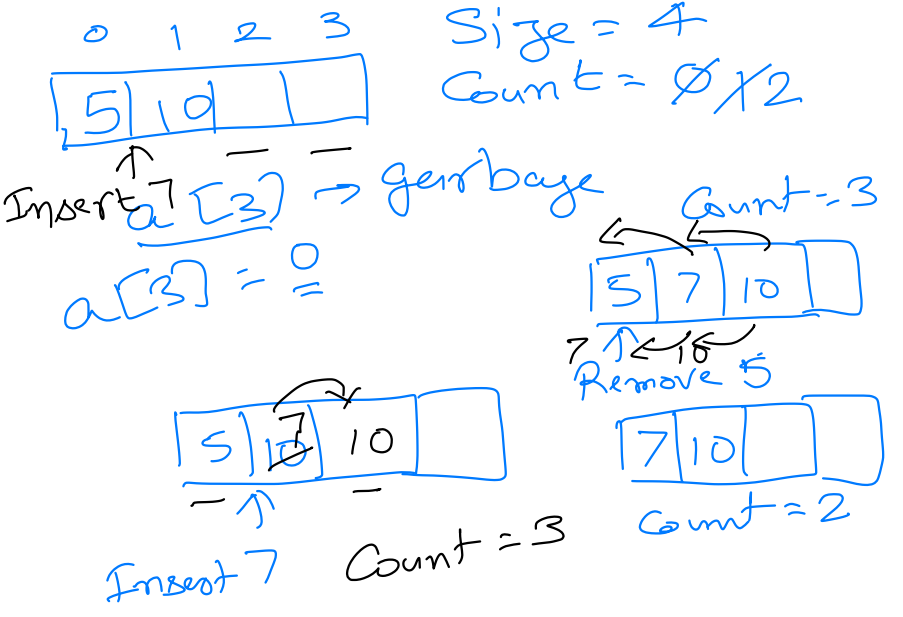
Array

- Need for an array?

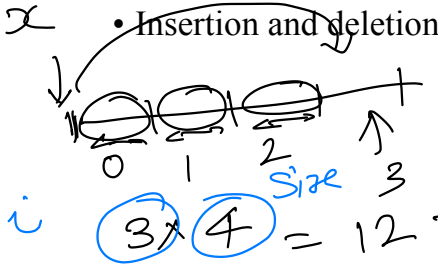


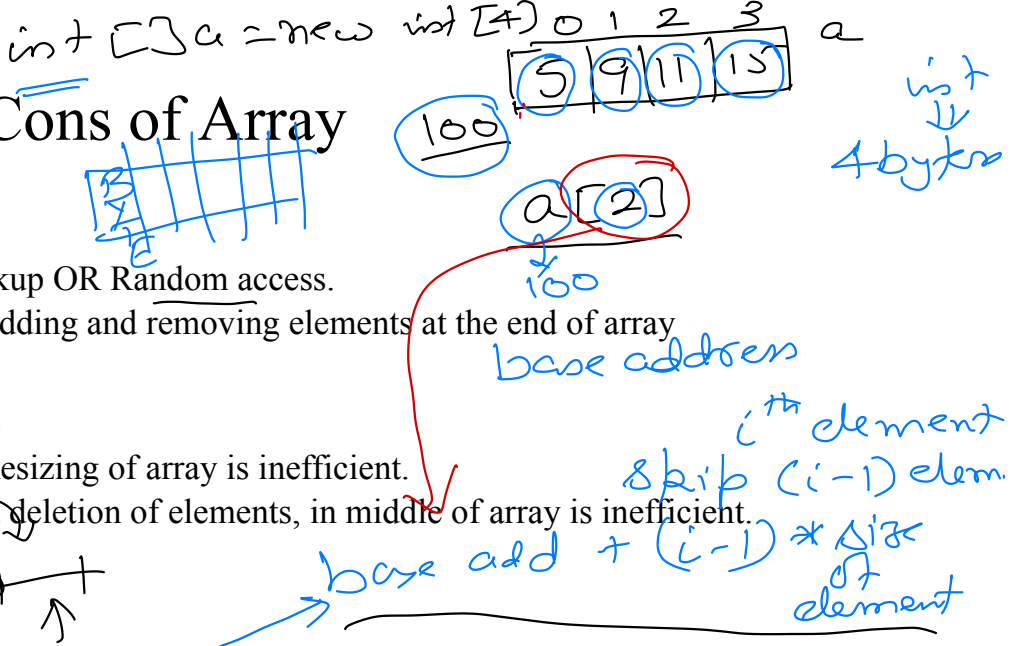
Properties of Array

- Data Structure that stores multiple elements, all of the same type.
- All elements of an array are stored sequentially in memory, one after another.

Pros and Cons of Array

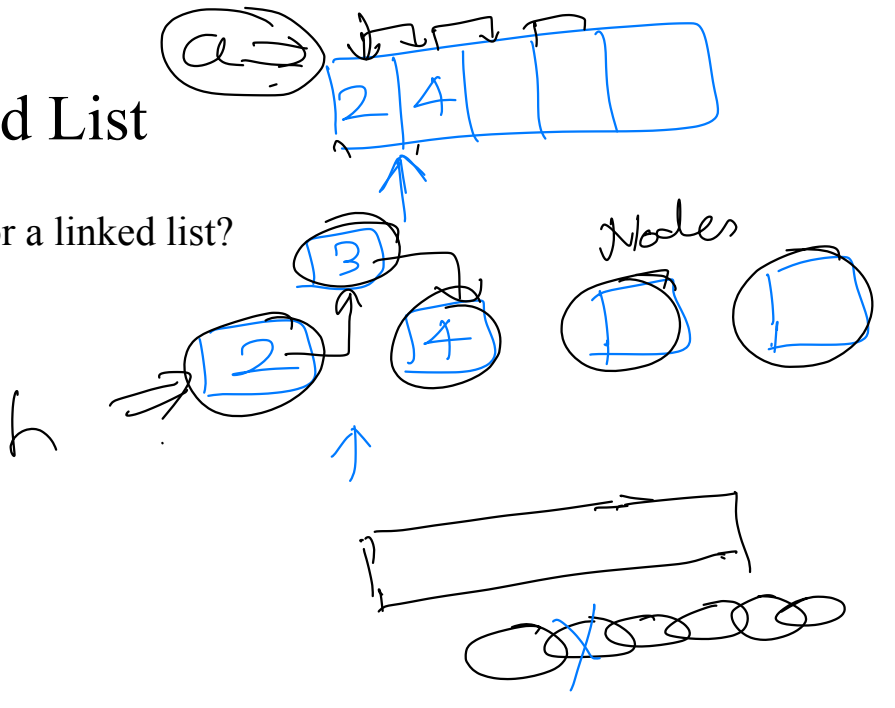
- Advantages
  - Efficient lookup OR Random access.
  - Efficient in adding and removing elements at the end of array
- Disadvantages
  - Fixed size. Resizing of array is inefficient.
  - Insertion and deletion of elements, in middle of array is inefficient.





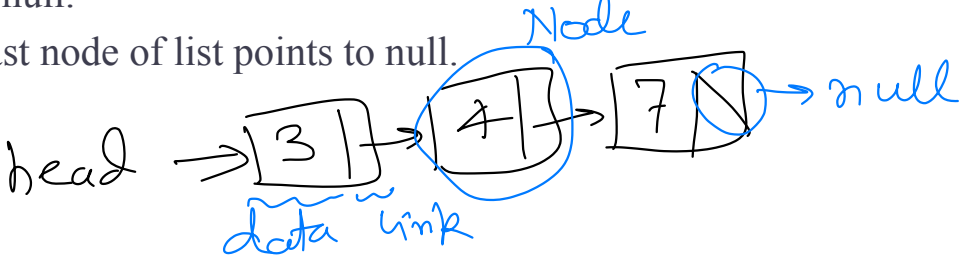
Linked List

- Need for a linked list?

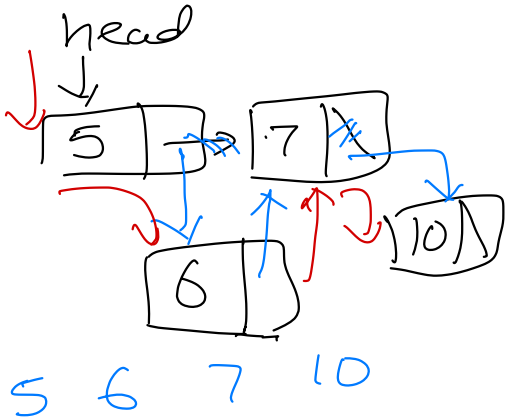


Properties of Linked List

- Stores data as a **chain** of **nodes**.
- Each node contains **data** and **a pointer to the next node** in the chain.
- First node of linked list is pointed by "**head**". When list is empty, head is null.
- Last node of list points to null.

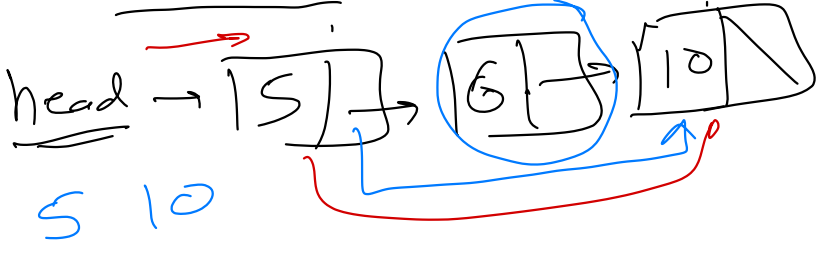


Pros and Cons of Linked List



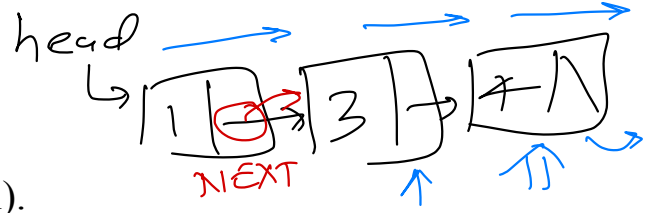
- Advantages
  - Can dynamically grow or shrink is size.
  - Efficient in insertion and deletion of elements.

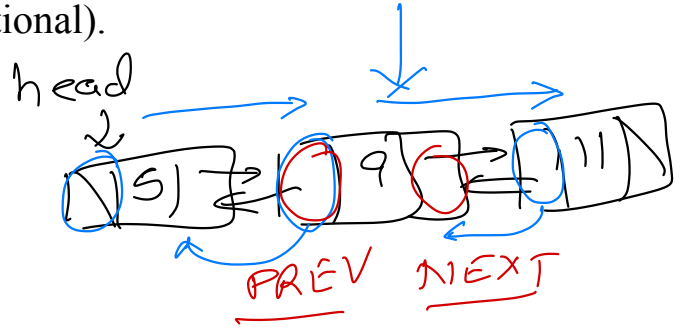
- Disadvantages
  - Lookup OR Random access is inefficient.



Types of Linked List

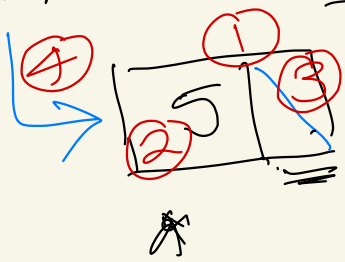
- Single linked list (Uni-directional).
- Doubly linked list (Bi-directional).
- Circular linked list.





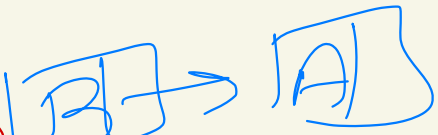
Initially, list is empty
head → null

Add a node
→ at beginning
→ at end
→ insert



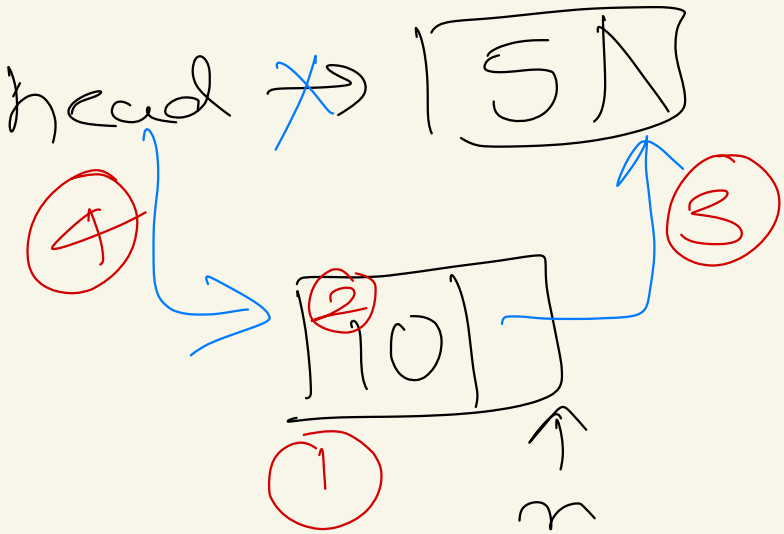
n

A comes after B



Add a new node at start of list

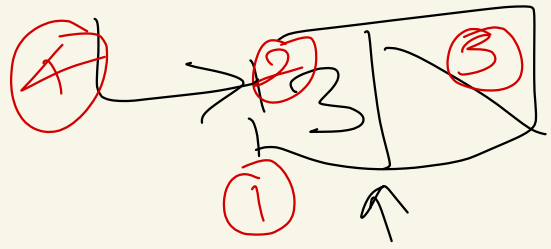
1. Create a new node. (n
2. Store data in new node.
3. Make head node come after new node.
4. Make new node as head.



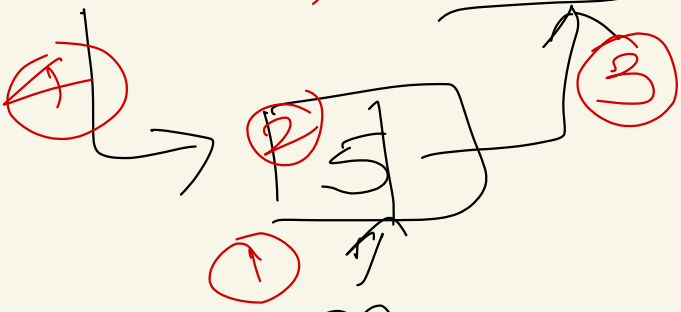
3 5 10 ← add

head → null Empty list

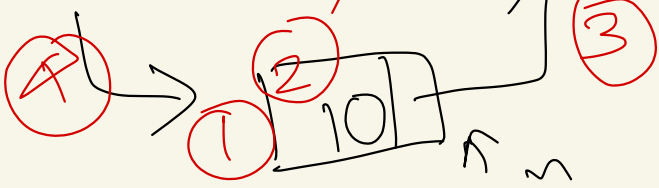
head → ~~null~~ Add 3

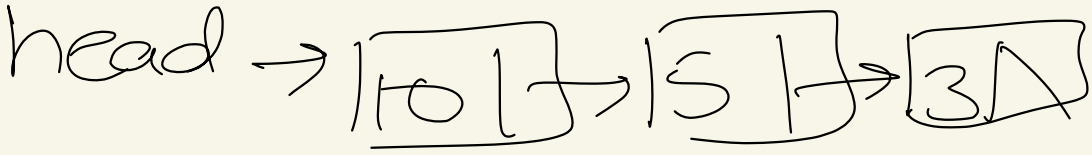


head → 3 Add 5



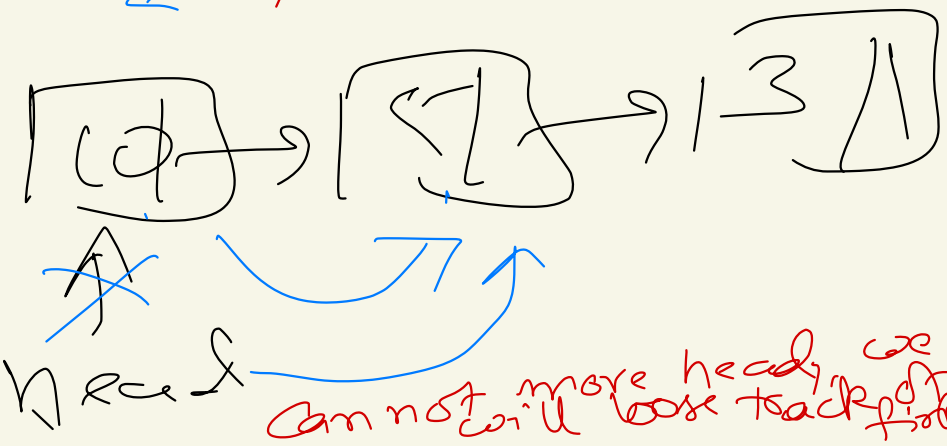
head → 5 → 3 Add 10

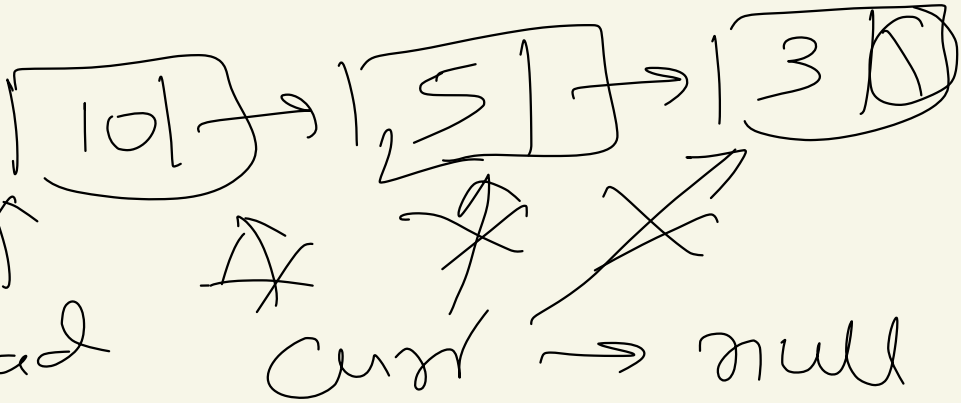




Traverse the list

→ Sequential

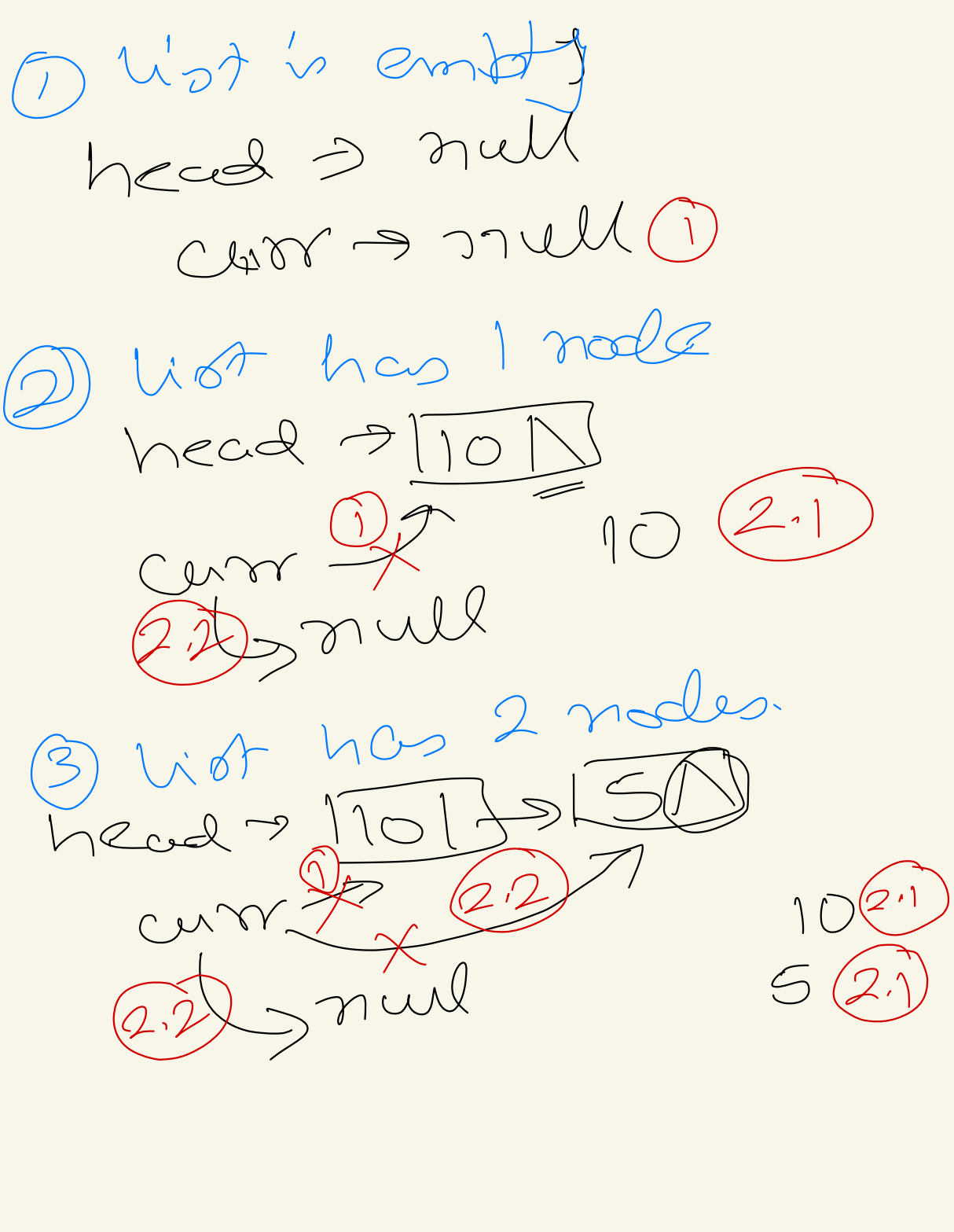




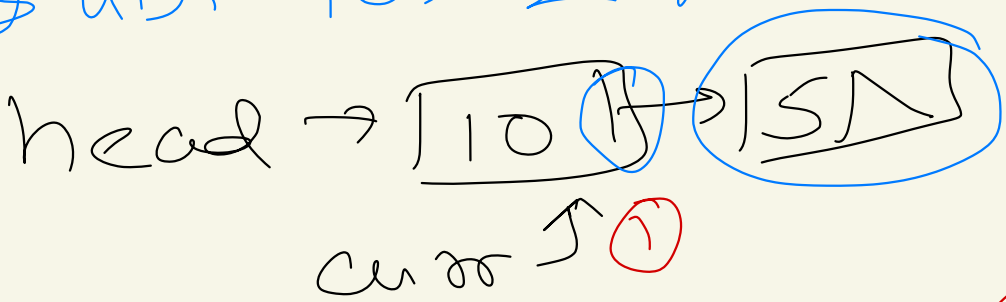
Traverse a list

1. Set curr to head.
2. while (curr != null)
   - 2.1 Process curr's data
   - 2.2 Make curr point to curr's next node.

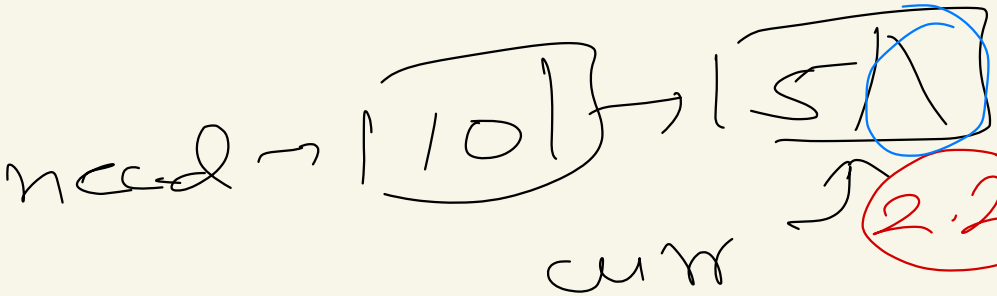
1. Empty list
2. List with one node.
3. List with 2 nodes.


1 List is empty
head → null
curr → null 1
2 List has 1 node
head → 10
curr 1
2.2 → null
10 2.1
3 List has 2 nodes.
head → 10 → 5
curr 1 2.2
2.2 → null
10 2.1
5 2.1

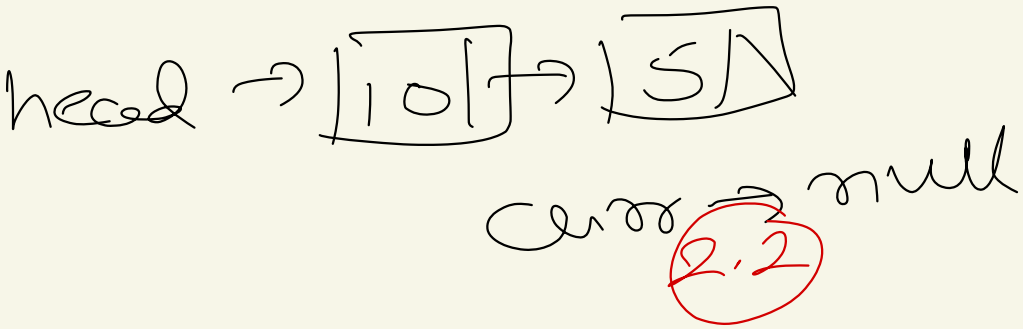
(3) list has 2 node.

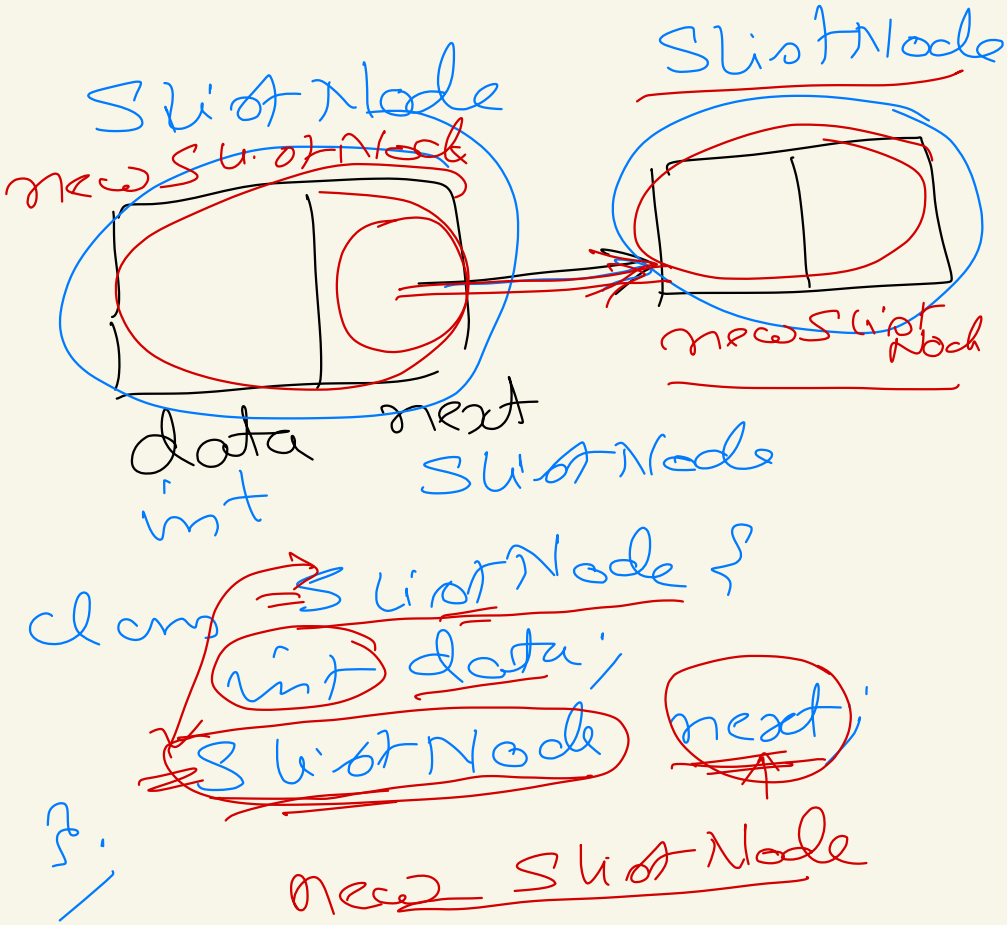


10 2.1



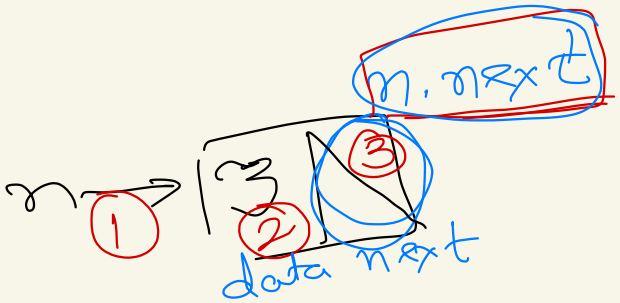
5 2.1




SListNode
SListNode
new SListNode
new SListNode
data
next
int
SListNode
class SListNode {
int data;
SListNode next;
};
new SListNode

1. SListNode n = new SListNode();
2. n.data = no;
3. n.next = head;

head → null



Store 5 in a.

a = 5;