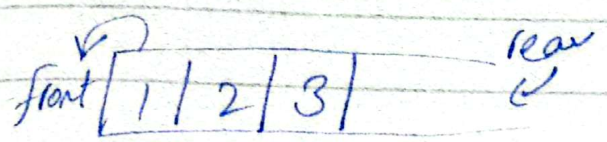
Queues

FIFO (First in First out)

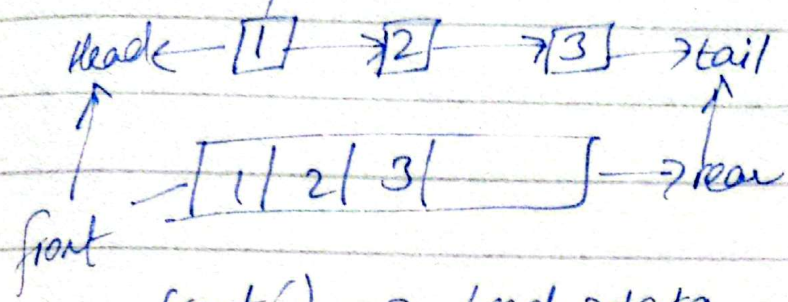
Queue < start / End



Operations:
O(1)
① push → rear/back
② pop → front
③ front → front element

enqueue
dequeue } different
deque

→ Queue Using Linked List



front() → head→data
push() → insert data at tail of LL
pop() → delete data from head of LL.

```
class Node{
int data;
Node* next;
Node(int val){
data=val;
next=NULL;
}
};

class Queue{
Node* head;
Node* tail;
Queue(){
head=tail=NULL
}
```

① void push()
② void pop()
③ int front()
④ bool empty

```
① void push(int data){
    Node* newNode= new Node(data)
    if(empty()){
        head=tail = newNode }
    else
        tail→next=newNode
        tail=newNode
}

② void pop(){
    if(empty()){
        cout<<"LL is Empty";
        return;
    }
    Node* temp =head;
    head= head→next;
    delete temp;
```

③ int front(){
if(empty()){
cout<<"it is Empty";
return -1
}
return head->data
}

④ bool empty(){
return head==NULL;
}

→ Queue Using STL

queue <int> q

All the built-in functions are used rather than building them from scratch.

Deque (Data structure)

↳ Double Ended Queue

Double functionalities:

push → back, front
pop → front, back
front → front, back

Exists built-in

deque <int> dq

dq.push_back()
dq.push_front(3)
—.pop_back(4)
—.pop_front(5)