② 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) ⇒ 20 | 10 | 30

insertRear(40) ⇒ 20 | 10 | 30 | 40

deleteFront() ⇒ 10 | 30 | 40

deleteRear() ⇒ 10 | 30

- Applications of Dequeue

① Act as both stack and queue.

② maintaining history of action

③ A steal process scheduling algorithm

⇒ used in multiple processor and have their own queue. Here processor fetches job from rear of another processors queue.

④ Inputting a priority queue with two types of priorities

⑤ maximum / minimum of all subarrays of size k.

Deque Python

```
from collections import deque
```

All insert and delete from both ends is O(1) in nature.

d = deque() ⇒ create empty deque

d = deque([1,2,3]) ⇒ create deque from list

d.append(10) ⇒ insert at rear

d.appendleft(20) ⇒ insert at front

d.pop(30) ⇒ delete from end

d.popleft() ⇒ delete from front.

d.insert(2, 10) → element, → index

d.count(10) → element

d.remove(10) → remove first occurence

d.extend([50,10]) ⇒ append a list into deque

d.extendleft([50,10]) ⇒ append to left side

Some new functions

d.rotate(2) ⇒ clockwise rotation of 2 elements

d.rotate(-2) ⇒ anticlockwise 2 elements

d.reverse() → reverse of deque

- Index based Access

d[2] → 100

d[2] = 10 update

d[0] → returns First

d[-1] → returns Last

Note :→ Slicing is not allowed in Deque

Deque Internally utilizes Doubly linked list as Doubly () allows O(1) operation for insertion & deletion

- Time Complexities

append (x)
pop ()
appendleft(x) → O(1)
popleft()

d[i]
count (x) → O(n)
insert (i, x)

rotate (k) → O(abs (k))

extend (l)
extendleft (l) → O(len(l))
↳ dependent on length of list we provide

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 insertRear(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.sz += 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           # removing any existing connection
    return temp.data
```

The logic now goes same for deleteRear & insertFront.