دانشگاه شهید باهنر کرمان

چهار: صف

ساختمان داده ها و الگوریتم

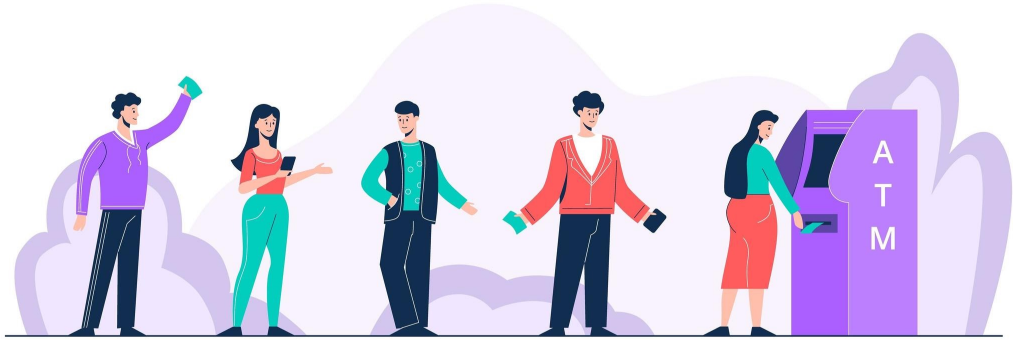
مدرس: دکتر نجمه منصوری

نگارنده: سجاد هاشمیان

صف

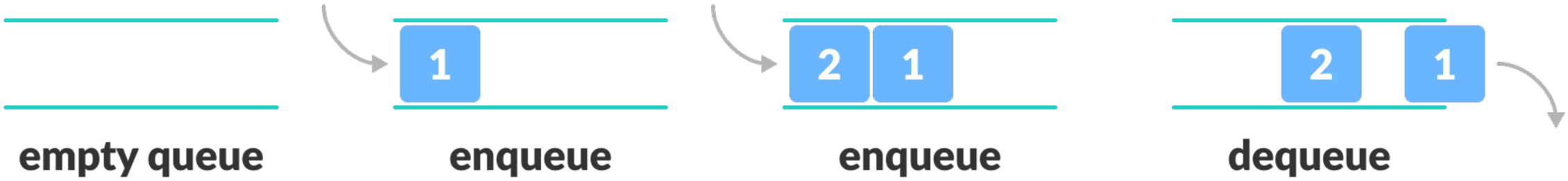
تفاوت صف با پشته در این است که درج و حذف عناصر در صف از اصل «اولین ورودی، اولین خروجی(FIFO)» پیروی میکند.

به عبارت دیگر هر زمان که بخواهیم میتوانیم یک عنصر به انتهای صف اضافه کنیم، اما در هنگام حذف تنها اجازه داریم اولین عنصر (عنصری که بیشتر از بقیه‌ی عناصر در صف بوده است) را از صف حذف کنیم.



عملیات اصلی صف

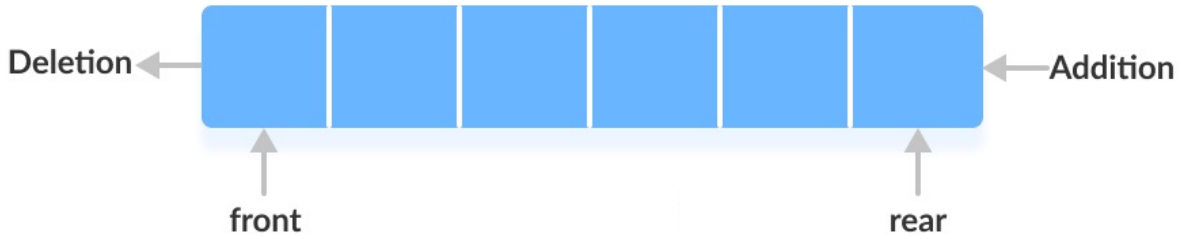
- درج: افزودن یک عنصر به انتهای صف
- حذف: خارج کردن یک عنصر از ابتدای صف



پیاده‌سازی

برای پیاده‌سازی صف ها از آرایه‌ها کمک می‌گیریم، البته می‌توان صف ها را به کمک دیگر ساختمان داده های پایه نیز پیاده سازی کرد.

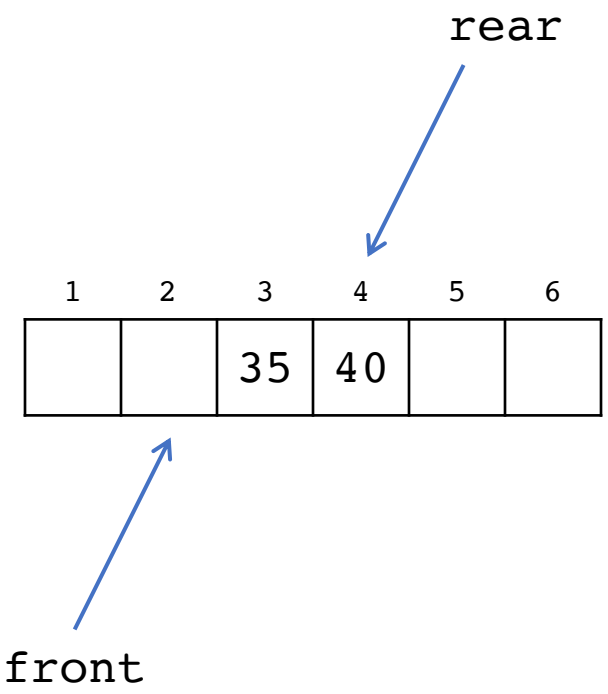
به‌طور کلی برای پیاده سازی یک صف، آرایه ای nتایی همانند $Q[1..n]$ استفاده میکنیم و برای کنترل حدود مرزی از دو اشاره‌گر کمک می‌گیریم.



اشاره‌گر ها

اشاره‌گر front به خانه قبل از خانه اول صف اشاره می‌کند.

اشاره‌گر rear به خانه آخر صف اشاره میکند.

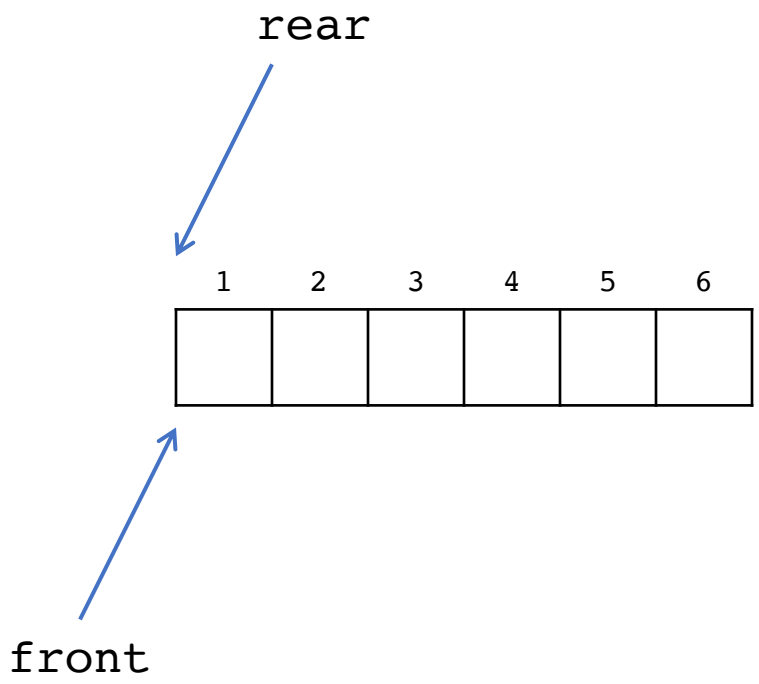


شرایط اشاره‌گر ها

اشاره‌گر front به خانه قبل از خانه اول صف اشاره می‌کند.

اشاره‌گر rear به خانه آخر صف اشاره میکند.

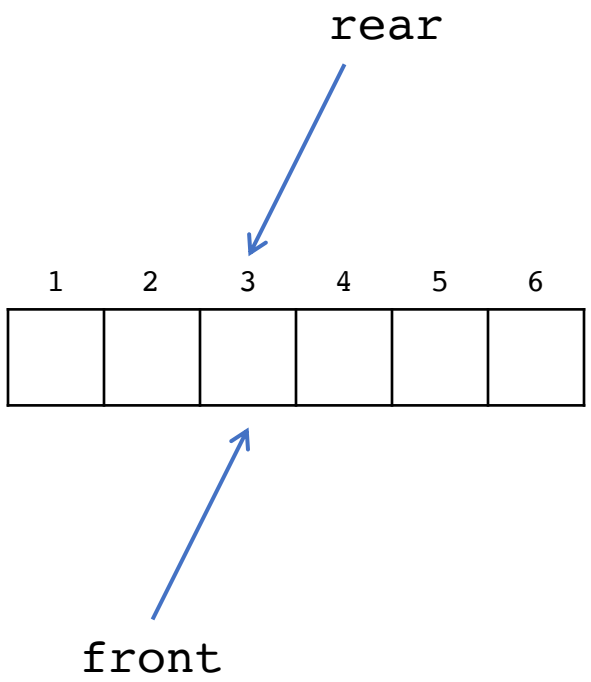
وضعیت اولیه: $front = rear = 0$



شرایط اشاره‌گر ها

اشاره‌گر front به خانه قبل از خانه اول صف اشاره می‌کند.

اشاره‌گر rear به خانه آخر صف اشاره میکند.



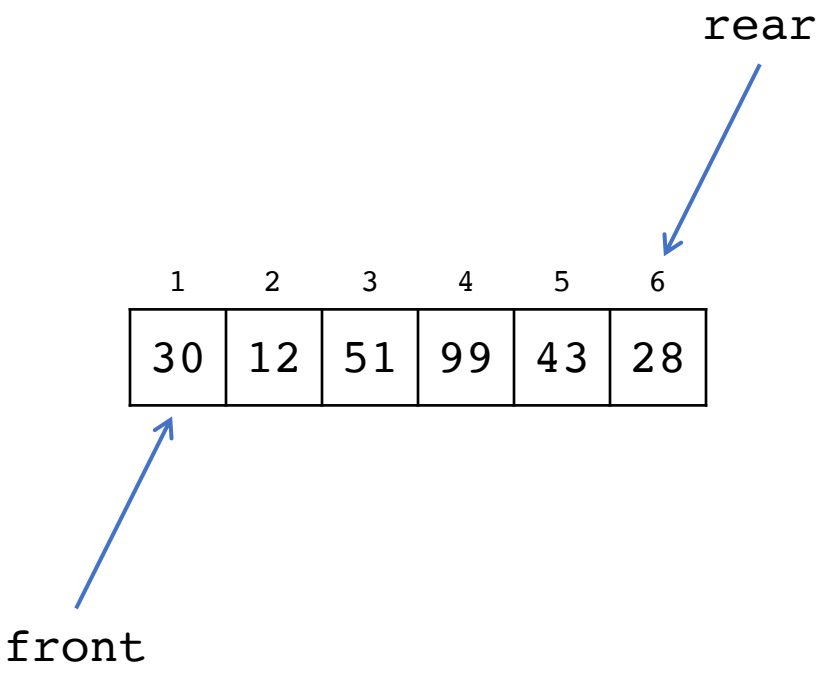
خالی بودن صف : $front = rear$

شرایط اشاره‌گر ها

اشاره‌گر front به خانه قبل از خانه اول صف اشاره می‌کند.

اشاره‌گر rear به خانه آخر صف اشاره میکند.

پر بودن صف: $rear = n$



عملیات درج

```
def insert(item):
    if(rear==n):
        return "Queue is Full"
    else:
        rear=rear+1
        Q[rear]=item
```

دیده می‌شود که چون rear به خانه آخر صف اشاره می‌کند، برای عمل درج ابتدا rear یک واحد به جلو حرکت کرده سپس داده item در خانه خالی انتهای صف درج می‌شود.

عملیات حذف

```
def delete():
    if(rear == front):
        return "Queue is Empty"
    else:
        front = front+1
        return Q[front]
```

دیده می‌شود که چون front همیشه به قبل از اول صف اشاره می‌کند برای عمل حذف ابتدا front واحد به جلو حرکت کرده سپس داده جدید ابتدای صف را بر می‌گرداند.

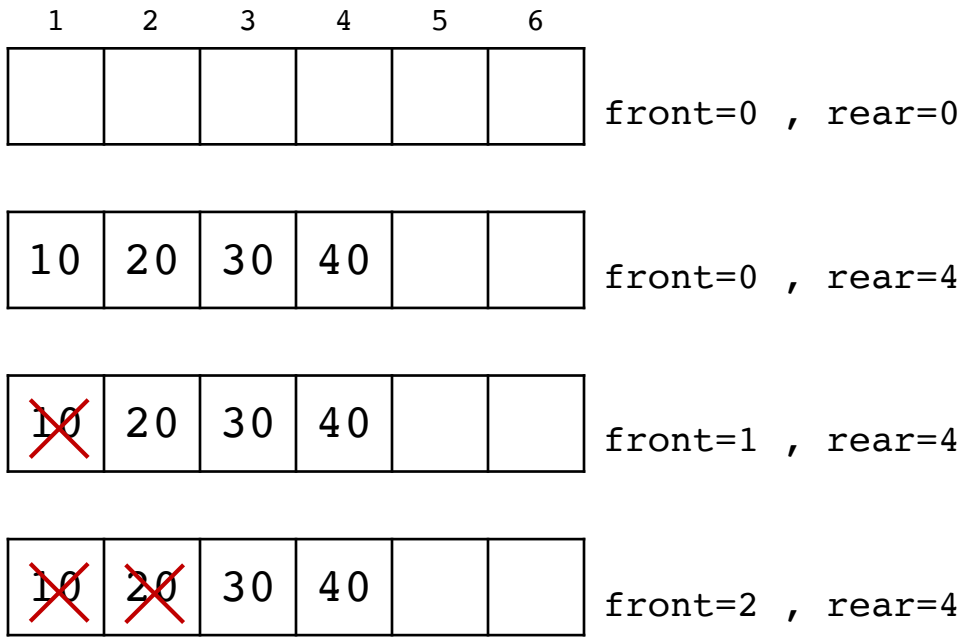
مثال

۱. صف ۶ عضوی روبه‌رو را در نظر بگیرید.

۲. درج عناصر ۱۰، ۲۰، ۳۰، ۴۰

۳. حذف داده

۴. حذف داده



دقت کنید که در وضعیت بالا با این که ۲ داده از ابتدای صف حذف شده‌اند اما فقط دو خانه ۵ و ۶ برای درج وجود دارند چون حرکت front به سمت جلو بوده و امکان استفاده از خانه‌های قبلی را ندارد.

مشکل صف‌های خطی

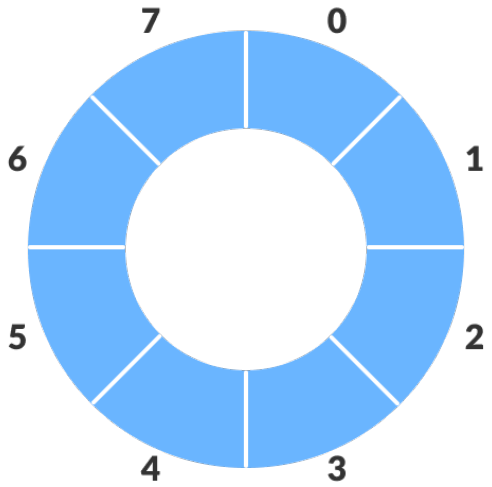
حرکت تدریجی عناصر به سمت انتهای صف و عدم امکان استفاده از خانه‌های ابتدای صف که در اثر حذف خالی شده‌اند.

۱. **شیفت خانه‌های انتهای صف** به سمت ابتدای صف که این عمل در صورتی‌که تعداد خانه‌های صف زیاد باشد هزینه بالایی خواهد داشت. (بنابراین بعد از هر t عمل حذف باید یک عملیات شیفت از مرتبه O(t) انجام دهیم)

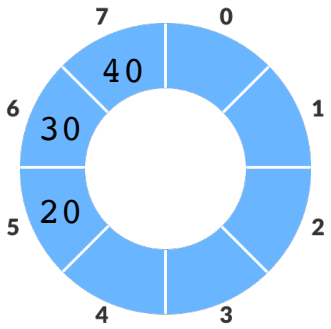
۲. **استفاده از صف حلقوی** به این مفهوم که زمانی که به انتهای صف رسیدیم عمل درج را دوباره از ابتدای صف انجام می‌دهیم و این حرکت چرخشی می‌باشد. (بنابراین با زمان اجرای ثابت عمل درج و حذف انجام می‌شود.)

صف حلقوی

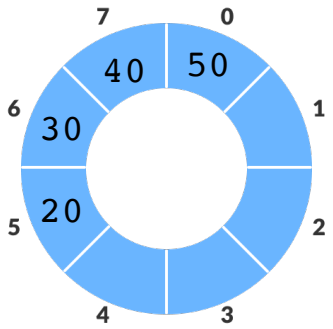
آرایه nتایی Q[0...n-1] را می‌توان به صورت یک صف حلقوی در نظر گرفت به طوری که در این صف زمانی که rear برابر n-1 می‌شود، عنصر بعدی در خانه 0 قرار می‌گیرد.

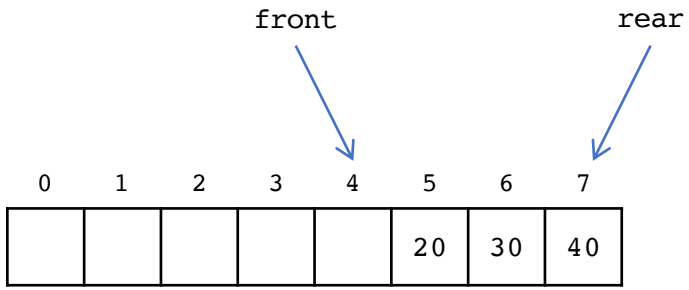


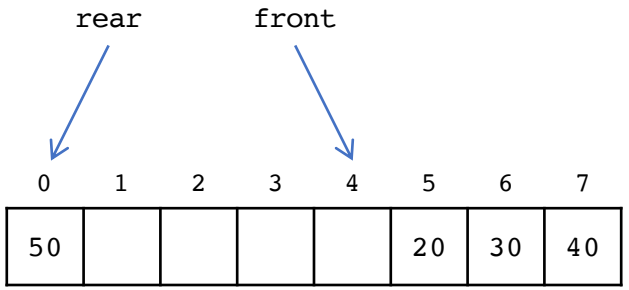
صف حلقوی



درج ۵۰







شرایط اشاره‌گرها

خالی(Empty) :

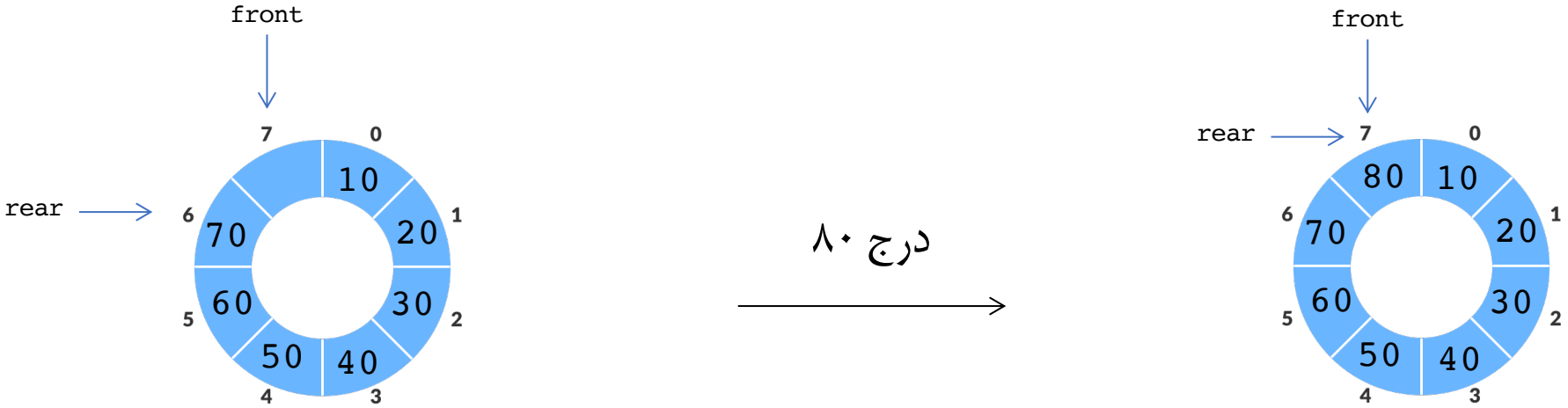
$$front = rear$$

پر (Full) :

$$front = rear + 1 \ (mod \ n)$$

در هر صف حلقوی Q[0...n-1] با ظرفیت n همیشه یک خانه باید خالی باشد و حداکثر از n-1 خانه صف می‌توان استفاده کرد؛ این به آن علت است که بتوان وضیت پر یا خالی بودن صف حلقوی را تشخیص داد.

خانه‌های قابل استفاده در صف حلقوی



بعد از درج ۸۰ در صف حلقوی و استفاده از همان یک خانه‌ای که باید در صف خالی می‌ماند front=rear=7 می‌شود که همان شرط خالی بودن صف است در صورتی که صف پر شده و این یک مشکل است برای نبودن تناقض فوق و امکان تشخیص وضعیت پر یا خالی بودن صف حلقوی یک خانه را خالی نگه می‌داریم.

عملیات درج و حذف

در هر دو روش حذف و درج به ترتیب از جملات

```
front=front+1 (mod n)
```

```
rear =rear +1 (mod n)
```

استفاده شده است که علت استفاده از mod در شرایطی است که front یا rear به خانه آخر اشاره کرده باشند و حرکت به جلو آن‌ها را به ابتدای صف منتقل می‌کند.

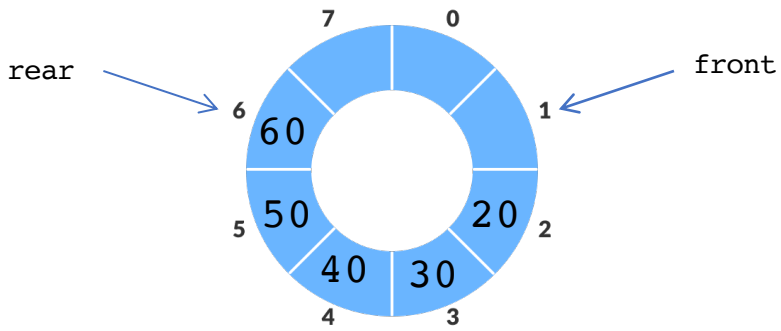
```
def delete():
    if(rear == front):
        return "Queue is Empty"
    else:
        front = (front+1) % n
        return Q[front]
```

```
def insert(item):
    if((rear+1) % n==front):
        return "Queue is Full"
    else:
        rear = (rear+1) % n
        Q[rear]=item
```

مثال: درج و حذف

برای مثال میخواهیم ۷۰ را در این لیست درج کنیم.

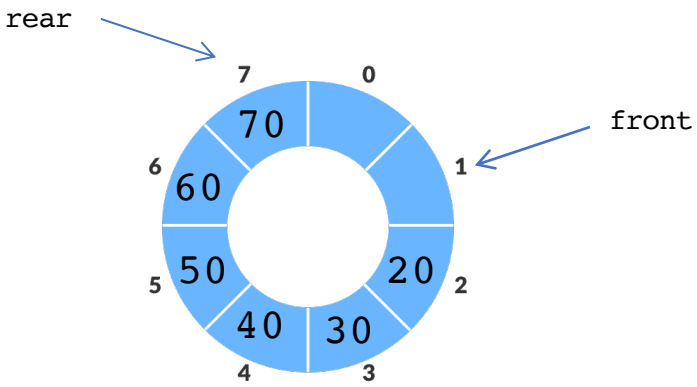
```
def insert(item):
    if((rear+1) % n==front):
        return "Queue is Full"
    else:
        rear = (rear+1) % n
        Q[rear]=item
```



مثال: درج و حذف

برای مثال میخواهیم ۷۰ را در این لیست درج کنیم.

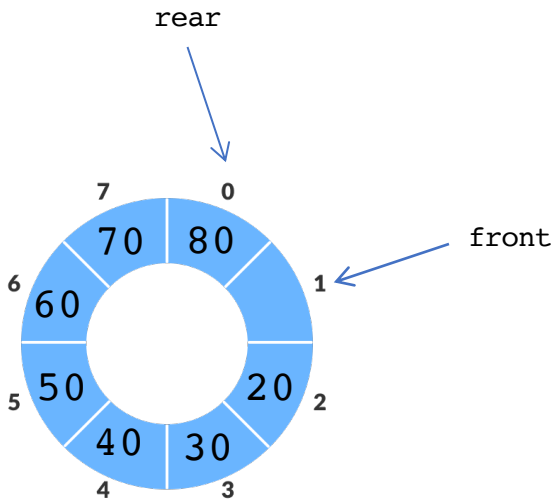
```
def insert(item):
    if((rear+1) % n==front):
        return "Queue is Full"
    else:
        rear = (rear+1) % n
        Q[rear]=item
```



مثال: درج و حذف

```
def insert(item):
    if((rear+1) % n==front):
        return "Queue is Full"
    else:
        rear = (rear+1) % n
        Q[rear]=item
```

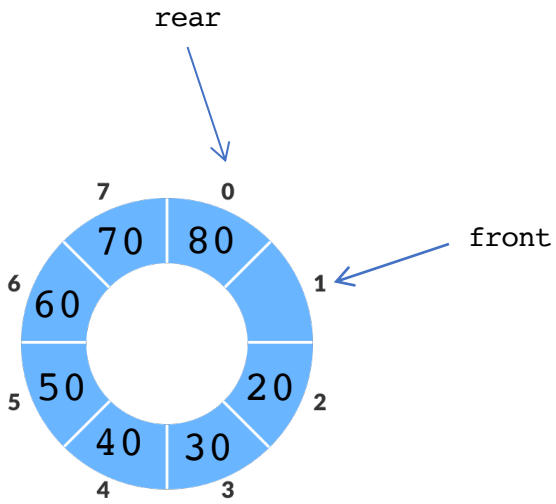
برای مثال میخواهیم ۸۰ را در این لیست درج کنیم.



مثال: درج و حذف

```
def insert(item):
    if((rear+1) % n==front):
        return "Queue is Full"
    else:
        rear = (rear+1) % n
        Q[rear]=item
```

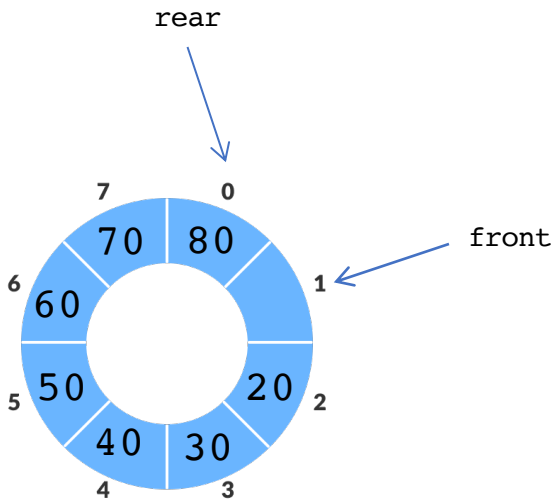
حالا لیست کاملا پر شده و درج عدد جدیدی ممکن نیست.



مثال: درج و حذف

بیاید حذف را امتحان کنیم:

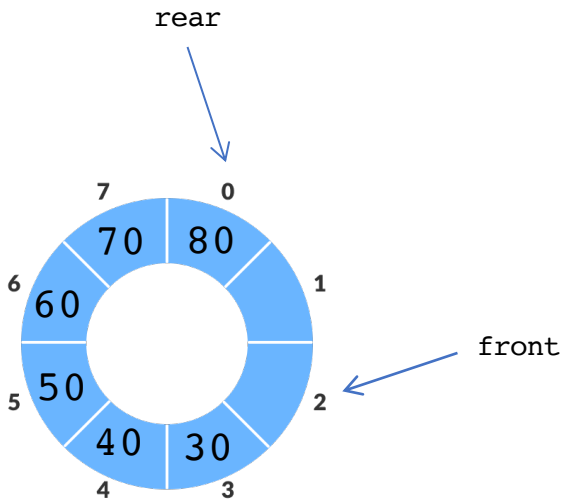
```
def delete():
    if(rear == front):
        return "Queue is Empty"
    else:
        front = (front+1) % n
        return Q[front]
```



مثال: درج و حذف

```
def delete():
    if(rear == front):
        return "Queue is Empty"
    else:
        front = (front+1) % n
        return Q[front]
```

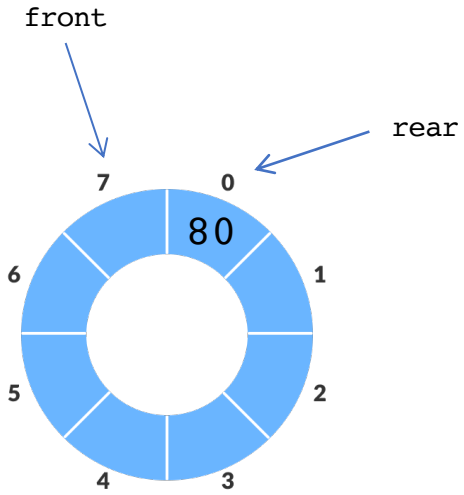
عدد ۲۰ از ابتدا لیست حذف کرده‌ایم.



مثال: درج و حذف

```
def delete():
    if(rear == front):
        return "Queue is Empty"
    else:
        front = (front+1) % n
        return Q[front]
```

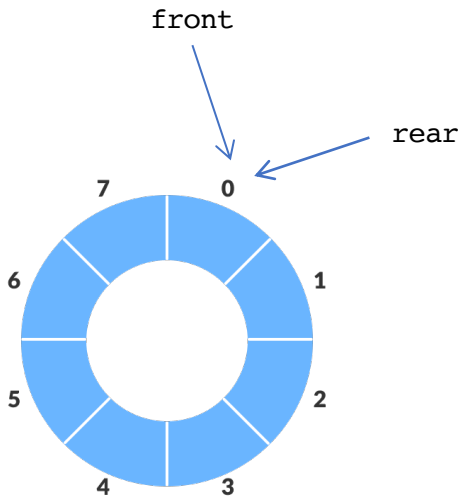
برای حذف ۸۰ از لیست زیر هم داریم:



مثال: درج و حذف

```
def delete():
    if(rear == front):
        return "Queue is Empty"
    else:
        front = (front+1) % n
        return Q[front]
```

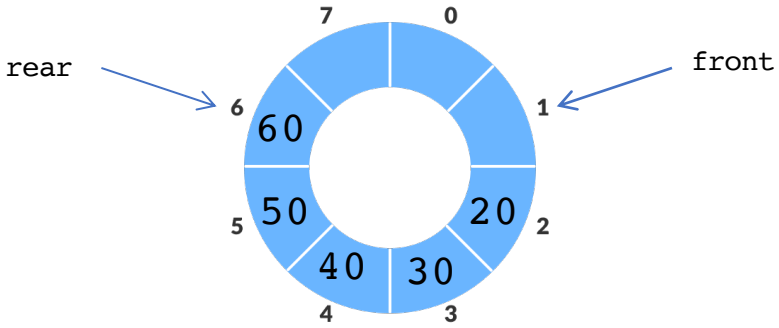
حالا دیگر لیست خالی شده و عمل حذف دیگر ممکن نیست.



وضعیت خانه های یک صف

برای یک صف حلقوی Q[0...n-1]، داریم:

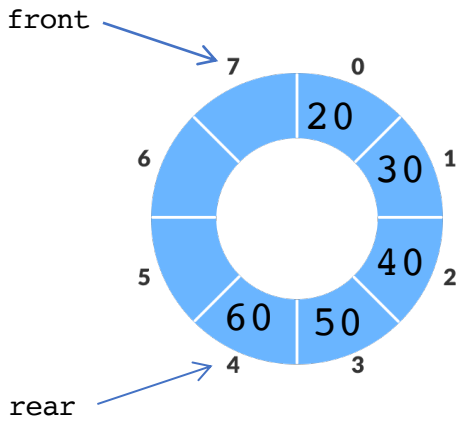
اگر $rear \geq front$ آنگاه $rear - front$ برابر با تعداد خانه‌های پر صف خواهد بود.



وضعیت خانه های یک صف

برای یک صف حلقوی Q[0...n-1]، داریم:

اگر $front > rear$ آنگاه $n - (front - rear)$ برابر با تعداد خانه‌های پر صف خواهد بود.



مثال

اگر Q1 و Q2 دو صف باشند که به صورت زیر تعریف شده باشند(عنصر سمت چپ، ابتدای صف ها است)، مطلوب است، Q3 را محاسبه کنید.

```
Q1={10,25,17,41,19,26,75}
Q2={1,5,7,4,9,6}
Q3={}

i=0
while(not empty(Q1) and not empty(Q2)):
    i=i+1
    x=delete(Q1)
    y=delete(Q2)
    if(y==i):
    add(Q3,x)

return Q3
```

مثال

در این قطعه برنامه ابتدا یک صف خالی به نام Q3 ایجاد می‌شود. سپس در حلقه while تا زمانی که صف Q1 و صف Q2 هیچ‌کدام خالی نباشد یکی به شمارنده i اضافه شده و دو عنصر یکی از صف Q1 دیگری از صف Q2 حذف می‌شود و به ترتیب در Q3 قرار می‌گیرد. اگر مقدار y با شمارنده i برابر باشد، مقدار x در صف Q3 قرار خواهد گرفت.در این صورت خواهیم داشت:

i	x	y	شرط حلقه	شرط y=i	Q3
0	-	-	+		-
1	10	1	+	+	10
2	25	5	+	-	10
3	17	7	+	-	10
4	41	4	+	+	10, 41
5	19	9	+	-	10, 41
6	26	6	-	+	10, 41, 26