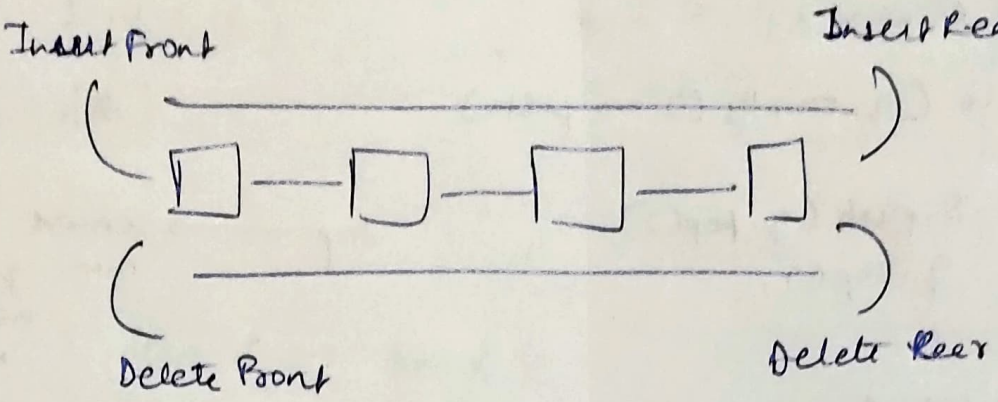
Deque Data Structure

(1) Deque (pronounced deck) → means doubly ended queues.



(2) Can be implemented using Array & LinkedList. (DLL only)*

* DLL only bcz, single LL cannot implement deletion in rear in O(1).

→ For Array, circular array is used, normal array can't be used.

(3) Deque can be used as Stack (or) Queue.

Maintain history of actions

Steal Process Scheduling Algorithm

other OS problems

Array Implementation of Deque

insertFront()
insertRear()
deleteFront()
deleteRear().

Extra
size.
isFull.
isEmpty.

Core implementation

```
class Deque {
    int size, capacity;
    int[] arr;

    Deque(int c) {
        capacity = c;
        size     = 0;
        arr      = new int[cap];
    }
    ------
}
```

Other functions implem (Simple Implementation)

Rear → O(1)
Front → O(n).

```
bool isFull() { return (size == cap); }
bool isEmpty() { return (size == 0); }

void deleteRear()
{ if (isEmpty()) return;
  size--;
}

void insertRear()
{ if (isFull()) { return; }
  arr[size] = x;
  size++;
}
```

```
int getRear(){
    if (isEmpty()) return;
    return arr[size-1];
}
```

```
void insertFront(int x)
{ if(isFull()) return;
  for(int i=size-1; i>=0; i--)
  { arr[i+1] = arr[i]; }
  arr[0] = x;
  size++;
}
```

```
void deleteFront()
{ if (isEmpty()) return;
  for(int i=0; i<size-1; i++)
    { arr[i] = arr[i+1]; }
  size--;
}
```

```
int getFront() {
    if(isEmpty()) return -1;
    else return 0;
}
```

Efficient func (Circular buffer & % used)

(All operations O(1))