

## ③ Deque Datastructure

Doubly ended queue. we can do insertion and deletion from both end.

Insertfront(), Insertrear(), Deletefront(),  
DeleteRear()

Additional operations :

getFront(), getRear(), isFull(), isEmpty()  
& size()

Ex

insertFront(10), insertFront(20) 20 | 10

insertRear(30)  $\Rightarrow$  20 | 10 | 30

insertRear(40)  $\Rightarrow$  20 | 10 | 30 | 40

deleteFront()  $\Rightarrow$  10 | 30 | 40

deleteRear()  $\Rightarrow$  10 | 30

### • Applications of Dequeue

- ① Act as both stack and queue.
- ② maintaining history of action
- ③ A steal process scheduling algorithm  
 $\Rightarrow$  used in multiple processor and have their own queue. Here processes fetches job from rear of another's processes queue.
- ④ Implementing a priority queue with two types of priorities
- ⑤ maximum / minimum of all subarrays of size R.

## 3) Deque Python

from collections import deque

All insert and delete from both ends

is  $O(1)$  in nature.

$d = deque()$   $\Rightarrow$  creates empty  
deque

$d = deque([1, 2, 3]) \Rightarrow$  creates  
deque from  
list

$d.append(10)$   $\Rightarrow$  insert at rear

$d.appendleft(20)$   $\Rightarrow$  insert at front

$d.pop()$   $\Rightarrow$  delete from end

$d.popleft()$   $\Rightarrow$  delete from front.

$d.insert(2, 10)$   $\rightarrow$  element  
 $\downarrow$  index

$d.count(10)$   $\rightarrow$  element

$d.remove(10)$

$\rightarrow$  remove first occurrence

$d.extend([50, 10]) \Rightarrow$  append a list into  
deque

$d.extendleft([50, 10]) \Rightarrow$  append to left  
side

Some new functions

$d.rotate(2) \Rightarrow$  clockwise rotation of  
2 elements

$d.rotate(-2) \Rightarrow$  anti-clockwise 2 elements

d-reverse()  $\rightarrow$  reverse of deque

• Index based Access

d[2]  $\rightarrow$  100

d[2] = 10 update

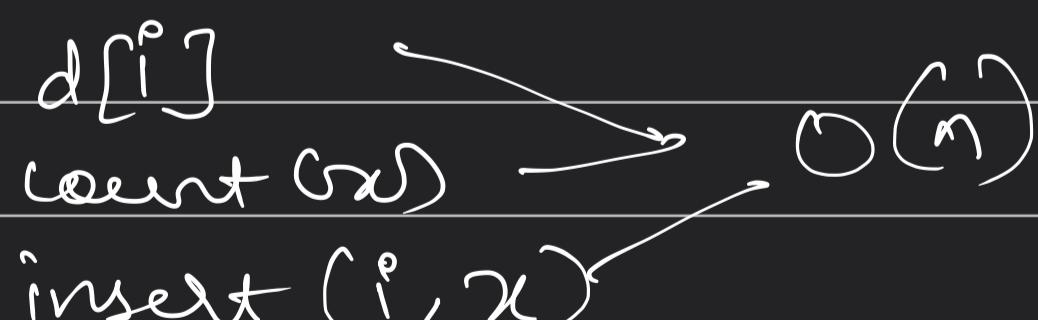
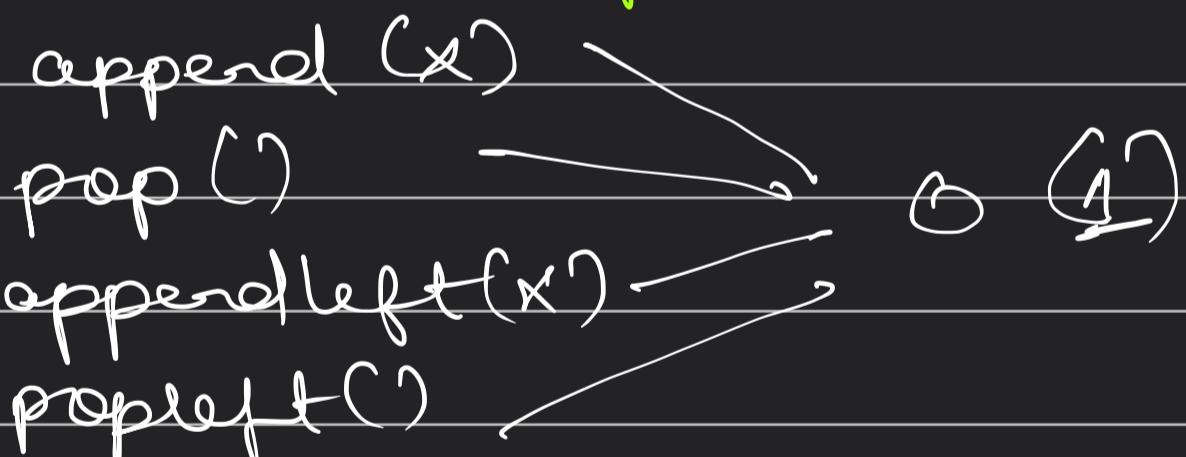
d[0]  $\rightarrow$  returns first

d[-1]  $\rightarrow$  returns last

Note:  $\rightarrow$  Slicing is not allowed in Deque

Deque internally utilizes Doubly Linked List as Doubly LL also, O(1) operation for insertion & deletion

• Time Complexities



rotate( $\delta$ )  $\rightarrow$  O(|abs( $\delta$ )|)

extend(l)  $\rightarrow$  O(len(l))

extendleft(l)  $\rightarrow$  O(len(l))  
 $\hookrightarrow$  dependent on length of list we provide

2 Data structure with min / max op.

Question is to design a DS with  
min / max

from collections import deque

class MyDS:

def \_\_init\_\_(self):  
 self.dq = deque()

def insertMin(self, x):  
 self.dq.appendleft(x)

def insertMax(self, x):  
 self.dq.append(x)

def extractMin(self):  
 return self.dq.popleft()

def extractMax(self):  
 return self.dq.pop()

def getMin():  
 return self.dq[0]

def getMax():  
 return self.dq[-1]

## Implementation of Deque using linked list.

class Node :

```
def __init__(self, data):  
    self.data = data  
    self.next = None  
    self.prev = None
```

class myDq :

```
def __init__(self):  
    self.front = None  
    self.rear = None  
    self.sz = 0
```

def isEmpty (self) :

return self.sz == 0

def isFull (self) :

return self.sz == K

def insertFront (self, x) :

node = Node(x)

if self.rear is None :

self.rear = node

self.front = node

else :

self.rear.next = node

node.prev = rear

rear = node

self.size += 1

def deletefront():

if self.isEmpty():

return -1

temp = self.front

front = front.next

if front is None: # One node only

rear = None

else

front.prev = None

temp.next = None # Severs

return temp.data.

existing connection

The logic now goes same for  
delete rear & insert front.

