入到事件表
      {
        GetHead(q[i],Customer);
        InsertList(ev, (en.OccurTime+Customer.Durtation, i));
 }
```

仿真示例

- 初始状态

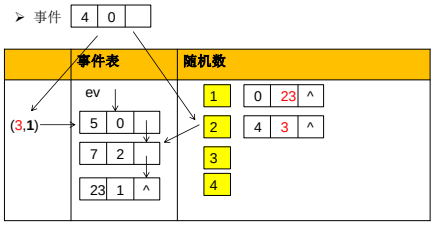
随机数	事件表	队列状态
	ev → 0 0 ^	1 2 3 4

仿真示例

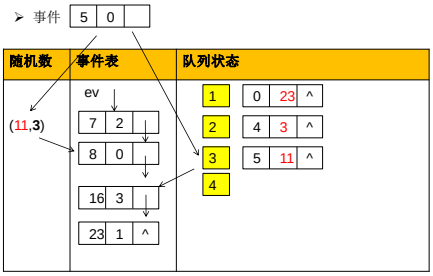
- 事件 0 0

随机数	事件表	队列状态
(23,4) →	ev → 4 0 23 1 ^	1 → 0 23 ^ 2 3 4

仿真示例



仿真示例



2.3.4 队列应用：划分子集问题

问题描述：已知集合$A=\{a_1,a_2,.....a_n\}$，及集合上的关系$R=\{(a_i,a_j) \mid a_i,a_j \in A, i \neq j\}$，其中$(a_i,a_j)$表示$a_i$与$a_j$间存在冲突关系。要求将$A$划分成互不相交的子集$A_1,A_2,.....A_k,(k \leq n)$，使任何子集中的元素均无冲突关系，同时要求划分子集个数尽可能少。

例 A={1,2,3,4,5,6,7,8,9}
R={ (2,8), (9,4), (2,9), (2,1), (2,5), (6,2), (5,9), (5,6), (5,4), (7,5), (7,6), (3,7), (6,3) }

子集划分为：
A1={ 1,3,4,8 }　A2={ 2,7 }　A3={ 5 }　A4={ 6,9 }

- 算法基本思想：利用循环筛选。从第一个元素开始，凡与第一个元素无冲突的元素划归一组；再将剩下的元素重新找出互不冲突的划归第二组；直到所有元素进组。
- 所用数据结构
  - ✓ 冲突关系矩阵
    r[i][j]=1，i,j有冲突
    r[i][j]=0，i,j无冲突
  - ✓ 循环队列cq[n]
    数组result[n]存放每个元素分组号
    工作数组newr[n]

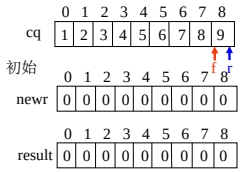
- 工作过程
  - 初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
  - 第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”,下一个元素出队
    - » 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
    - » 若其在newr中对应位置为“0”,无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
  - 如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
  - 令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束

A={1,2,3,4,5,6,7,8,9}

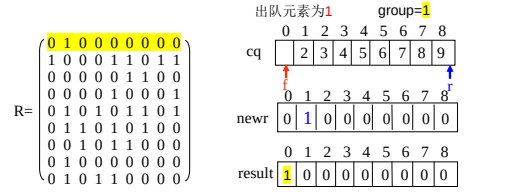
R={ (2,8), (9,4), (2,9), (2,1), (2,5), (6,2), (5,9), (5,6), (5,4), (7,5), (7,6), (3,7), (6,3) }

初始状态：A中元素放于cq中，result和newr数组清零，组号group=1

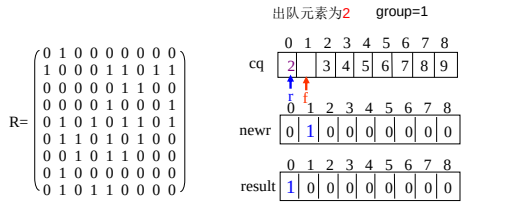
$$R=\begin{pmatrix}0&1&0&0&0&0&0&0&0\\1&0&0&0&1&1&0&1&1\\0&0&0&0&0&1&1&0&0\\0&0&0&0&1&0&0&0&1\\0&1&0&1&0&1&1&0&1\\0&1&1&0&1&0&1&0&0\\0&0&1&0&1&1&0&0&0\\0&1&0&0&0&0&0&0&0\\0&1&0&1&1&0&0&0&0\end{pmatrix}$$



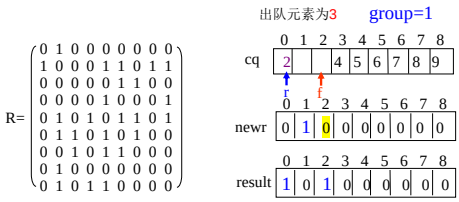
初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束



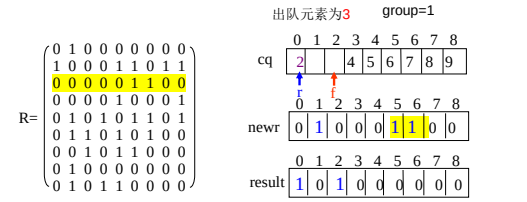
初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束



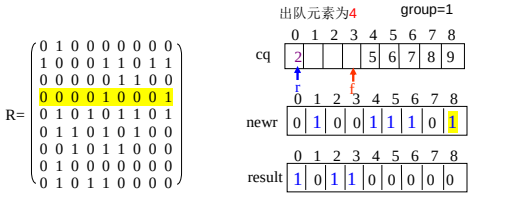
初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束



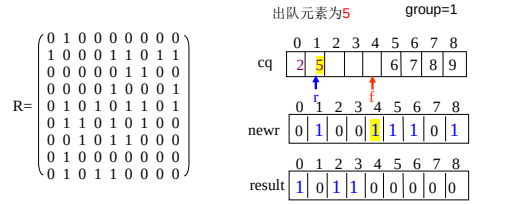
初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束



初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束



初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束



初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束

出队元素为6　group=1

R=
0 1 0 0 0 0 0 0 0
1 0 0 0 1 1 0 1 1
0 0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 1
0 1 0 1 0 1 1 0 1
0 1 1 0 1 0 1 0 0
0 0 1 0 1 1 0 0 0
0 1 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0

cq

0	1	2	3	4	5	6	7	8
2	5	6				7	8	9

r　f

newr

0	1	2	3	4	5	6	7	8
0	1	0	0	1	1	1	0	1

result

0	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0

初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束

出队元素为7　group=1

R=
0 1 0 0 0 0 0 0 0
1 0 0 0 1 1 0 1 1
0 0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 1
0 1 0 1 0 1 1 0 1
0 1 1 0 1 0 1 0 0
0 0 1 0 1 1 0 0 0
0 1 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0

cq

0	1	2	3	4	5	6	7	8
2	5	6	7				8	9

r　f

newr

0	1	2	3	4	5	6	7	8
0	1	0	0	1	1	1	0	1

result

0	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0

初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束

出队元素为8　group=1

R=
0 1 0 0 0 0 0 0 0
1 0 0 0 1 1 0 1 1
0 0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 1
0 1 0 1 0 1 1 0 1
0 1 1 0 1 0 1 0 0
0 0 1 0 1 1 0 0 0
0 1 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0

cq

0	1	2	3	4	5	6	7	8
2	5	6	7					9

r　f

newr

0	1	2	3	4	5	6	7	8
0	1	0	0	1	1	1	0	1

result

0	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	1	0

初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束

出队元素为9　group=1

R=
0 1 0 0 0 0 0 0 0
1 0 0 0 1 1 0 1 1
0 0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 1
0 1 0 1 0 1 1 0 1
0 1 1 0 1 0 1 0 0
0 0 1 0 1 1 0 0 0
0 1 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0

cq

0	1	2	3	4	5	6	7	8
2	5	6	7	9				

r　f

newr

0	1	2	3	4	5	6	7	8
0	1	0	0	1	1	1	0	1

result

0	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	1	0

初始状态：A中元素放于cq中，result和newr数组清零，组号group=1
第一个元素出队，将r矩阵中第一行“1”拷贝到newr中对应位置，这样，凡与第一个元素有冲突的元素在newr中对应位置处均为“1”，下一个元素出队
✓ 若其在newr中对应位置为“1”，有冲突，重新插入cq队尾，参加下一次分组
✓ 若其在newr中对应位置为“0”，无冲突，可划归本组；再将r矩阵中该元素对应行中的“1”拷贝到newr中
如此反复，直到9个元素依次出队，由newr中为“0”的单元对应的元素构成第1组,将组号group值“1”写入result对应单元中
令group=2,newr清零，对cq中元素重复上述操作，直到cq中front==rear,即队空，运算结束

出队元素为9　group=4

R=
0 1 0 0 0 0 0 0 0
1 0 0 0 1 1 0 1 1
0 0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0 1
0 1 0 1 0 1 1 0 1
0 1 1 0 1 0 1 0 0
0 0 1 0 1 1 0 0 0
0 1 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0

cq

0	1	2	3	4	5	6	7	8

f　r

newr

0	1	2	3	4	5	6	7	8
0	2	1	1	1	0	1	0	0

result

0	1	2	3	4	5	6	7	8
1	2	1	1	3	4	2	1	4

可行的子集划分为：
A1={ 1,3,4,8 }　A2={ 2,7 }　A3={ 5 }　A4={ 6,9 }

其它操作受限的线性表

输入受限的队列：限定在一端进行输入，可以在两端进行删除的队列。

输出受限的队列：限定在一端进行输出，可以在两端进行加入的队列。