

# Queue

# Queue



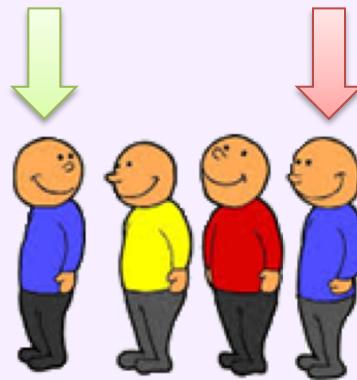
Queue  
?

<http://www.gograph.com/vector-clip-art/queue.html>

# Queue ແລວຄອຍ



front  
head      rear  
tail



**FIFO List**  
**FirstInFirstOut**



I need some !  
Where should I go ?

Next !  
Oh my turn ☺  
Who ?

# Queue Implementations



1. **(Python) list Implementation**
2. **(Python) deque (double-ended queue)**
3. **Linked Queue (Subset of Linked List)**

# Logical Abstract Data Type

Logical ADT :

1. Data : ของมีลำดับ มีปลาย หัว front ท้าย rear

2. Methods :

1. init()      init empty queue

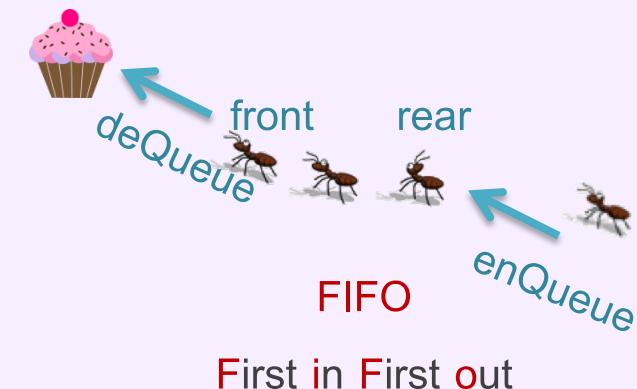
2. enQueue(i)    insert i ที่ rear / tail

3. i = deQueue() return + เอาของที่ front / head ออก

4. b = isEmpty() queue empty ?

5. b = isFull() queue full ?

6. i = size()      return จำนวนของใน queue



# Queue Implementation

Logical ADT :

Data Implementation ?

1. Data : ของมีลำดับ มีปลายหัว ท้าย Python List

2. Methods :

1. init()      init empty Q       $Q = []$

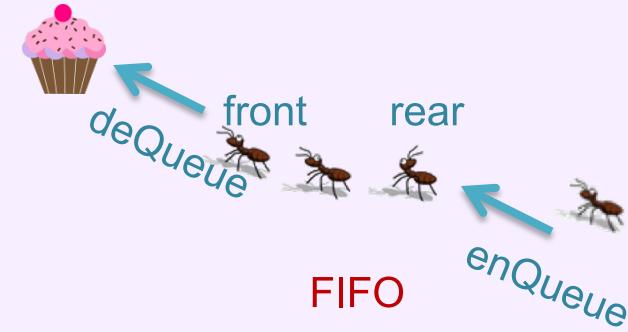
2. enQueue(i)    insert i ที่ rear       $Q.append(i)$  ใส่ท้าย

3. i = deQueue() return + เอาของที่ front ออก     $i = Q.pop(0)$  อันแรก

4. b = isEmpty() Q empty ?       $Q == [] ?$

5. b = isFull() Q full ?      list expansion -> Python List implementation

6. i = size()      return จำนวนของใน Q       $i = len(Q)$



First in First out



# Queue Data Implementation

## 1. Data :

`__init__()` : constructor ให้ค่าตั้งต้น



2 underscores    2 underscores

# Data Implementation : `__init__()`

## 1. Data Implementation : Queue แบบอยู่มีปลาย หัว ท้าย -> Python List

ทำใน constructor  
`__init__()`

self คือ object ที่เรียก method ในแต่ละครั้ง  
เช่น `q = Queue()`  
self หมายถึง q เมื่อเรียก `q = Queue(q)`  
self จะถูก pass เป็น arg. ตัวแรก โดยอัตโนมัติ

docstring : ใน triple quote  
`print(Queue.__doc__)`  
→ docstring

constructor function  
ถูกเรียกโดยอัตโนมัติเมื่อ  
instantiate instance ใหม่

instantiate instance (object) ใหม่  
โดยไม่ pass argument

```
class Queue:  
    """ class Queue  
        create empty Queue  
    """  
  
    def __init__(self):  
        self.items = []
```

`q = Queue()`

`print(q.items) []`

constructor :  
ใช้ define  
Instance Attributes ได้  
ในตัวอย่างนี้มี attribute  
เดียวคือ items

items :  
Instance Attributes  
สำหรับแต่ละ instance

# Default Argument

```
class Queue:  
    """ class Queue  
        default : empty Queue/  
        Queue([list])  
    """  
  
    def __init__(self, list = None):  
        if list == None:  
            self.items = []  
        else:  
            self.items = list
```

default argument  
ถ้าไม่มีการ pass arg. มา  
list = None  
ถ้า pass arg. มา  
list = ตัวที่ pass มา  
  
Object None  
ใช้เช็ค obj identity

```
q = Queue()  
print(q.items)                []  
  
q1 = Queue(['A', 'B', 'C'])  
print(q1.items)               ['A', 'B', 'C']
```

ไม่เหมือนกับ C++ & Java ใน Python ไม่สามารถมี constructor หลายตัวได้

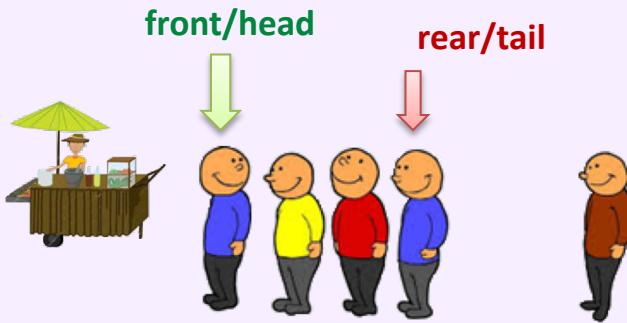


## Queue Operation Implementation

### 2. Methods :

1. `__init__()` : ให้ค่าตั้งต้น
2. ใส่ `enQueue()` : ด้านท้าย **rear**
3. เอาออก `deQueue ()` : ด้านหน้า **front**
4. `isEmpty()` : queue ว่าง ?
5. `size()` : มีของกี่อัน

# enQueue()



```
q = Queue()  
print(q.items)  
q.enqueue('A')  
print(q.items)  
q.enqueue('B')  
print(q.items)  
q.enqueue('C')  
print(q.items)
```

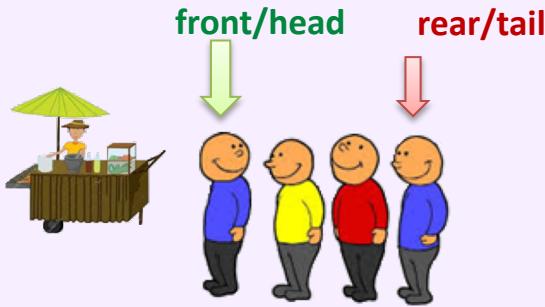
```
[]  
['A']  
['A', 'B']  
['A', 'B', 'C']
```

ใน class Queue:

```
def enqueue(self, i):  
    self.items.append(i) # insert i ที่ท้าย list
```



# deQueue()



```
print(q.items)  
print(q.deQueue())  
print(q.items)  
print(q.deQueue())  
print(q.items)  
  
q.deQueue()
```

ໃນ class Queue:

```
def deQueue(self):
```

```
    return self.item.pop(0) # pop out index 0
```

deQueue must check  
Queue Underflow

[ 'A' , 'B' ]

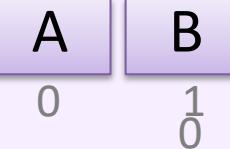
A

[ 'B' ]

B

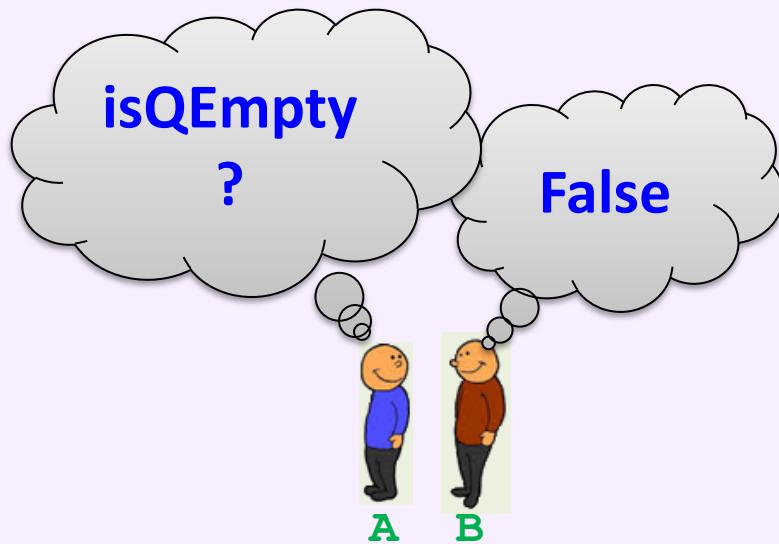
[]

front/head rear/tail



# error  
# Queue Underflow

# isEmpty()



ใน class Queue:

```
def isEmpty(self):  
    return self.items == []  
#return len(self.items) == 0
```



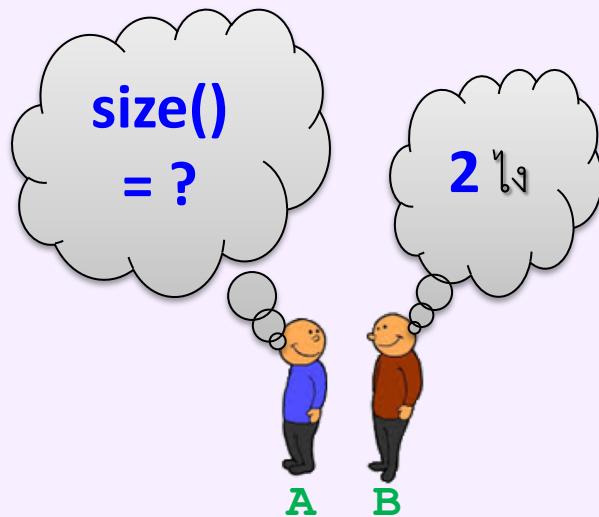
```
print(q.items)
```

```
print(q.isEmpty())
```

```
[ 'A' , 'B' ]
```

```
false
```

# size()



ใน class Queue:

```
def size(self):  
    return len(self.items)
```

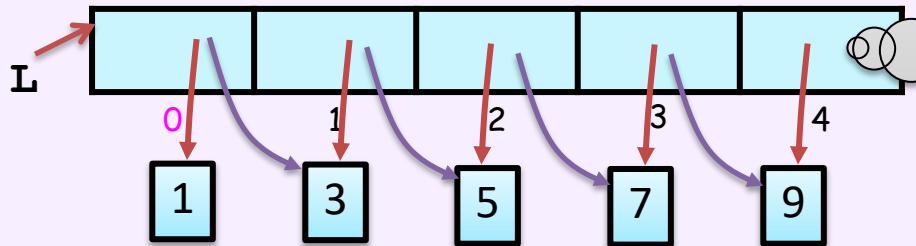
```
print(q.items)  
print(q.size())
```

[ 'A' , 'B' ]

2

# Python list Internal

L = [ 1, 3, 5, 7, 9 ]



Python List :

ข้างใน เป็น array

:  
index ต่อ กัน  
memory ติด กัน

Insert / Delete

ตั้นๆ list ແພງ

O(n)

shift out /  
shift in

print(L)

[1, 3, 5, 7, 9]

L.pop(0)

1

print(L)

[3, 5, 7, 9]

L.insert(0,2)

[2, 3, 5, 7, 9]

print(L)

insert ตั้น list :  
ต้องทำอะไรบ้าง ?

Insert / Delete หั้ง 2 ปลาย

ใช้ deque ถูกกว่า

(double-end queue)

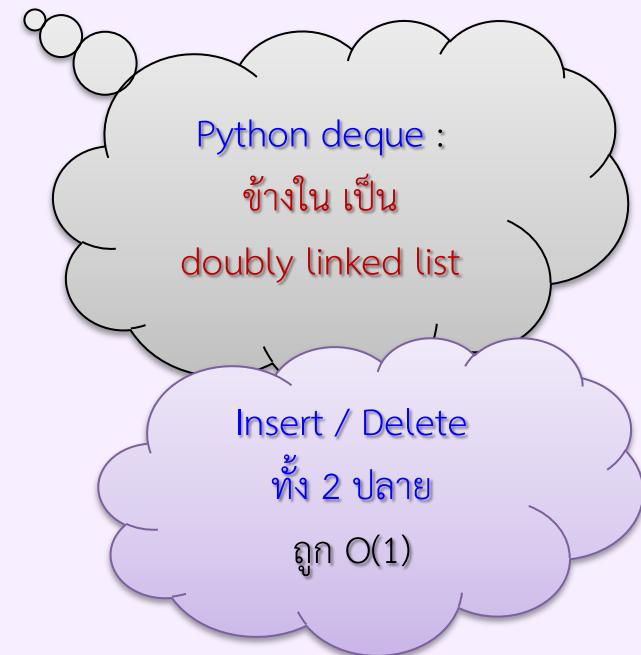
ข้างใน : doubly linked list

(เรียนในเรื่องถัดไป)

# Python deque (double-ended queue)



Operation	Average Case	Amortized Worst Case
Copy	$O(n)$	$O(n)$
append	$O(1)$	$O(1)$
appendleft	$O(1)$	$O(1)$
pop	$O(1)$	$O(1)$
popleft	$O(1)$	$O(1)$
extend	$O(k)$	$O(k)$
extendleft	$O(k)$	$O(k)$
rotate	$O(k)$	$O(k)$
remove	$O(n)$	$O(n)$



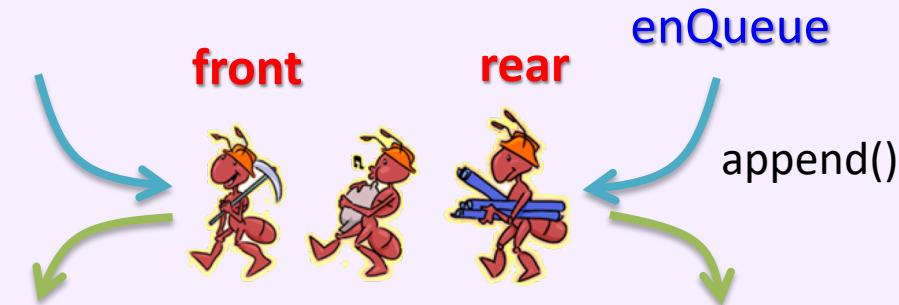
<https://wiki.python.org/moin/TimeComplexity>



## Queue Implementations

1. [\(Python\) List Implementation](#)
2. [\(Python\) deque \(double-ended queue\)](#)
3. [Linked Queue \(Subset of Linked List\)](#)

# Queue Implementation using Python deque (double-ended queue)



**deQueue**

`popleft()`

**d e f g h**

```
>>> from collections import deque  
>>> d = deque('def')  
>>> d  
deque(['d', 'e', 'f'])
```

```
>>> dd = deque()  
>>> dd  
deque([])
```

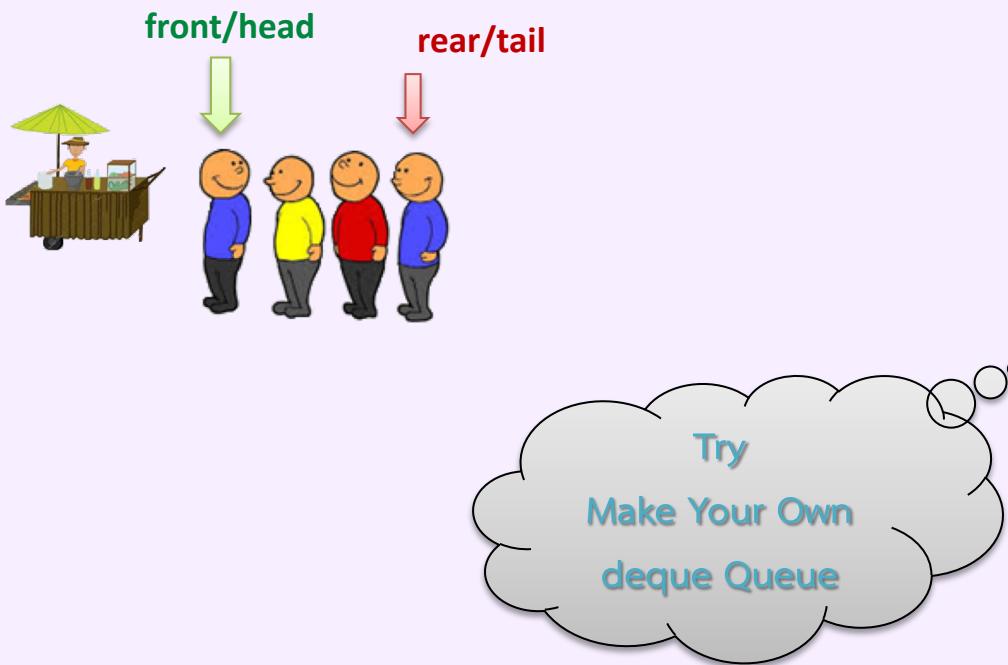
```
>>> d.append('g')  
>>> d.append('h')  
>>> d  
deque(['d', 'e', 'f', 'g', 'h'])
```

```
>>> pop1 = d.popleft()  
>>> pop2 = d.popleft()  
>>> d  
deque(['f', 'g', 'h'])
```

```
>>> print(pop1, pop2)  
d e
```

```
>>> len(d)  
3
```

# Queue() : Python deque (double-ended queue)



```
from collections import deque

class Queue: # use deque

    def __init__(self):
        self.items = deque()

    def enqueue(self, i):
        self.items.append(i)

    def dequeue(self):
        return self.items.popleft()

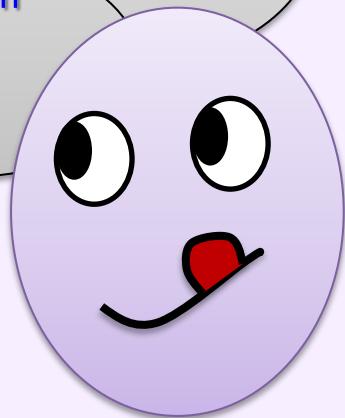
    def isEmpty(self):
        return len(self.items) == 0

    def size(self):
        return len(self.items)
```

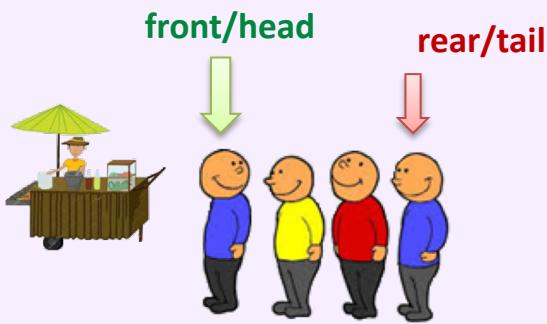
# Test Your deque Queue



1. สร้าง list อะไรก็ได้ เช่น [5, 7, 6, 3, 8, 4]
2. สร้าง q โดยใช้ class ที่สร้างขึ้นในหน้าที่แล้ว
2. loop enqueue element ใน list พร้อม print q
3. loop dequeue ทีละตัวจนหมด พิมพ์ของที่เอาออก  
และ q ที่เหลือ



# Queue() : Python deque (double-ended queue)



```
from collections import deque

class Queue: # use deque

    def __init__(self):
        self.items = deque()

    def enqueue(self, i):
        self.items.append(i)

    def dequeue(self):
        return self.items.popleft()

    def isEmpty(self):
        return len(self.items) == 0

    def size(self):
        return len(self.items)
```



## Queue Implementations

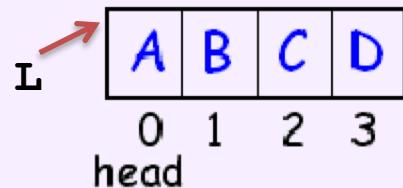
1. [\(Python\) List Implementation](#)
2. [\(Python\) deque \(double-ended queue\)](#)
3. [Linked Queue \(Subset of Linked List\)](#)

# Linked List



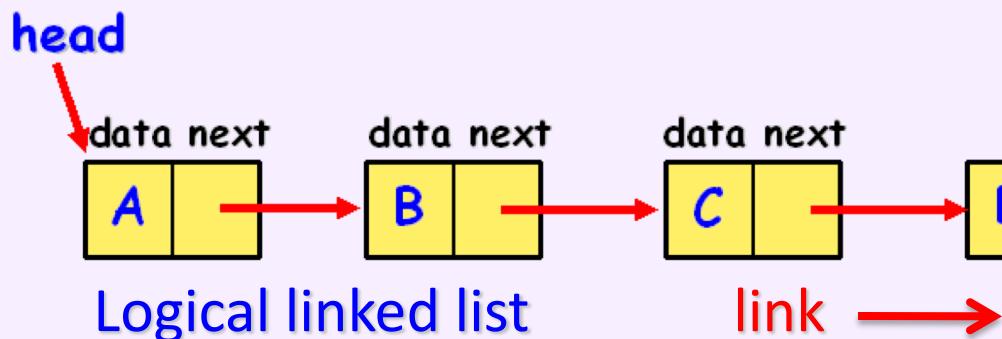
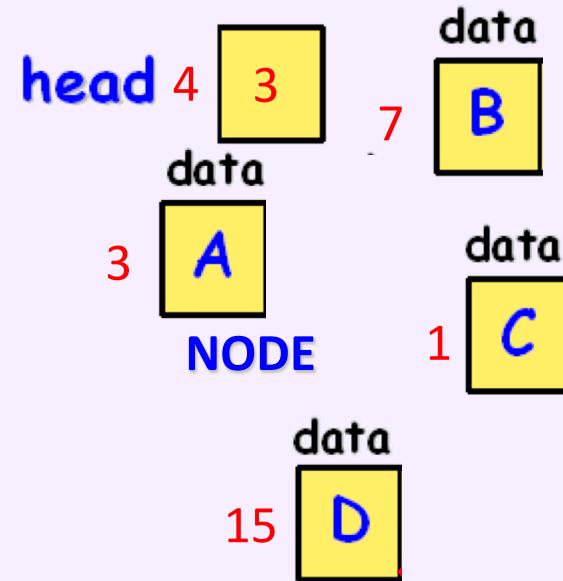
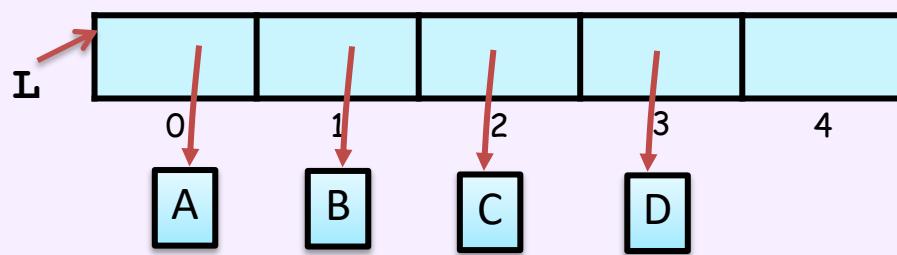
unfix positions

order?



Implicit Array  
Sequential Array

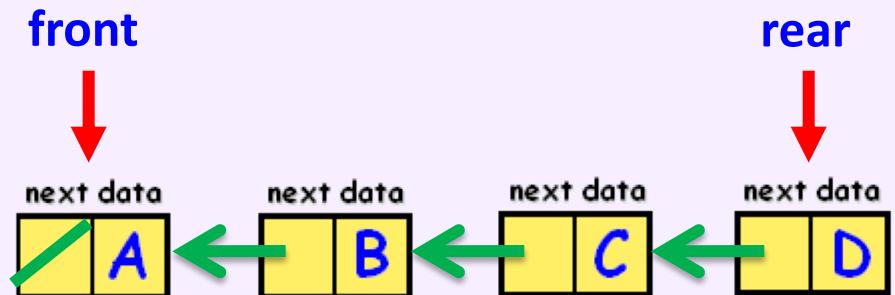
Problem : fix positions



Linked List

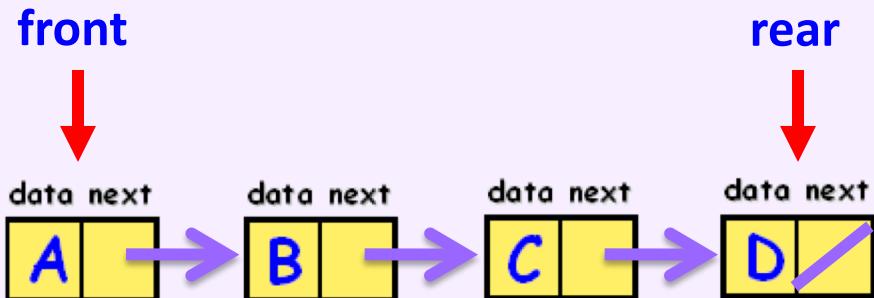
คำว่า Logical หมายความว่าในความคิดของเรามา เช่น link แทนด้วยลูกศร แทนการเชื่อมโยงกัน ในการ implement จะริง (physical) ลูกศรอาจเป็นได้หลายอย่าง เช่น pointer หรือ index ของ array

# Linked Queue



How do they link?

Support every operations ?



enQueue ?  
(insert)

deQueue ?  
(delete)

Every operations ?