به نام او، به یاد او، برای او

ساختمان داده‌ها و الگوریتم‌ها

صف‌ها

Q u e u e s

Dr. *Ali Valinejad*

valinejad.ir
valinejad@umz.ac.ir

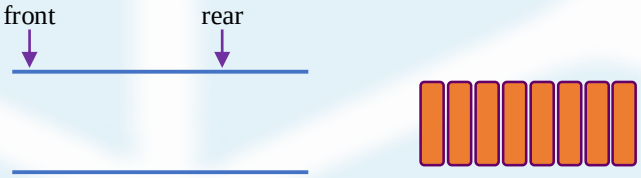
فهرست مطالب

تعریف صف

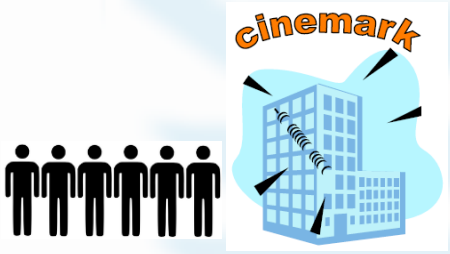
- صف (صف خطی) نوعی لیست خطی است که یک مجموعه از عناصر را طبق ترتیب خاصی ذخیره می‌کند.
- عملیات حذف و اضافه کردن یک عنصر طبق قانون زیر صورت می‌گیرد:

  First IN FIRST OUT (FIFO)

  اولین عنصر ورودی به صف، اولین عنصری است که خارج می‌شود.
- از دو نشانگر(اشاره‌گر) front و rear به ترتیب برای اشاره به **ابتدا** و **انتهای** صف استفاده می‌شود.



- عملیات حذف عناصر فقط از طریق front امکان پذیر است.
- عملیات اضافه کردن عناصر فقط از طریق rear امکان پذیر است.

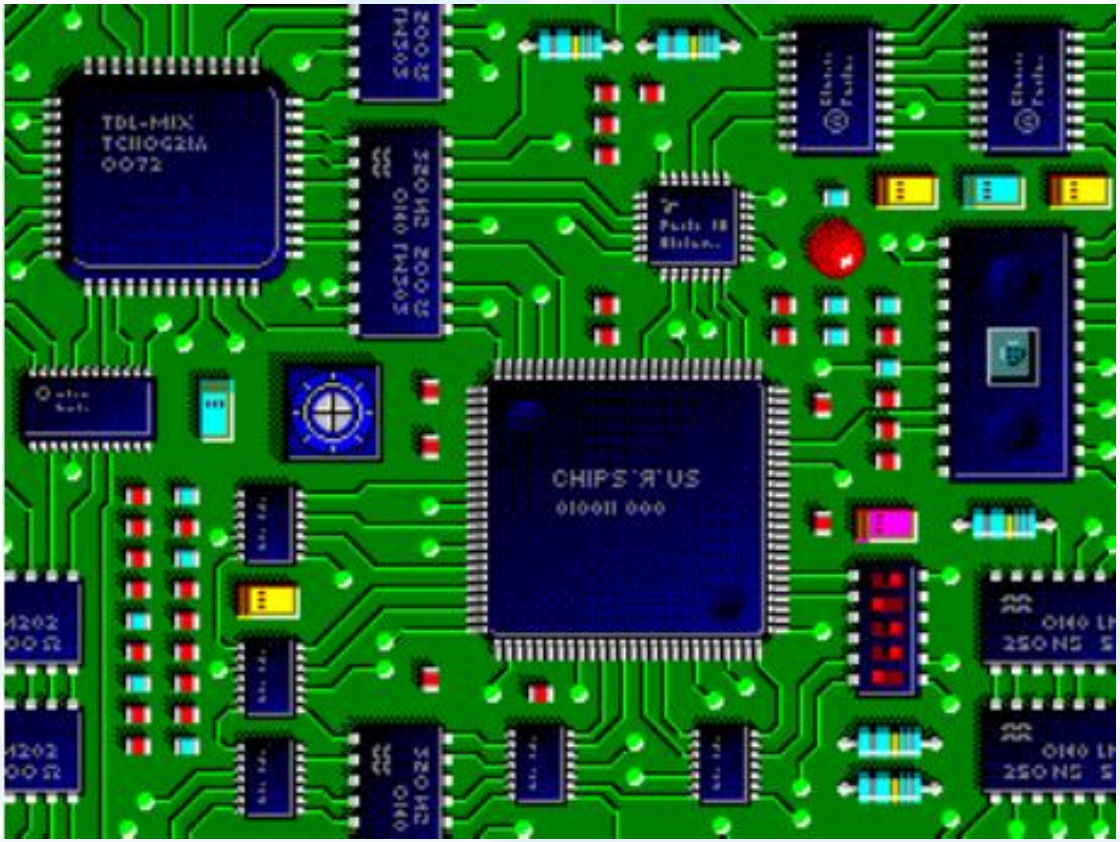


کاربردهای صف

- Real life
  - Waiting in line
  - Waiting on hold for tech support
- More applications related to computer science
  - Threads
  - Job scheduling (e.g. Round-Robin algorithm for CPU allocation

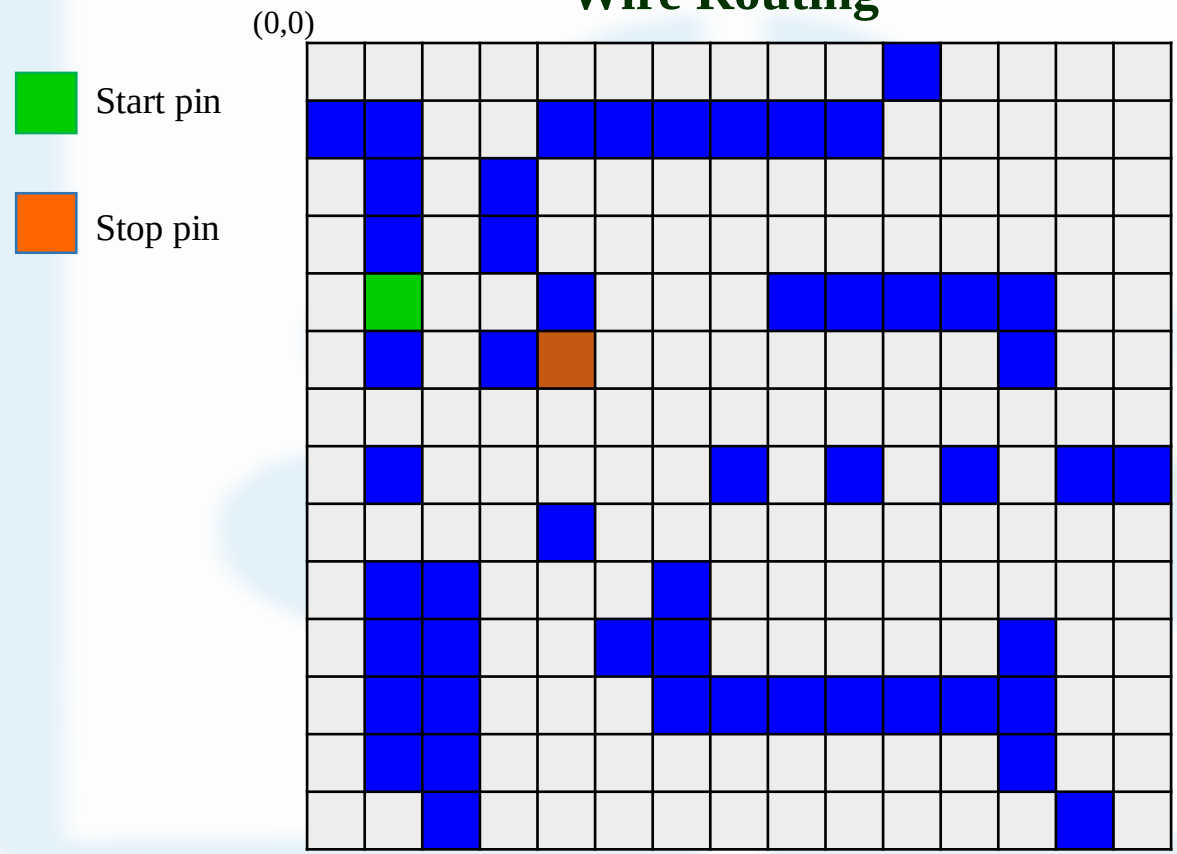
کاربردهای صف

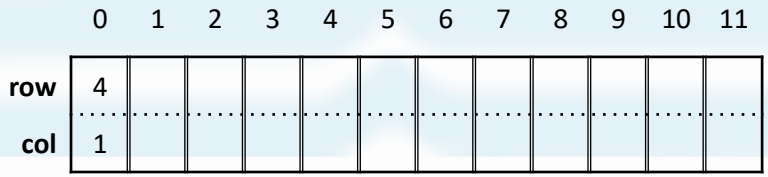
مسیریابی سیم



کاربردهای صف

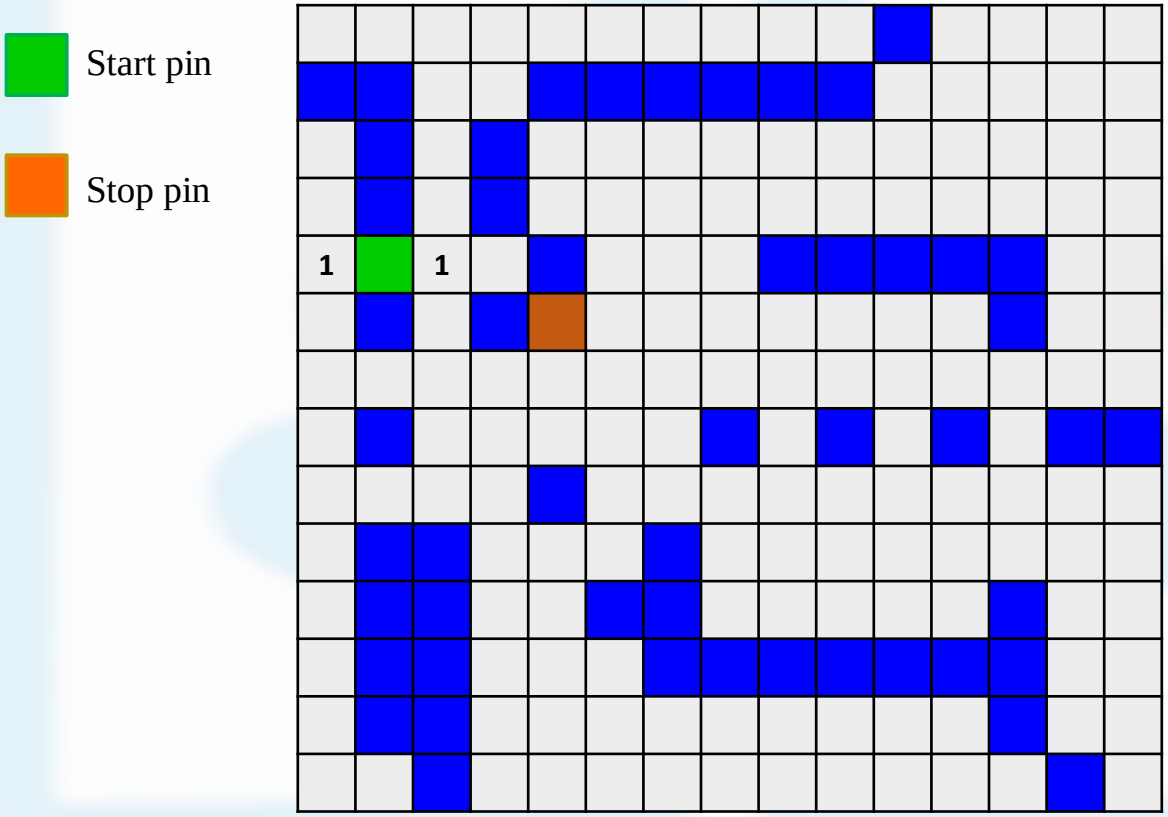
Wire Routing





کاربردهای صف

Wire Routing



برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۱ واحد از نقطه شروع فاصله دارند.

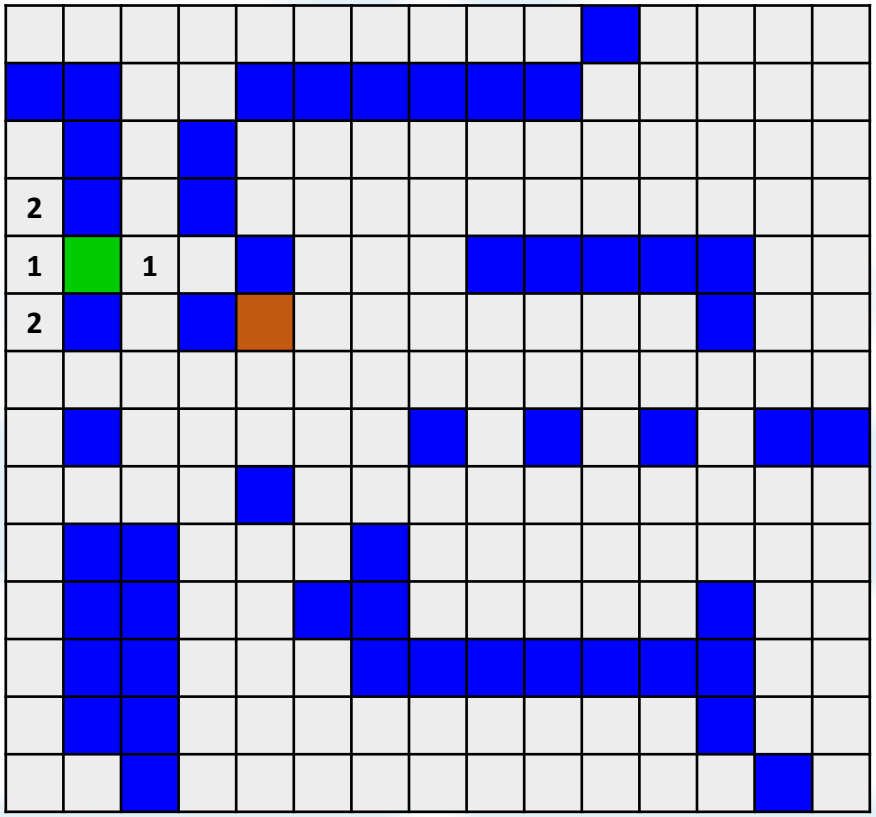
	0	1	2	3	4	5	6	7	8	9	10	11
row		4	4									
col		0	2									

کاربردهای صف

Wire Routing

Start pin

Stop pin



برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۲ واحد از نقطه شروع فاصله دارند.

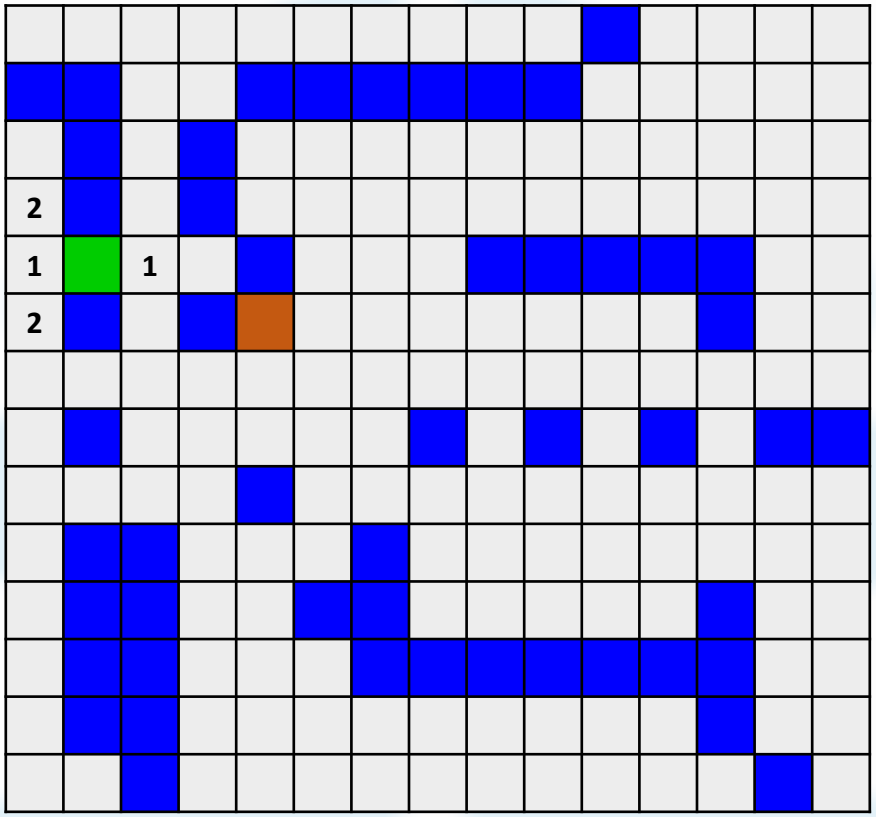
	0	1	2	3	4	5	6	7	8	9	10	11
row			4	3	5							
col			2	0	0							

کاربردهای صف

Wire Routing

Start pin

Stop pin



برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۲ واحد از نقطه شروع فاصله دارند.

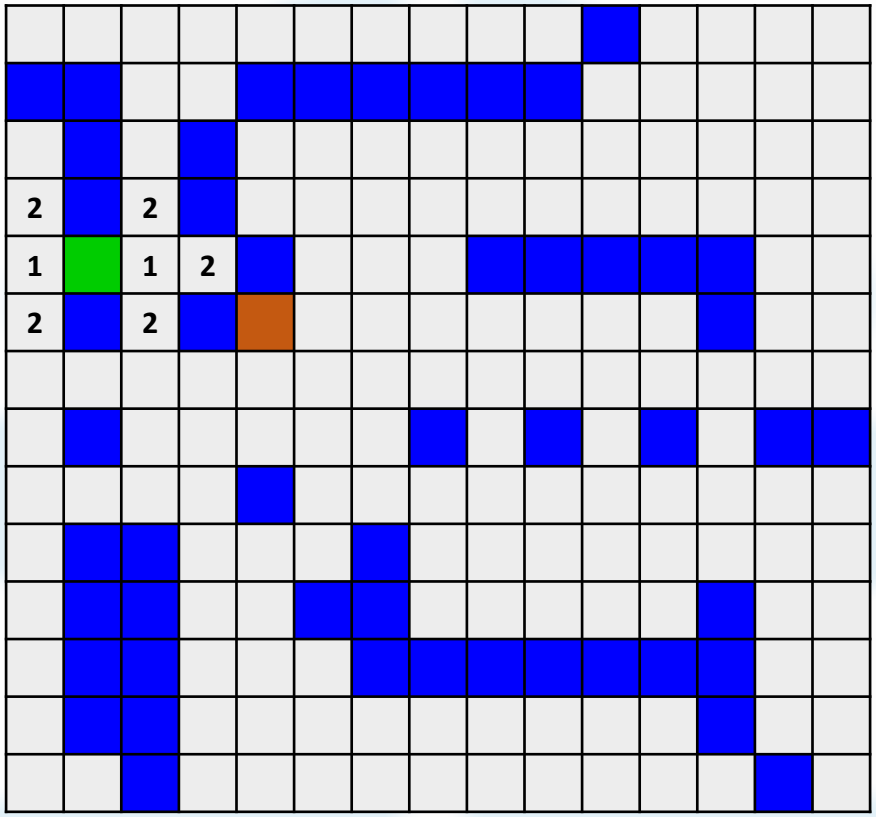
	0	1	2	3	4	5	6	7	8	9	10	11
row			4	3	5							
col			2	0	0							

کاربردهای صف

Wire Routing

Start pin

Stop pin



برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۲ واحد از نقطه شروع فاصله دارند.

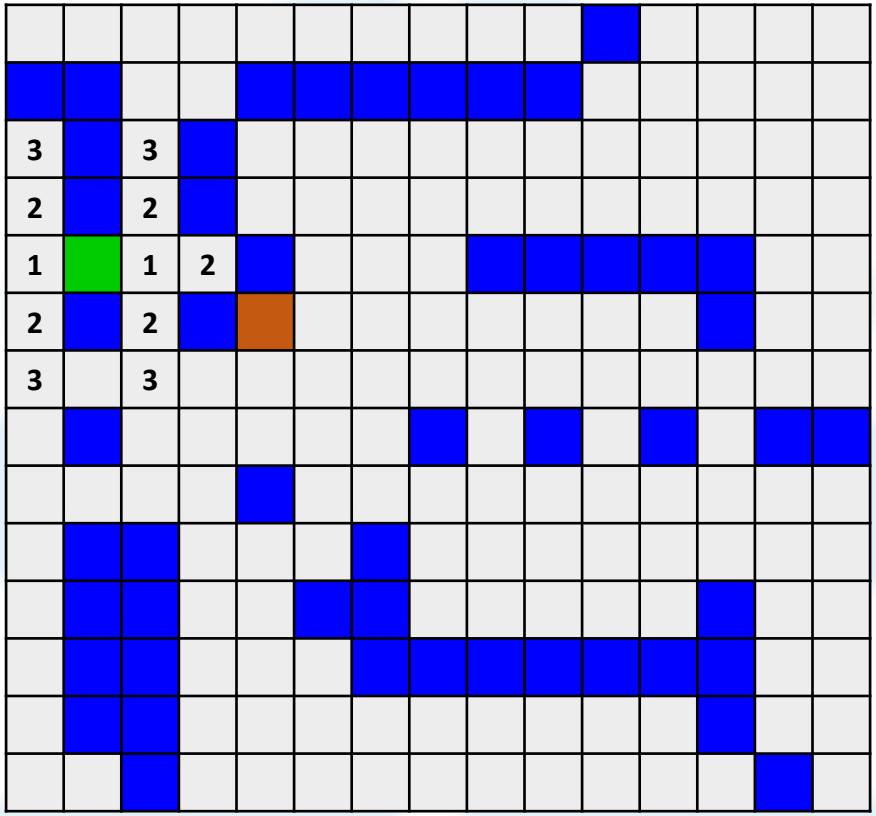
	0	1	2	3	4	5	6	7	8	9	10	11
row				3	5	3	4	5				
col				0	0	2	3	2				

کاربردهای صف

Wire Routing

Start pin

Stop pin



برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۳ واحد از نقطه شروع فاصله دارند.

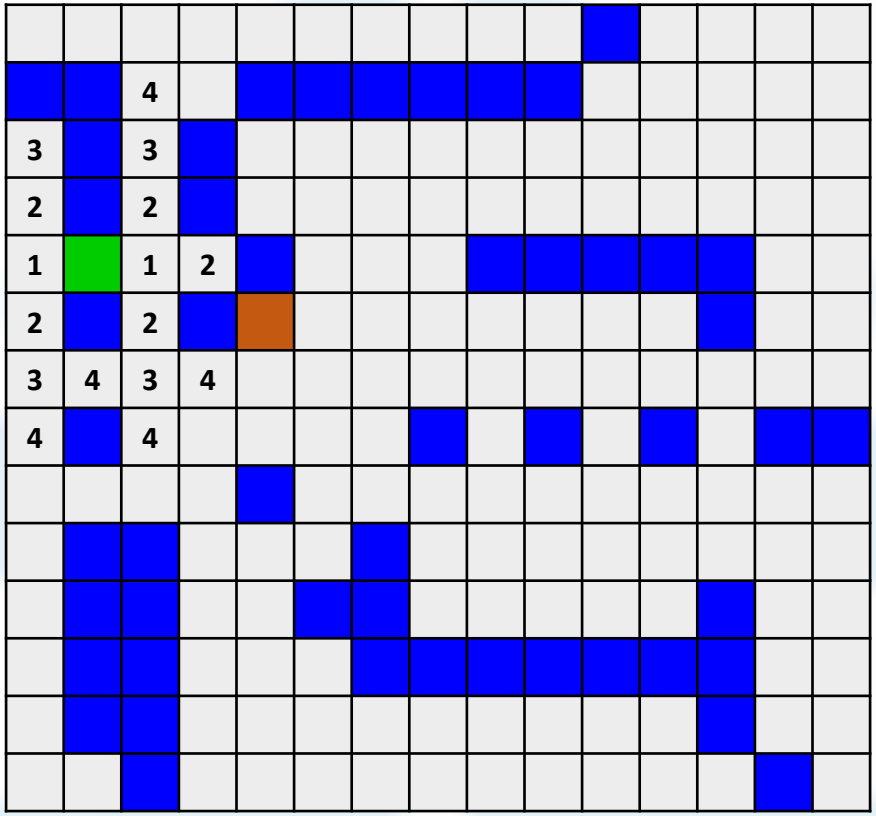
	0	1	2	3	4	5	6	7	8	9	10	11	
row									2	6	2	6	
col									0	0	2	2	

کاربردهای صف

Wire Routing

Start pin

Stop pin



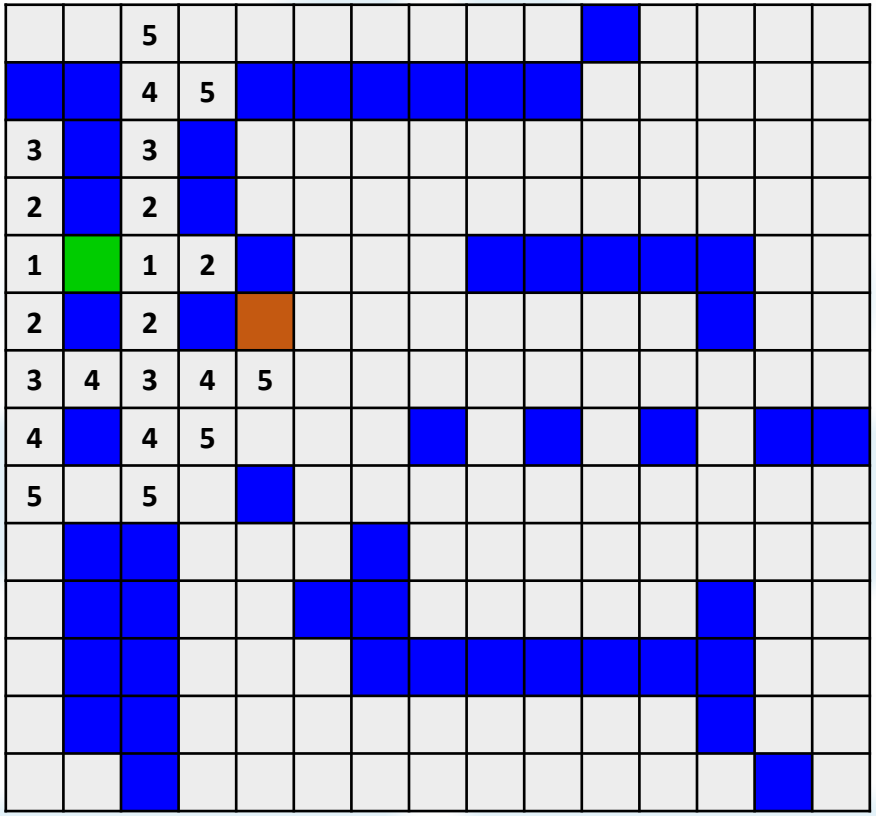
برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۴ واحد از نقطه شروع فاصله دارند.

کاربردهای صف

Wire Routing

Start pin

Stop pin



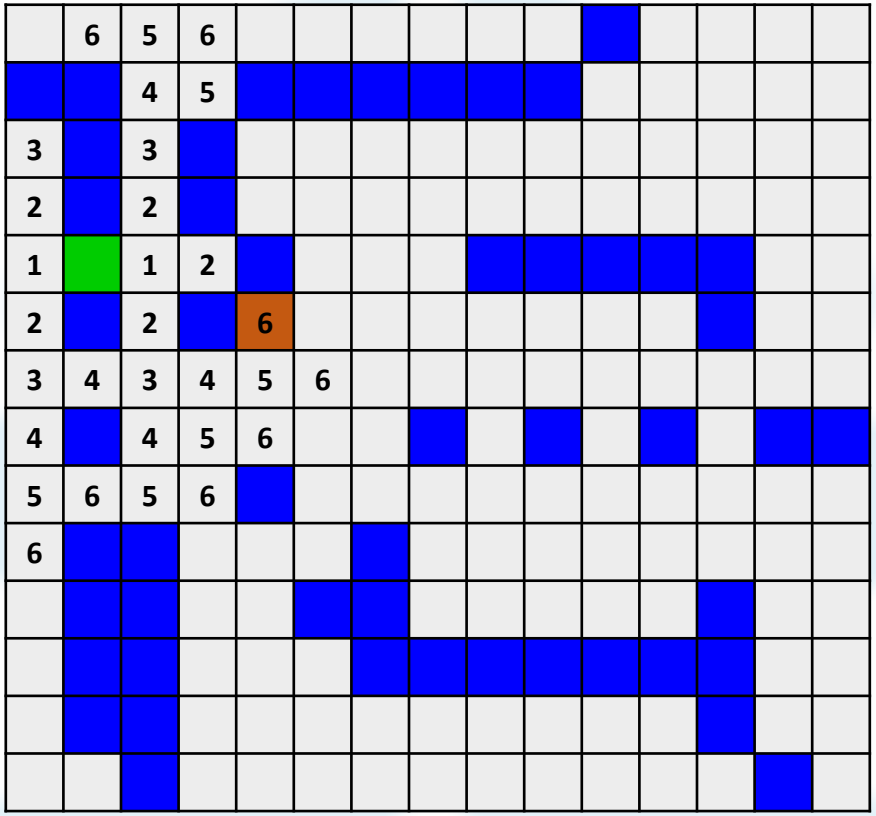
برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۵ واحد از نقطه شروع فاصله دارند.

کاربردهای صف

Wire Routing

Start pin

Stop pin



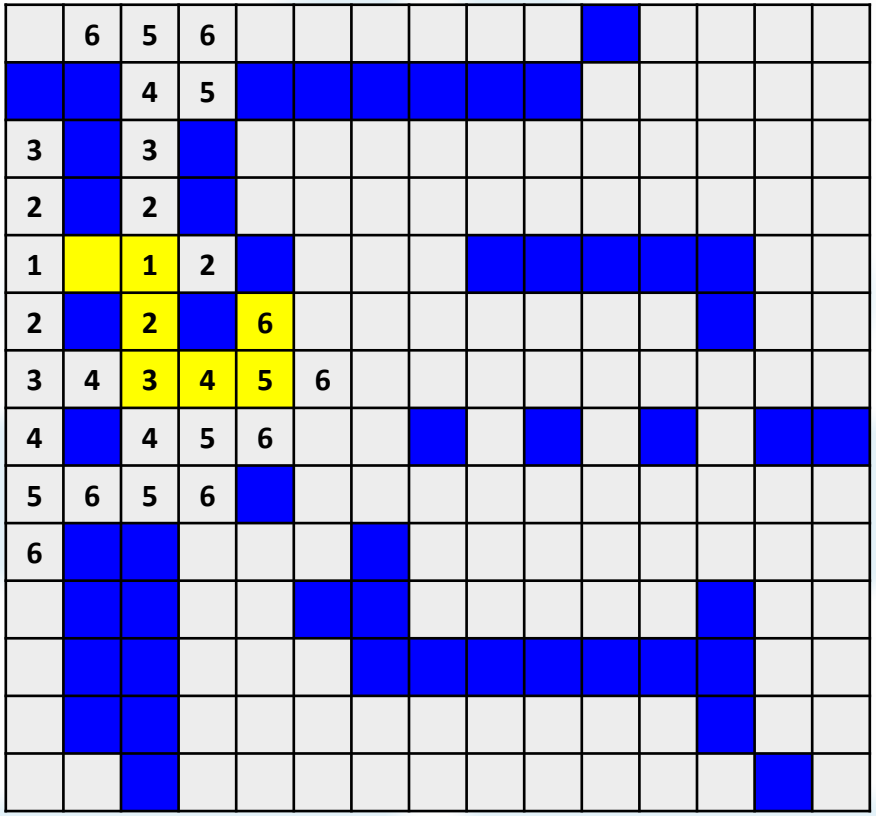
برچسب‌گذاری خانه‌هایی که برچسب ندارند و ۶ واحد از نقطه شروع فاصله دارند.

کاربردهای صف

Wire Routing

Start pin

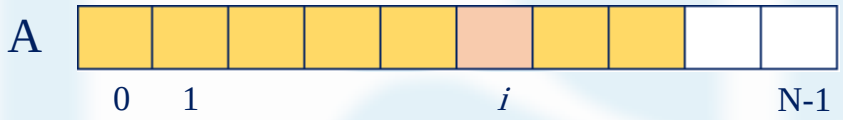
Stop pin



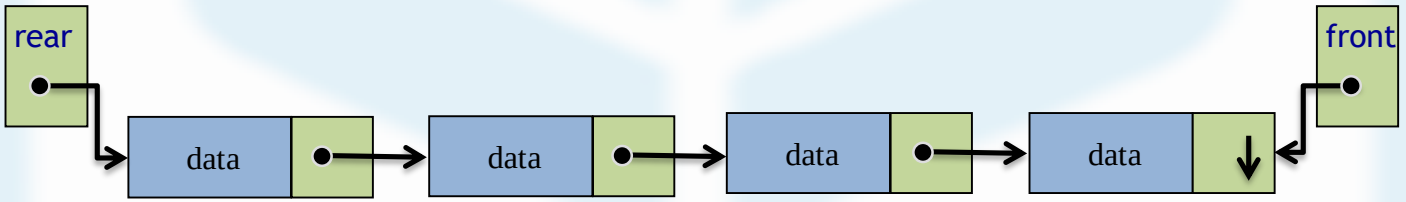
End pin reached. Traceback

پیاده سازی صف

❖ پیاده‌سازی صف براساس آرایه



❖ پیاده‌سازی صف براساس لیست پیوندی



نوع داده انتزاعی صف

نوع داده انتزاعی صف(Stack ADT)

تعریف صف:

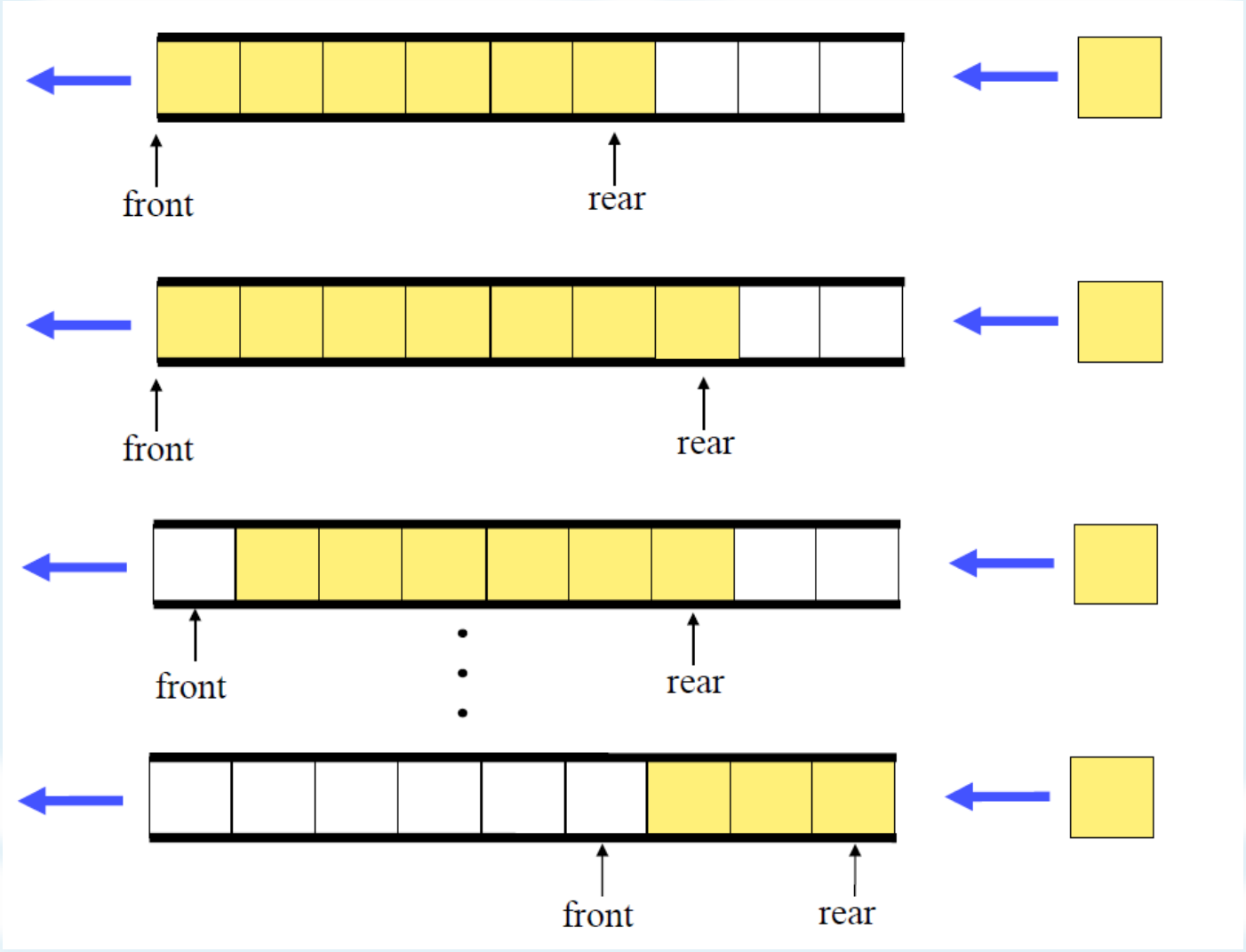
صف، لیستی خطی است که مجموعه‌ای از عناصر را طوری ذخیره می‌کند که طبق قانون **FIFO** عمل می‌کند.

عملیات اصلی:

- ایجاد صف خالی،
- امتحان خالی بودن صف،
- امتحان پر بودن صف (*)،
- اضافه کردن عنصر به انتهای صف،
- حذف عنصر از ابتدای صف،
- بازیابی عنصر ابتدایی صف (بدون حذف آن).

(*) برای پیاده‌سازی صف بر اساس لیست پیوندی، نیازی به امتحان پر بودن صف نداریم.

پیاده‌سازی صف بر اساس آرایه



پیاده‌سازی صف بر اساس آرایه

نکات کلیدی:

- تخصیص حافظه برای آرایه‌ای با اندازه ثابت جهت ذخیره‌سازی عناصر صف.
- تعریف متغیرهای صحیح *front* و *rear* به ترتیب برای اشاره به اندیس‌هایی از آرایه که متناظر با ابتدا و انتهای صف هستند.
- ذخیره‌سازی اولین عنصر صف در درایه‌ی با اندیس صفر از آرایه.
- یک واحد افزایش به مقدار *front* با حذف یک عنصر از صف،
- یک واحد افزایش به مقدار *rear* با اضافه شدن یک عنصر به صف،

پیاده‌سازی صف براساس آرایه

- **اعضای داده‌ای کلاس صف:**
  - اندازه ثابت برای آرایه
  - تخصیص حافظه برای آرایه‌ای با اندازه ثابت جهت ذخیره‌سازی عناصر صف.
  - تعریف متغیرهای صحیح *front* و *rear* با مقدار اولیه ۱-، به ترتیب برای اشاره به اندیس‌هایی از آرایه که متناظر با ابتدا و انتهای صف هستند.

پیاده‌سازی صف براساس آرایه

- **اعضای تابعی کلاس صف**:
  - ایجاد صف خالی،
  - امتحان خالی بودن صف،
  - امتحان پر بودن صف،
  - اضافه کردن عنصر به انتهای صف،
  - حذف عنصر از ابتدای صف،
  - بازیابی عنصر ابتدایی صف (بدون حذف آن).

پیاده‌سازی صف براساس آرایه

```
class ArrayQueue1:
    def __init__(self,n):
        self._N = n
        self._data = [None]*self._N
        self.front = -1
        self.rear = -1


    def is_empty(self):

    def is_full(self):

    def push(self, e):

    def front_element(self):

    def pop(self):

    def show(self, text):
```

پیاده‌سازی صف براساس آرایه

- **تابع ایجاد صف خالی،**

وظایف تابع:

۱- تعریف یک آرایه
۲- تعریف متغیرهای صحیح *front* و *rear* با مقدار اولیه ۱-، به ترتیب برای اشاره به اندیس‌هایی از آرایه که متناظر با ابتدا و انتهای صف هستند.

```
def __init__(self,n):
    self._N = n
    self._data = [None]*self._N
    self.front = -1
    self.rear = -1
```

پیاده‌سازی صف براساس آرایه

- تابع امتحان خالی بودن صف،

```
def is_empty(self):
    return self.front == -1
```

- تابع امتحان پر بودن صف،

```
def is_full(self):
    return self.rear+1 == self._N
```

- بازیابی عنصر ابتدایی صف (بدون حذف آن).

```
def front_element(self):
    if self.is_empty():
        return 0,None
    return 1, self._data[self.front]
```

پیاده‌سازی صف براساس آرایه

- تابع اضافه کردن عنصر به صف،

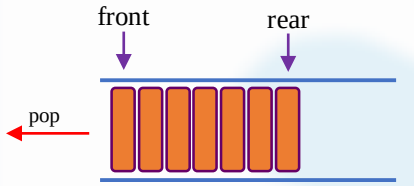
وضعیت صف قبل از اضافه شدن عنصر جدید

وضعیت صف بعد از اضافه شدن عنصر جدید

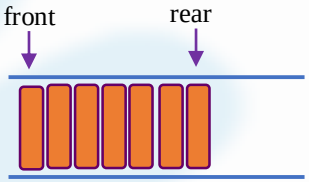
```
def push(self, e):
    if not self.is_full():
        self._data[self.rear+1] = e
        self.rear += 1
        if self.front ==  -1:
            self.front += 1
        return 1
    else:
        return 0
```

پیاده‌سازی صف براساس آرایه

- تابع حذف عنصر از صف،



وضعیت صف قبل از حذف عنصر



وضعیت صف بعد از حذف عنصر

```
def pop(self):
    if self.is_empty():
        return 0,None
    else:
        x = self._data[self.front]
        self.front += 1
        if self.front > self.rear:
            self.front = -1
            self.rear = -1
    return 1,x
```

پیاده‌سازی صف براساس آرایه

- تابع نمایش عناصر صف

```
def show(self, text):
    print(text, end = '')
    if self.front != -1:
        for x in range(self.rear,self.front-1,-1):
            print(self._data[x], end = '->')
        print()
```

پیاده‌سازی پشته براساس آرایه

- تابع نمایش عناصر صف

```
if __name__ == '__main__':
    Q = ArrayQueue1(5)

    print('elements pushed to queue:')
    for i in range(10):
        flag=Q.push(i)
        if (flag==0):
            print('Can\'t push new element(%d) on queue: Queue is full' % (i))
            break
        print(i)
    Q.show(Queue after elements are pushed:    ')
```

```
elements pushed to queue:
0
1
2
3
4

Can't push new element(5) on queue: Queue is full
Queue after elements are pushed:    4→3→2→1→0→
```

پیاده‌سازی صف براساس آرایه

- تابع نمایش عناصر صف(ادامه)

```
flag,element=Q.front_element()
if flag:
    print('\n front element:',element)

print('\nelements popped from queue:')
for i in range(10):
    flag, element =Q.pop()
    if (flag== 0):
        print('Can\'t pop  element from queue: Queue is empty')
        break
    print(element)
Q.show(Queue after elements are popped:')
```

```
front element: 0

elements popped from Queue:
0
1
2
3
4

Can't pop  element from queue: Queue is empty
Queue after elements are popped:
```

پیاده‌سازی صف براساس آرایه(با استفاده از قابلیت کلاس list)

کلاس ArrayQueue1 را می‌توان با استفاده ار قابلیت کلاس لیست پایتون بازنویسی کرد.

- برای ذخیره‌سازی عناصر صف، از شی‌ای از کلاس list پایتون استفاده می‌کنیم.
- برای اضافه و حذف کردن عنصر از توابع append و pop استفاده می‌کنیم.
- نیازی به تعریف متغیرهای صحیح *front* و *rear* نیست.

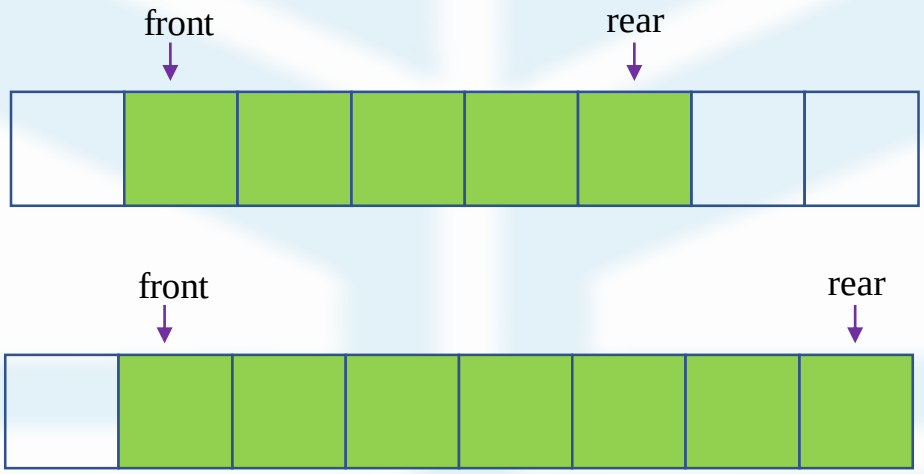
پیاده‌سازی صف براساس آرایه(با استفاده از قابلیت کلاس list)

```
class Empty(Exception):
    pass

class ArrayQueue2 :
    def __init__(self):
        self._data = []

    def __len__(self):
        return len(self._data)

    def is_empty(self):
        return len(self._data) == 0

    def push(self, e):
        self._data.append(e)

    def pop(self):
        if self.is_empty():
            raise Empty('Queue is empty ..')
        return self._data.pop(0)

    def front_element(self):
        if self.is_empty():
            raise Empty('Queue is empty')
        return self._data[0]

    def show(self, text):
        print(text, end='')
        for x in self._data:
            print(x, end='')
            print('←', end='')
        print()

if __name__ == '__main__':
    Q = ArrayQueue2()
    Q.push(5)
    Q.show('after push(5): ')
    Q.pop()
    Q.show('after pop(): ')
```

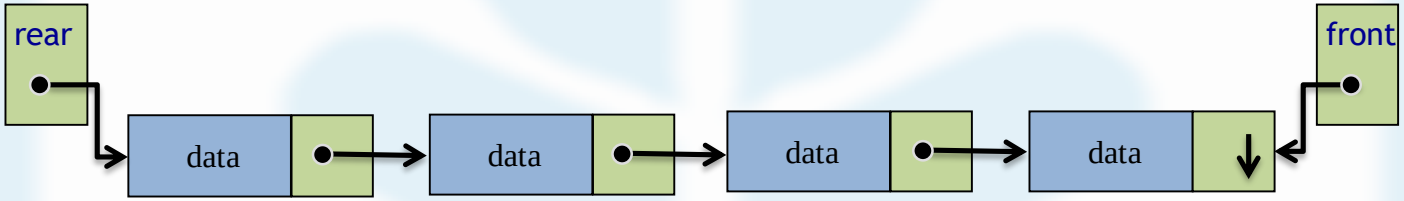
پیاده‌سازی صف براساس آرایه

معایب و راه حل پیاده‌سازی صف براساس آرایه

معایب	راه حل
اگر خانه ای از صف خطی خالی شود، دیگر قابل استفاده نیست.	۱- استفاده از صف حلقوی ۲- پیاده‌سازی براساس لیست پیوندی
به دلیل ثابت بودن اندازه آرایه، پس از پر شدن صف، نمی‌توان عنصر جدیدی را به آن اضافه نمود.	۱- پیاده‌سازی براساس لیست پیوندی



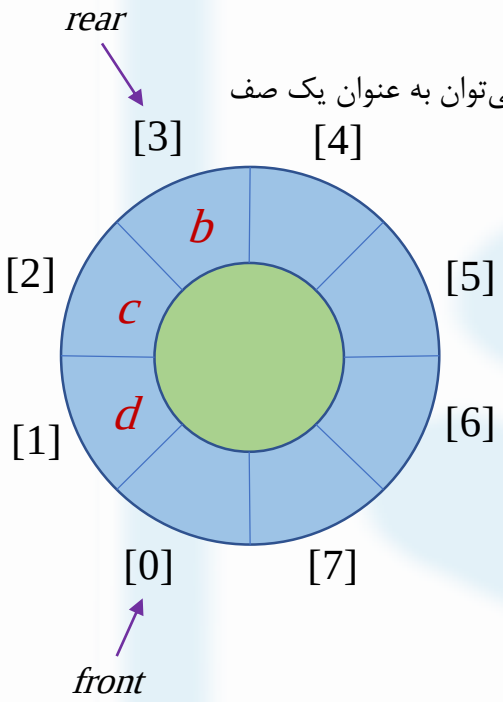
پیاده‌سازی صف براساس لیست پیوندی



صف حلقوی

تعریف صف حلقوی:

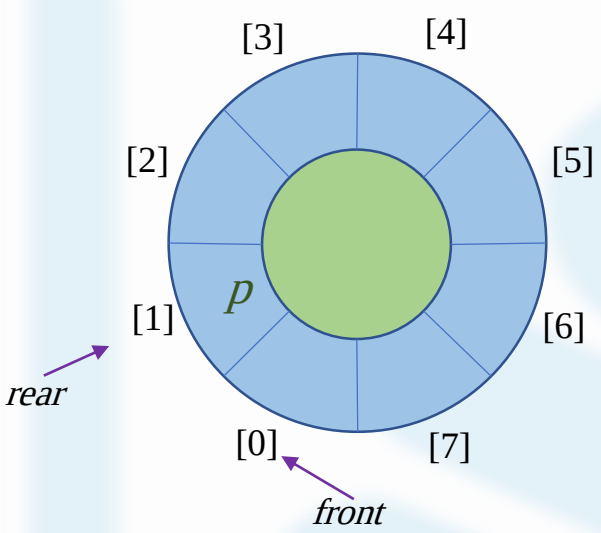
اگر انتهای یک آرایه را به ابتدای آن وصل کنیم، یک شکل حلقوی به دست می آید که از آن می‌توان به عنوان یک صف حلقوی برای ذخیره‌سازی اطلاعات استفاده نمود.



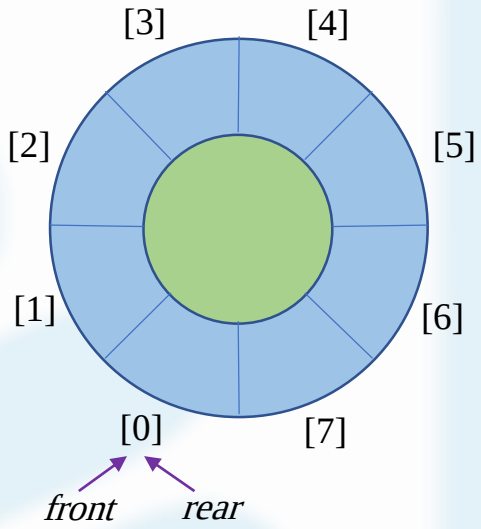
مشخصات صف حلقوی:

- اولین عنصر آرایه، بلافاصله بعد از آخرین عنصر آرایه قرار دارد(در جهت خلاف ساعتگرد).
- اگر آخرین عنصر آرایه پر شده باشد، در صورت خالی بودن عنصر اول آرایه، می‌توان مقدار جدید را در آن خانه اضافه کرد.
- از متغیرهای صحیح *front* و *rear*، با مقدار اولیه صفر، به ترتیب برای اشاره به ابتدا و انتهای صف حلقوی استفاده می‌شود.
- بعد از اضافه شدن یک عنصر جدید، یک واحد به متغیر *rear* اضافه می‌شود.
- بعد از حذف شدن یک عنصر، یک واحد به متغیر *front* اضافه می‌شود.
- نمی‌توان از شرط *rear*<*front* برای تشخیص پر بودن صف استفاده کرد.

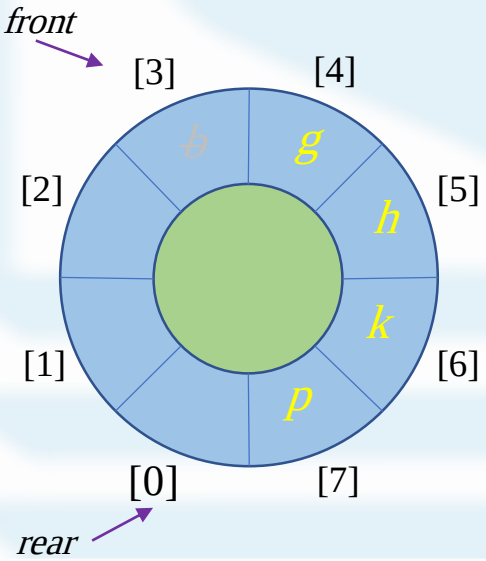
صف حلقوی



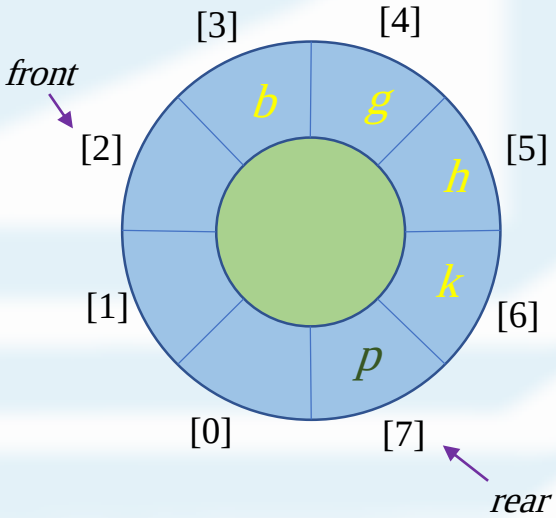
اضافه کردن اولین عنصر



صف خالی



حذف عنصر



اضافه کردن عنصر

پیاده‌سازی صف حلقوی براساس آرایه

```
class ArrayCircularQueue:
    def __init__(self,n):
        self._N = n
        self._data = [None]*self._N
        self.front = 0
        self.rear = 0
        self.last_opt= 0

    def is_empty(self):

    def is_full(self):

    def Dequeue(self, e):

    def front_element(self):

    def Enqueue(self):

    def __len__(self):

    def show(self, text):
```

پیاده‌سازی صف حلقوی براساس آرایه

- **تابع ایجاد صف حلقوی خالی،**

وظایف تابع:

۱- تعریف یک آرایه

۲- تعریف متغیرهای صحیح *front* و *rear* با مقدار اولیه 0، به ترتیب برای اشاره به اندیس‌هایی از آرایه که متناظر با ابتدا و انتهای صف هستند.

۳- تعریف متغیر *last_opt* با مقدار اولیه 0.این متغیر به همراه یک شرط دیگر پر یا خالی بودن صف را مشخص می کند.

```
def __init__(self,n):
    self._N = n
    self._data = [None]*self._N
    self.front = 0
    self.rear = 0
    self.last_opt= 0
```

پیاده‌سازی صف حلقوی براساس آرایه

- تابع امتحان خالی بودن صف،

```
def is_empty(self):
    return (self.front == self.rear)  and (last_opt == 0)
```

- تابع امتحان پر بودن صف،

```
def is_full(self):
    return (self.front == self.rear)  and (last_opt == 1)
```

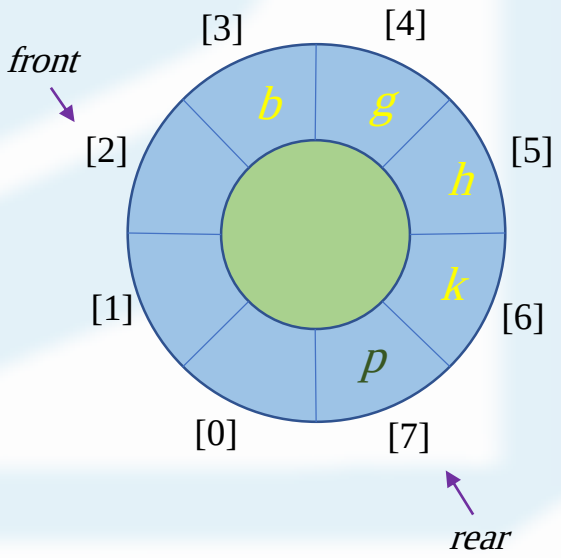
- بازیابی عنصر ابتدایی صف (بدون حذف آن).

```
def front_element(self):
    if self.is_empty():
        return 0,None
    return 1, self._data[self.front]
```

پیاده‌سازی صف حلقوی براساس آرایه

- تابع اضافه کردن عنصر به صف،

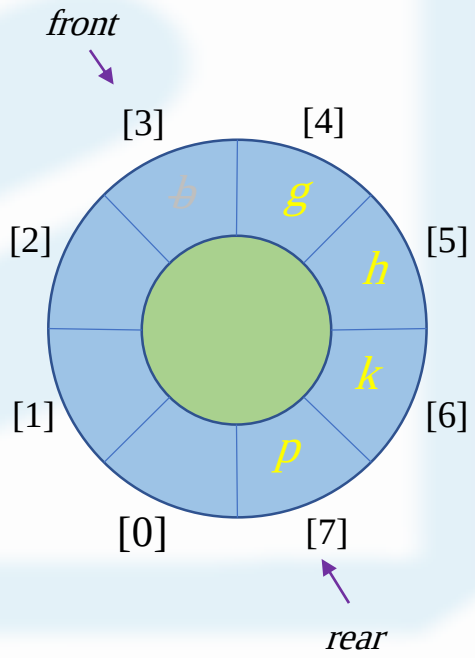
```
def Enqueue(self, e):
    if not self.is_full():
        self._data[self.rear] = e
        self.rear = (self.rear+1) % self._N
        if self.front == self.rear:
            self.last_opt = 1
        return 1
    else:
        return 0
```



پیاده‌سازی صف حلقوی براساس آرایه

- تابع حذف عنصر از صف،

```
def Dequeue(self):
    if self.is_empty():
        return 0,None
    else:
        x = self._data[self.front+1% self._N]
        self.front = (self.front +1) % self._N
        if self.front == self.rear:
            self.last_opt= 0
    return 1,x
```



پیاده‌سازی صف حلقوی براساس آرایه

- تابع نمایش عناصر صف

```
def __len__(self):
    if self.is_empty():
        return 0
    if self.rear > self.front:
        return self.rear-self.front
    return self.rear+ len(self._data) -self.front


def show(self, text):
    print(text, end = '')
    if not self.is_empty():
        for x in range(self.front, self.front +len(self)):
            print(self._data[x], end = '->')
        print()
```

[1] Text Book, *Data Structures and Algorithms in Python* [Goodrich, Tamassia & Goldwasser 2013.
[2] http://xpzhang.me/teach/DS18_Fall
[3] https://www.slideshare.net/sohidulislam6/circular-queue