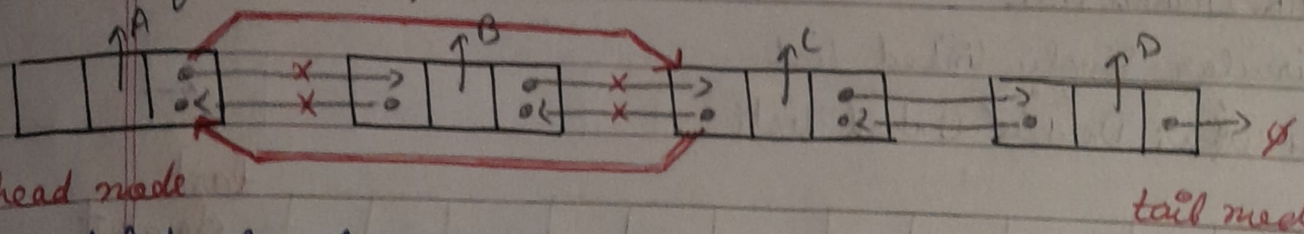
Deletion of a node, IN B/W



① To delete B, A·next → C·previous A·next·next

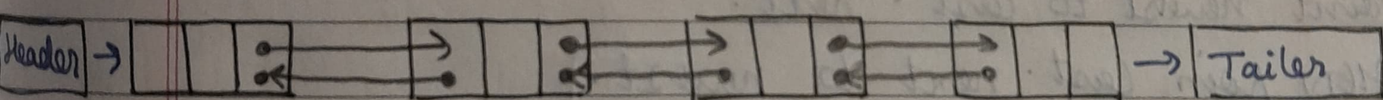
② C·previous → A·next ~~Eeprev~~ C·prev·prev

③ Decrease size = size−1.

Complexity of insertion

Insertion at head O(1)	Deletion at head
Insertion at tail (O(n)	Deletion at tail
Insertion in b/w O(n)	Deletion in b/w

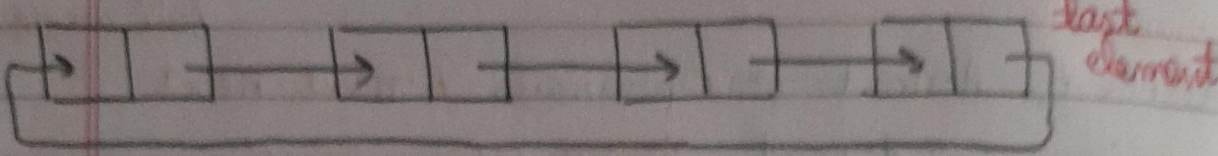
★ TO GENERALISE INSERTION, insert_element(e, pred, succ)

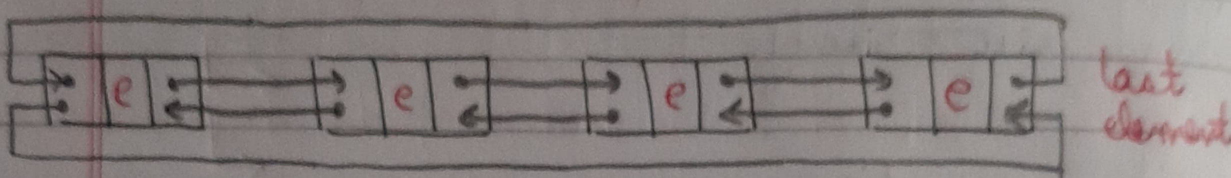



```
def insert_in_b/w(self, e, ~~xxx~~ predecessor, successor)
    newest = Self.Node(e, predecessor, successor)
    predecessor.next = newest
    successor.prev = newest
    Self.Size += 1
    return newest
```

CIRCULAR LINKED LIST

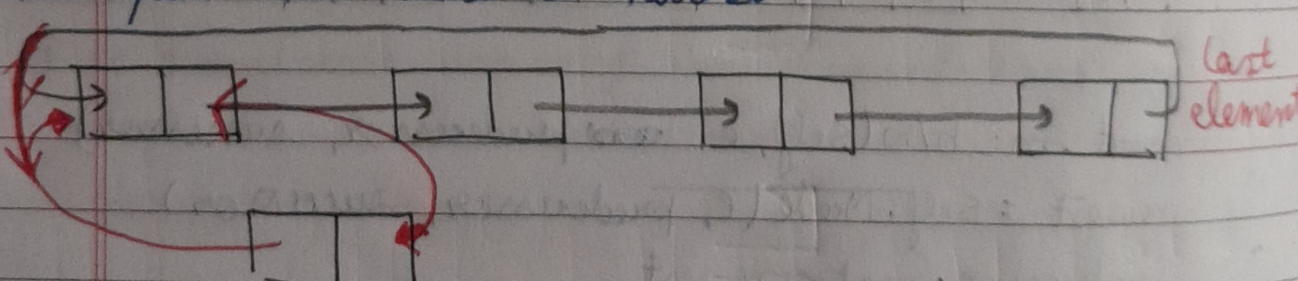
① Circular Singly linked list
② Circular Double linked list

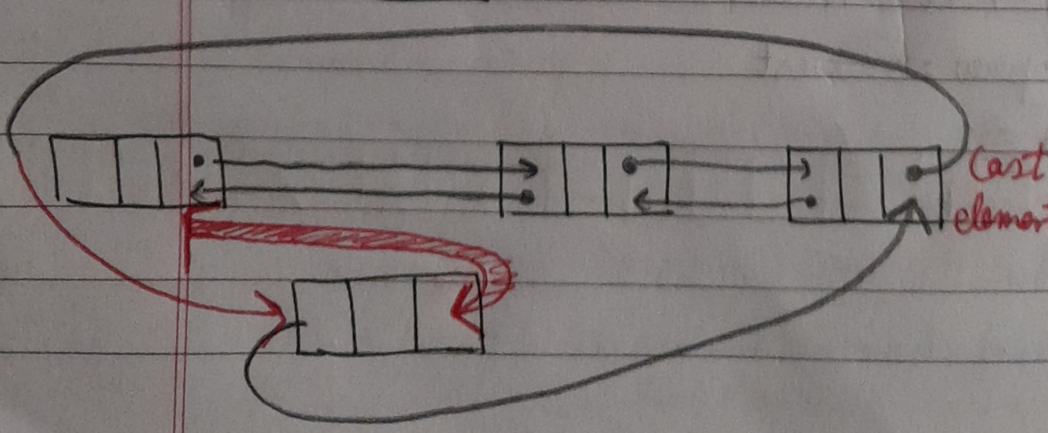




Insertion at beginning –

Create a new node, newest.
We have explicit refrence to last element.
~~Point last next to newest.~~
Point newest to last · next.
THEN poin last · next to newest.

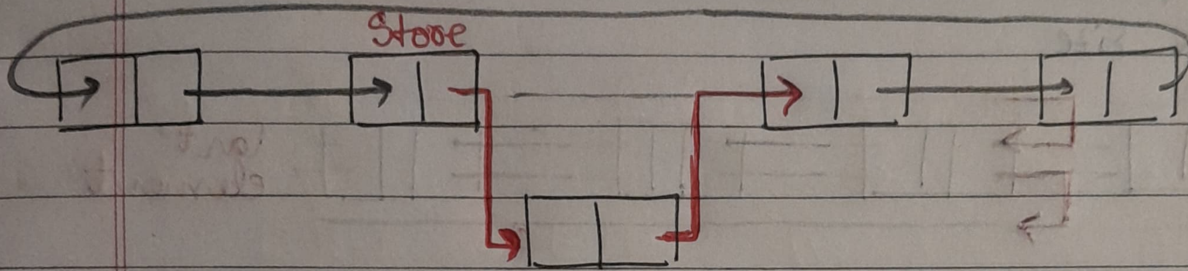


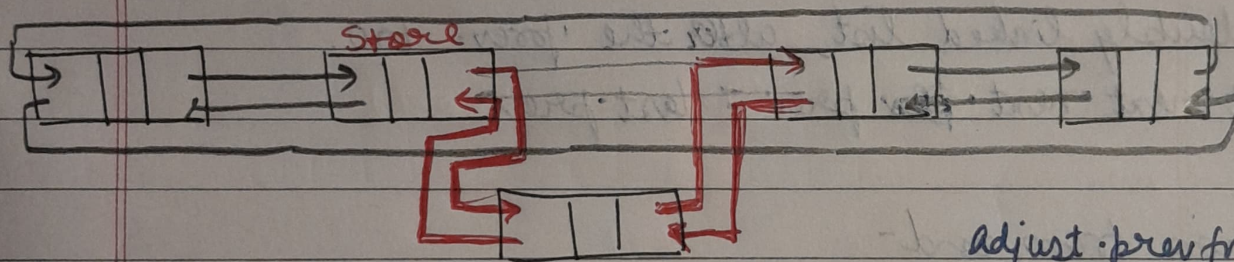


Here, point prev
as needed.

Insertion in b/w nodes —

Travel to the node where you want to enter, Store it
create a new node.
newnode.next = storednum.next
stored num.next = newnode

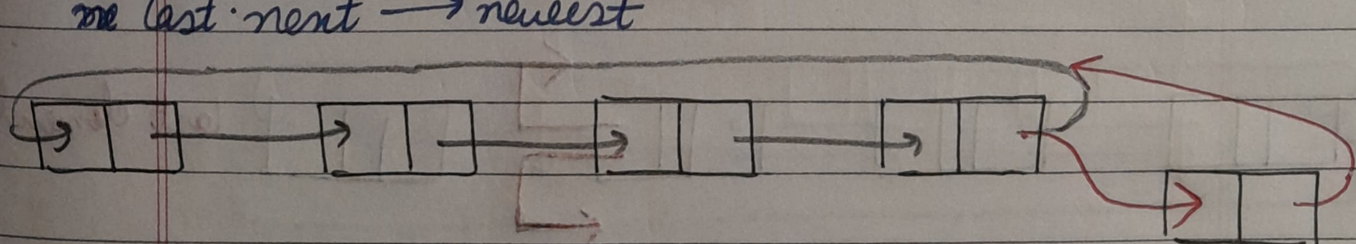


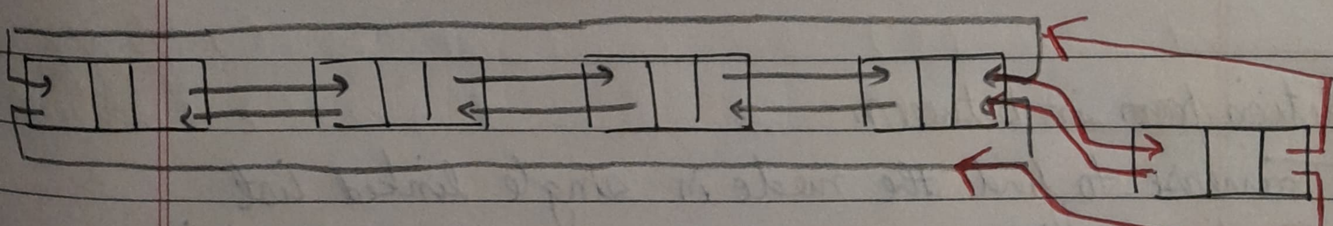


adjust.prev for DLL

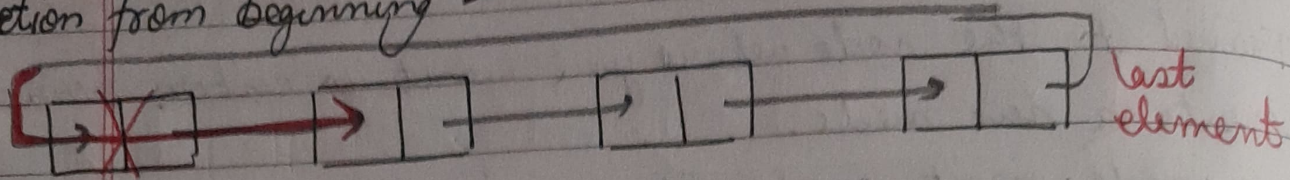
Insertion at the end (after last) -

Create new node.
newnode.next → last.next
~~ne~~ last.next → newest

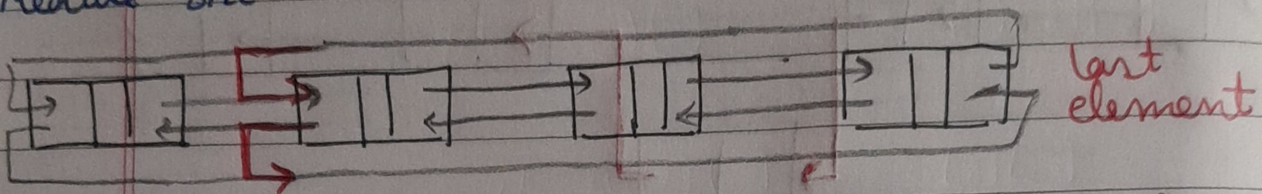




Deletion from beginning -

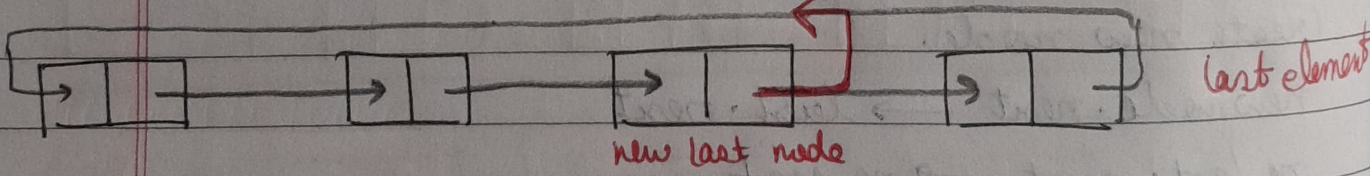


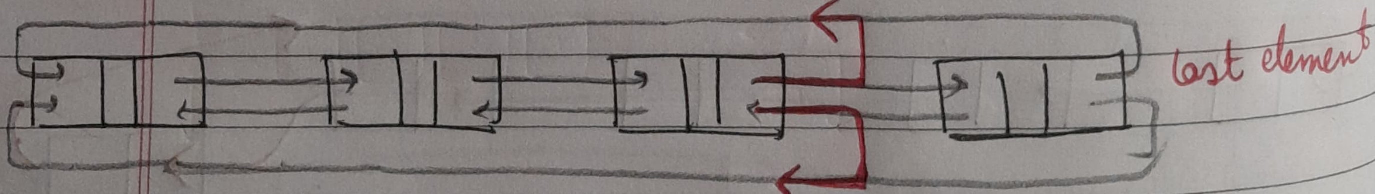
Simple point last.next = last.next.next.
Reduce size



In doubly linked list, alter the prev. too.
last.next.next.prev = last.prev

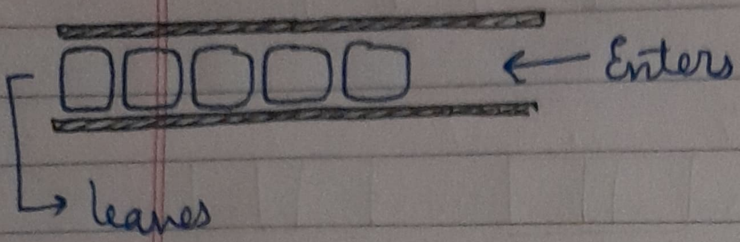
Deletion from end -
Reassign the last node (traverse)
Then assume the node to be first node



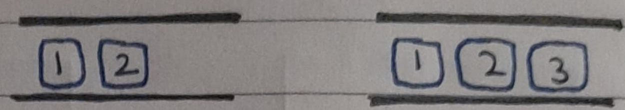


Deletion from in between -
Traverse to find the node in single linked list
In double linked list, we have explicit declaration

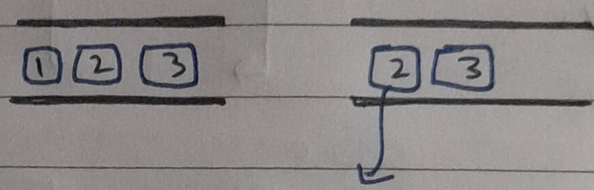
QUEUE FIFO: first in first out.



Enque (insert element) -



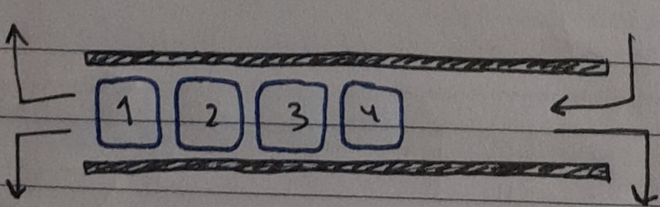
Deque (deletion of element) -



IS_full()   IS_empty()   Peek (front)

Complexity   enqueue is O(1)
deque is O(n) ☆ we solve this using circular and double ended queues

DOUBLE ENDED QUEUE



☆ insert_at_end() and insert_at_start()
☆ delete_at_end() and delete_at_start()