

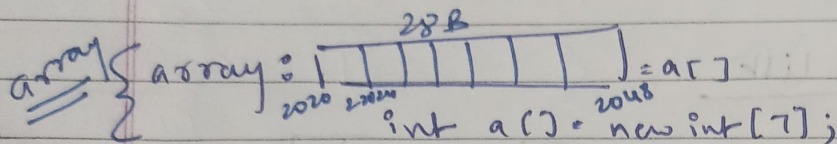
→ Linked list Basics:

linear data structure series of connected node

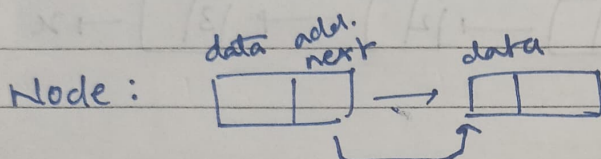
→ linear form data → no size

Initial state

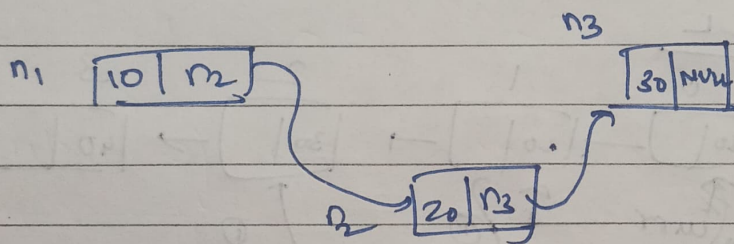
int → 4B



memory = 4 × 7
= 28B



code:



n1.next = n2;

n2.next = n3;

Stack: n1, n2, head

Linked list

Array

→ not continuous

→ continuous

→ non constant size

→ constant size

→ more memory

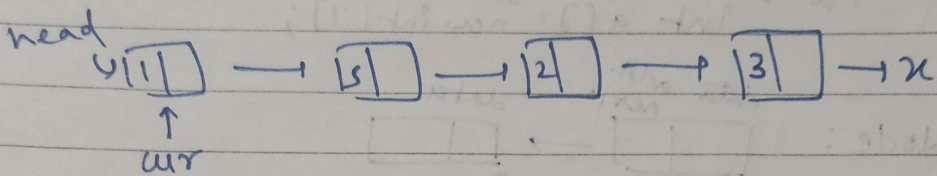
→ less memory

→ accessing not easy
traverse the whole linked list

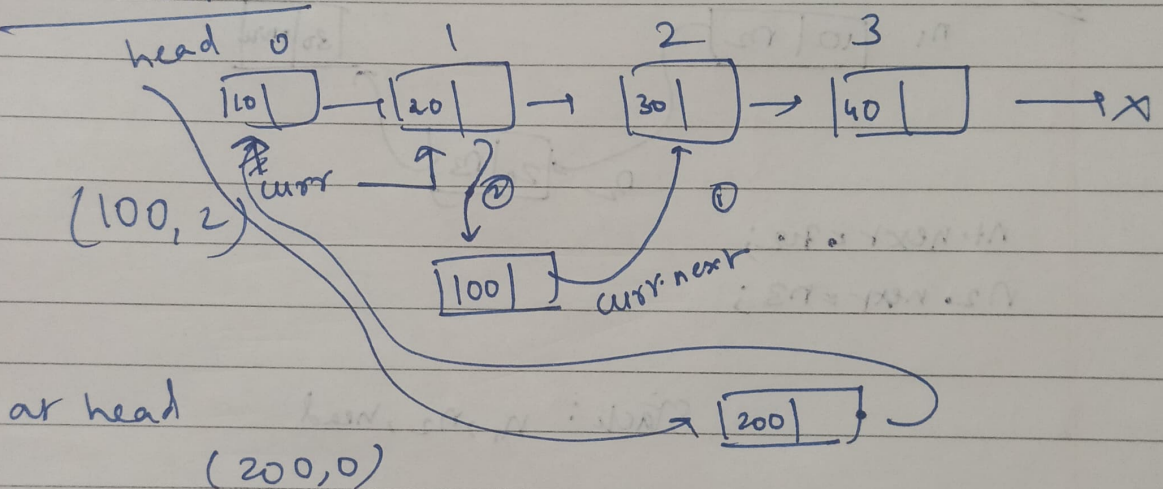
→ accessing very easy
Index can help in direct accessing

- memory allocated dynamically
- insertion is easy and deletion $O(n)$ efficient
no swappers
- non dynamically
- not easy $O(n)$

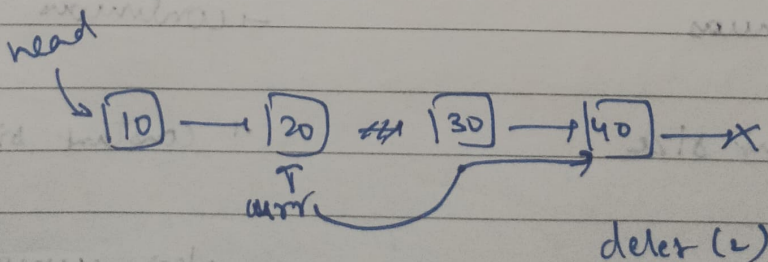
Traverse in Linked list



Insert in LL



Delete in a linked list



Find the middle element in LL

2 traversal to see no.

and to return

1 traversal

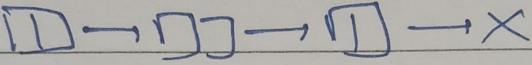
one pointer slow

$slow = slow \rightarrow next$

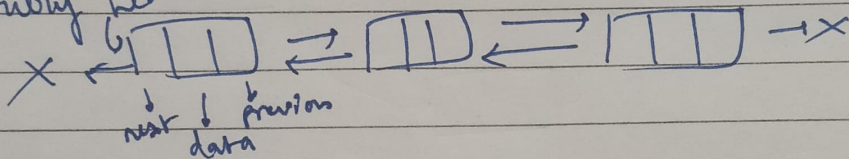
one pointer fast

$fast = fast \rightarrow next \rightarrow next$

Types of Linked list

single \rightarrow 

doubly head



circular

