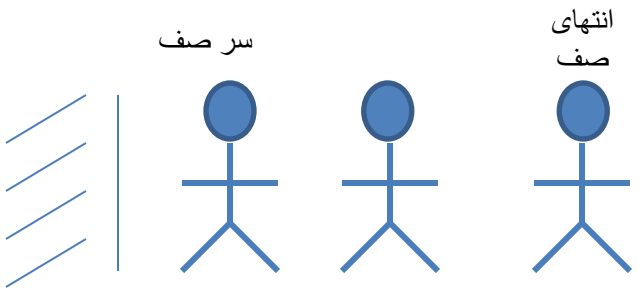
فایل چهارم درس ساختمان های داده

- فایل آماده شده برای جلسه معرفی ساختار صف و کاربردهای آن و صرفا برای کلاس آنلاین در این ترم می باشد.

همواره سلامت و موفق باشید- اسبقی

صف (QUEUE)

برای درک بهتر ساختار صف، مثال زیر را در نظر بگیرید:



ساختار صف با رفتار(FIFO)(FIRST IN FIRST OUT): خروج به ترتیب ورود

آیا این ساختار برای پیاده سازی دسترسی های عادلانه مناسب است؟

صف (QUEUE)

صف بعنوان یک ADT:

- تعریف:

مجموعه ای مرتب از عناصر بطوریکه عمل درج در آن از انتهای صف(rear) و عمل حذف از آن، از ابتدای صف(front) انجام می شود.

- عملیات:

-تعریف صف خالی

-درج عنصر در صف(Add_Queue)- در انتهای صف

-حذف عنصر از صف(Remove_Queue)- از ابتدای صف

-بازیابی عنصر از صف(Retrieve_Queue)- از ابتدای صف

صف (QUEUE)

- پیاده سازی صف:

❖ <u>پیاده سازی خطی صف</u>

✓ با استفاده از آرایه

❖ <u>پیاده سازی غیرخطی صف</u>

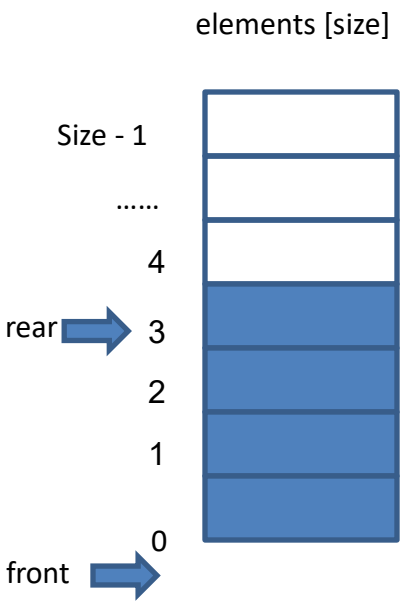
✓ با استفاده از لیست پیوندی

پیاده سازی خطی صف با استفاده از آرایه

- مولفه های مورد نیاز برای پیاده سازی صف:

✓ آرایه خطی

✓ متغیرهای front و rear برای نگهداری محل ابتدا و انتهای صف



تعریف صف خالی

- برای تعریف صف از ساختمان(struct) استفاده می کنیم، چرا؟ از ساختارها و انواع داده ای دیگر نیز می توان استفاده نمود؟

تعریف ساختمان:

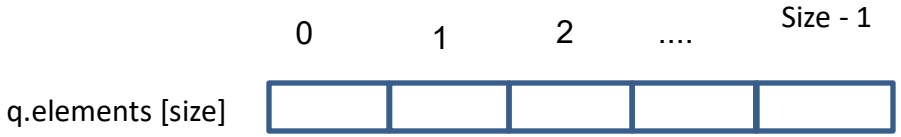
```
struct   نام{
بدنه ساختمان
};
 struct  QUEUE{
 int  elements[size];
 int  front;
 int rear;};
```

- تعریف صف خالی:

```
QUEUE    q;
q.front = -1;
q.Rear = -1;
```

مثال

- چندین دستور Add_Queue و Remove_Queue برای روشن شدن مطلب:



q.front = -1;

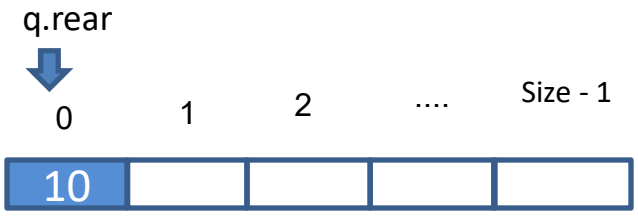
q.Rear = -1;

1)Add_Queue(q , 10);

برای اینکار ابتدا باید rear یک واحد اضافه شده و سپس مقدار 10 در آن محل آرایه درج شود:

q.rear ++;

q.elements[q.rear] = 10;



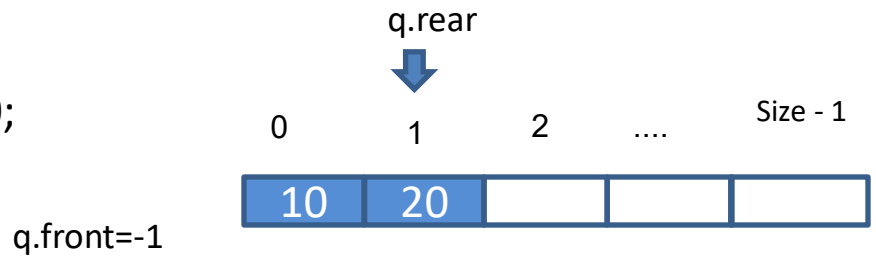
q.front=-1

مثال

2)Add_Queue(q , 20);

برای اینکارنیزابتدا باید rear یک واحد اضافه شده و سپس مقدار 20 در آن محل آرایه درج شود:

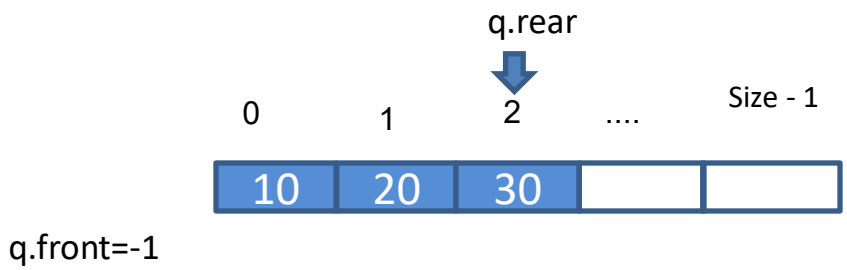
```
q.rear ++;
q.elements[q.rear] = 20;
```



3)Add_Queue(q , 30);

برای اینکار ابتدا باید rearیک واحد اضافه شده و سپس مقدار 30 در آن محل آرایه درج شود:

```
q.rear ++;
q.elements[q.rear] = 30;
```

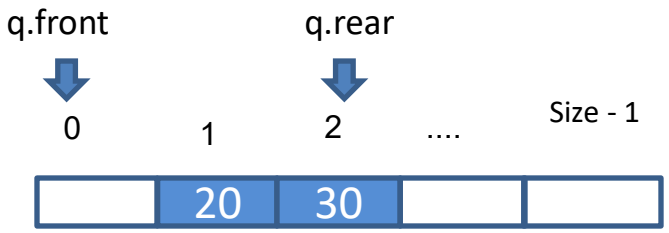


مثال

4) Remove_Queue(q);

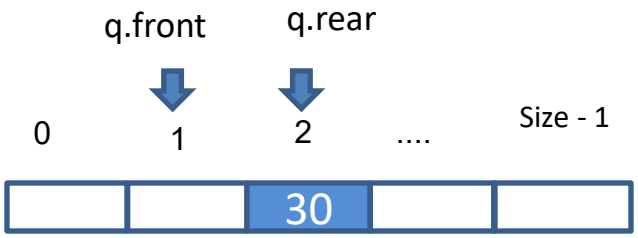
برای حذف یک مقدار از ابتدا، باید front یک واحد اضافه شده و سپس مقدار مورد نظر در محلی که front به آن اشاره دارد، خوانده شود.

```
q.front++;
X= q.elements[q.front];
```



5) Remove_Queue(q);

```
q.front++;
X= q.elements[q.front];
```



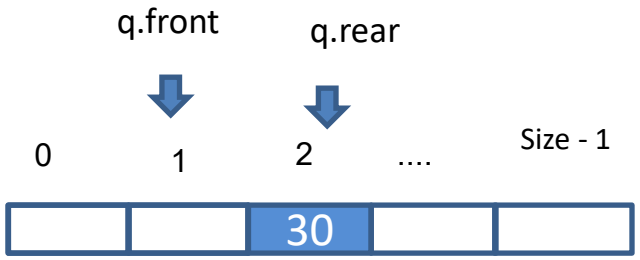
مثال

6)Retrieve_queue(q);

برای بازیابی یک مقدار از صف، باید هنگام خواندن، خانه front+1 خوانده شود. بازیابی به معنای خواندن بدون حذف کردن آن از صف می باشد.

X= q.elements[q.front + 1];

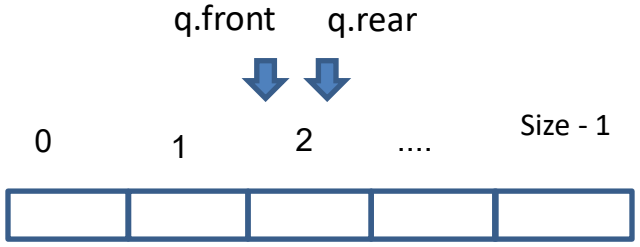
X = 30



7) Remove_Queue(q);

X= q.elements[++q.front];

X = 30



پیاده سازی تابع Add_Queue

```
void  Add_Queue(QUEUE  *q , int x)
{
   if(q-> rear ==size -1)
    { cout<<“QUEUE  overflow”;
      exit(1);}
   else{
      q->rear++;
      q->elements[q->rear]=x;}
}
```

اگر صف پر است؟

q->elements [++(q->rear)]=x;

پیاده سازی تابع Remove_Queue

```
int  Remove_Queue(QUEUE  *q )
{
   if(q-> rear == q->front)
     { cout<<“QUEUE  underflow”;
        exit(1);}
   else{
        q->front++;
        return (q->elements[q->front]);
}
  }
```

اگر صف خالی است؟

```
return(q->elements [++(q->front)]);
```

پیاده سازی تابع Retrieve_Queue

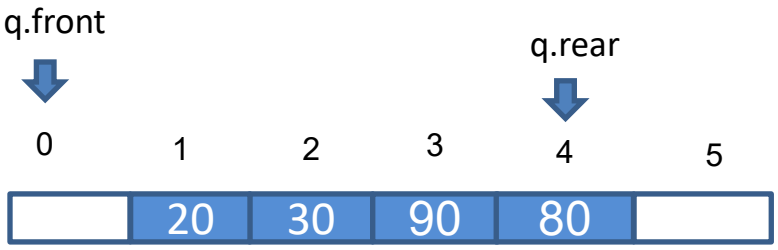
```
int  Retrieve_Queue(QUEUE  *q )
{
 if(q-> rear == q->front)
     { cout<<“QUEUE  underflow”;
        exit(1);}
    else
         return  (q->elements[q->front + 1]);
}
```



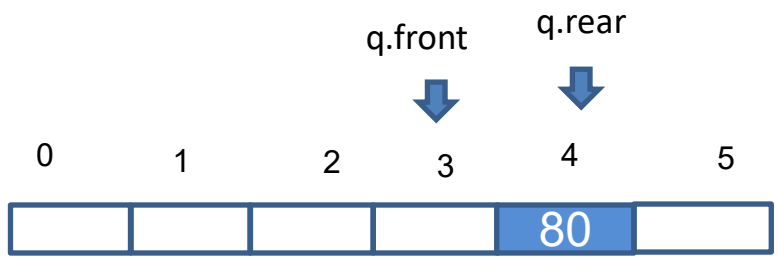
اگر صف خالی است؟

مشکلات پیاده سازی صف بصورت خطی

- مثال های زیر را در نظر بگیرید:



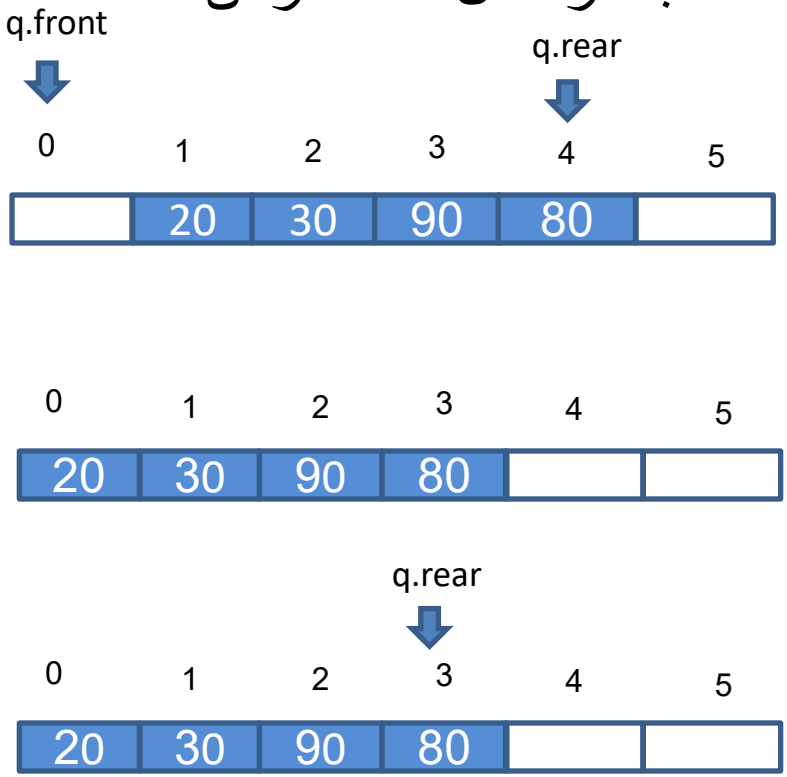
- تعداد محل خالی؟
- تعداد محل قابل استفاده؟



مشکلات پیاده سازی صف بصورت خطی

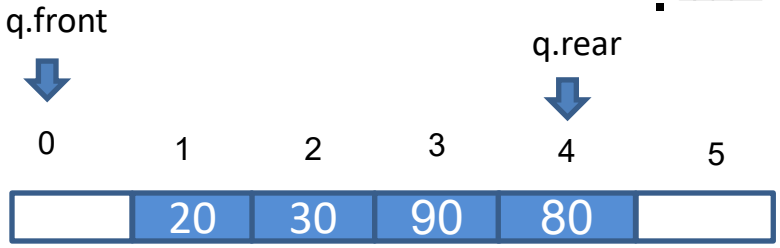
راه حل:

- شیفت عناصر صف به سمت ابتدای صف، با هر عمل حذف از آن

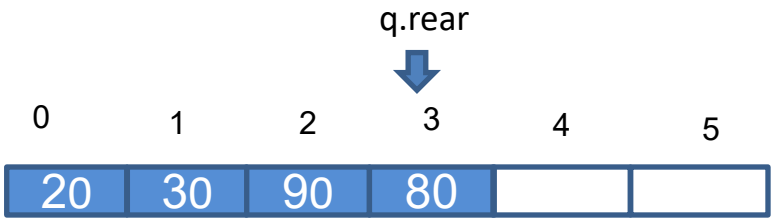


شیفت عناصر صف به سمت ابتدای صف، با هر عمل حذف از آن

- با اجرای الگوریتم زیر، تمامی عناصر موجود در صف یکی به سمت ابتدای صف شیفت پیدا می کنند.

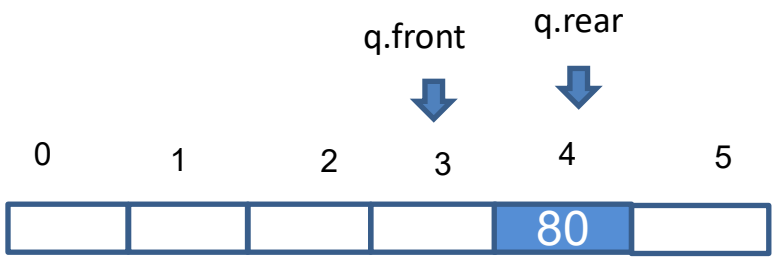



```
for (int i= 0; i< q.rear; i++)
      q.elements[i] = q.elements[i+1];
 q.rear --;
```

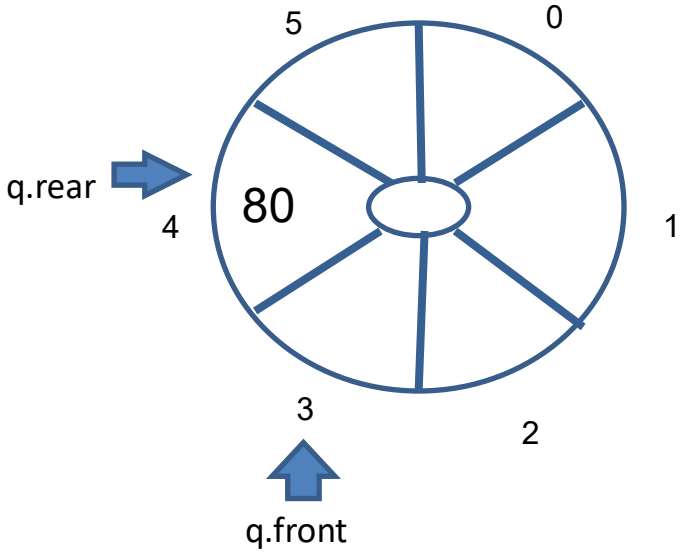


مشکلات پیاده سازی صف بصورت خطی

- به مثال زیر توجه کنید:

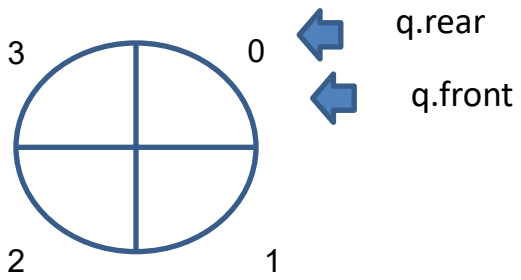


- برای استفاده بهینه از فضای حافظه، به عنوان راه حلی دیگر، صف های حلقوی پیشنهاد می شود:



صف حلقوی

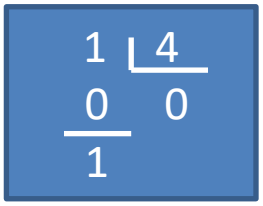
- با ذکر مثال، در یک صف حلقوی عناصر را درج و حذف می نماییم:

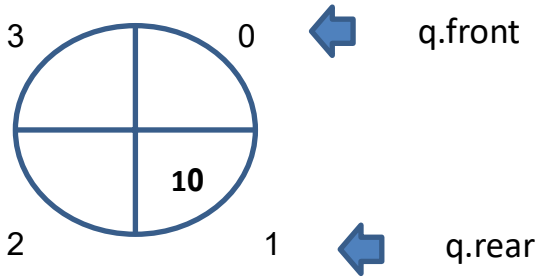


1) Add_Queue(q,10)

q.rear++%size ⇨ (0+1)%4 ⇨ 1

q.elements[q.rear] = 10;



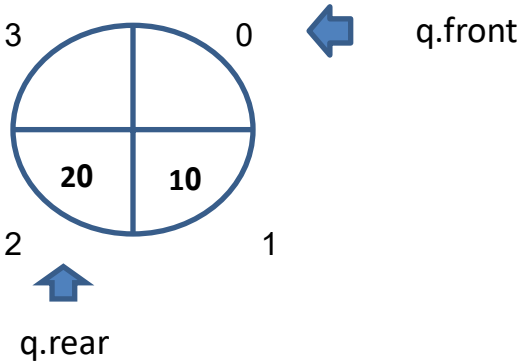


صف حلقوی

2) Add_Queue(q,20)

q.rear++%size ⇒ (1+1)%4 ⇒ 2

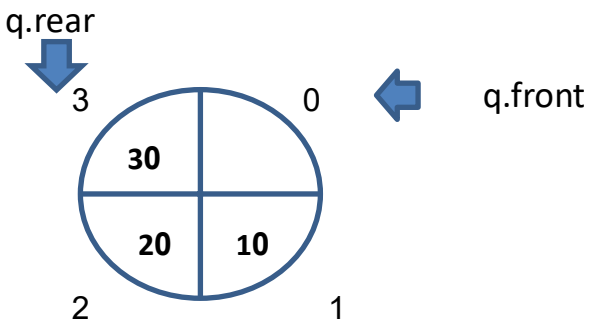
q.elements[q.rear] = 20;



3) Add_Queue(q,30)

q.rear++%size ⇒ (2+1)%4 ⇒ 3

q.elements[q.rear] = 30;

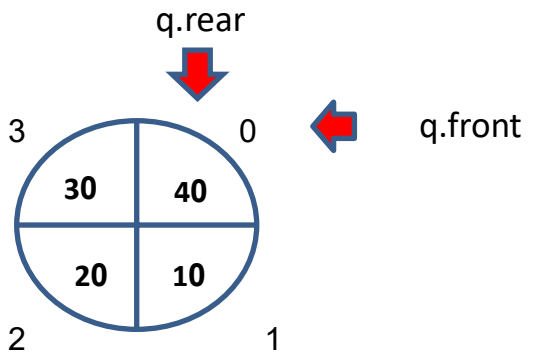


صف حلقوی

4) Add_Queue(q,40)

q.rear++%size ⇨ (3+1)%4 ⇨ 0

q.elements[q.rear] = 40;

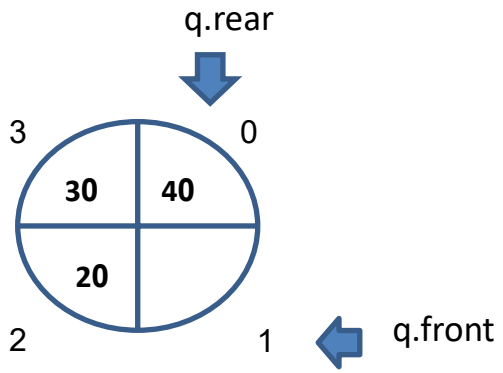


✓ صف پر است.

5) Remove_Queue(q)

q.front++%size ⇨ (0+1)%4 ⇨ 1

x = q.elements[q.front];

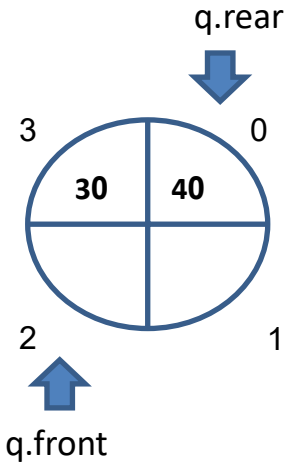


صف حلقوی

6) Remove_Queue(q)

q.front++%size ⇨ (1+1)%4 ⇨ 2

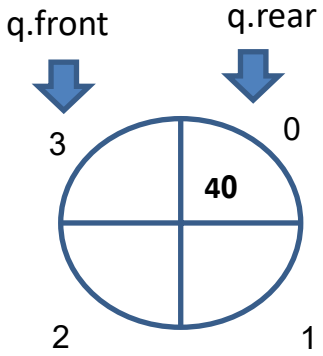
x = q.elements[q.front];



7) Remove_Queue(q)

q.front++%size ⇨ (2+1)%4 ⇨ 3

x = q.elements[q.front];

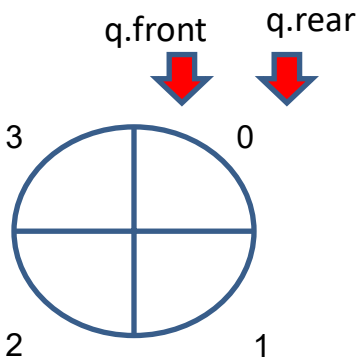


صف حلقوی

8) Remove_Queue(q)

q.front++%size ⇒ (3+1)%4 ⇒ 0

x = q.elements[q.front];



✓ صف خالی است.

مشکل صف حلقوی

- برابری front و rear در هر دو حالت خالی بودن و پر بودن صف.

راه حل:

1)استفاده از یک متغیر flag برای نگهداری آخرین عمل انجام شده در صف

اگر آخرین عمل صف، درج در آن باشد: برابری دو متغیر front و rear به معنای پر بودن صف خواهد بود.

اگر آخرین عمل صف، حذف از آن باشد: برابری دو متغیر front و rear به معنای خالی بودن صف خواهد بود.

مشکل صف حلقوی

- برابری front و rear در هر دو حالت خالی بودن و پر بودن صف.

راه حل:

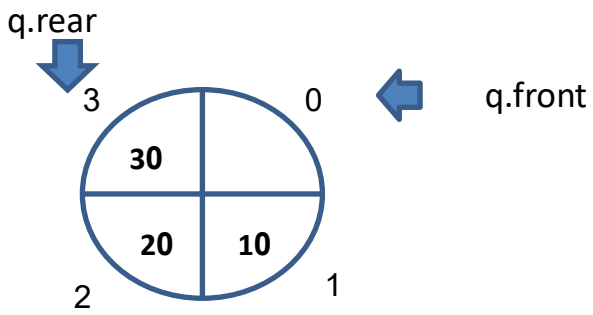
2) درنظر گرفتن یک فضای خالی برای حدفاصل میان front و rear بدین صورت که بدون وابستگی به سایز آرایه، همیشه یک فضای آرایه غیر قابل استفاده خواهد بود.

شرط پر بودن صف

```
if (q.rear++%size == q.front)
      cout<< “Queue overflow”;
```

شرط خالی بودن صف

```
if (q.rear== q.front)
      cout<< “Queue underflow”;
```



✓ صف پر است.

کاربردهای صف

- صف های اولویت
- الگوریتم های زمان بندی سیستم عامل
- سیستم های اشتراک زمانی
- .....