4. Linked List :

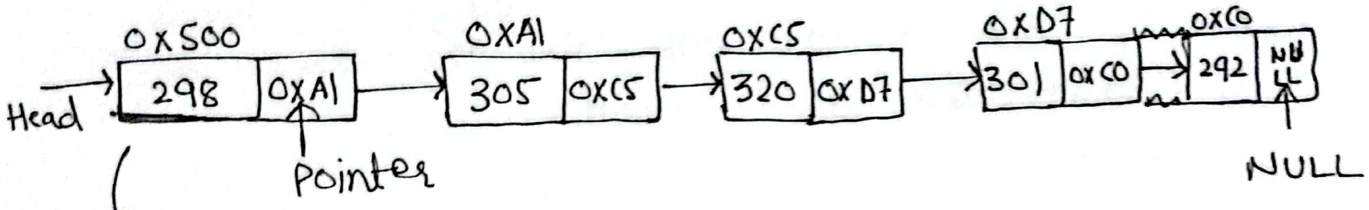
) Issues with Arrays,

① when you insert a element in array (in middle) or at index 1. Then all the elements have to copied & pasted one space ahead to make space. If we have large no of elements, it is not efficient way.

② Also when dynamic array's space is totally finished, for the new element all the elements are copied to new place having more memory.

↳ which is also not efficient.

In Linked Lists :

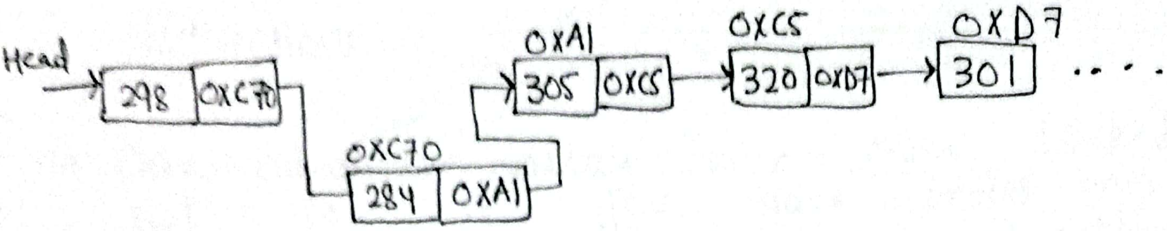


Here we have all the element stored at random addresses (unlike arrays where we have contigious memory locations). And those random addresses are linked together using a pointer.

↳ like 1st pointer has value (OXA1) that is address of second number. then so on and so for.

(Head points to next element)

→ One of ease linked lists gives us is insertion at random space. (start)
e.g If you want to insert 284 at index 1.



↳ so simply what we did is we change 1st pointer value to 'OX C70' that has 284

⇒ Insert element at beginning = O(1)
⇒ Delete element at beginning = O(1)
⇒ Insert | Delete element at the end = O(n)
↳ b/c you have to change all addresses.

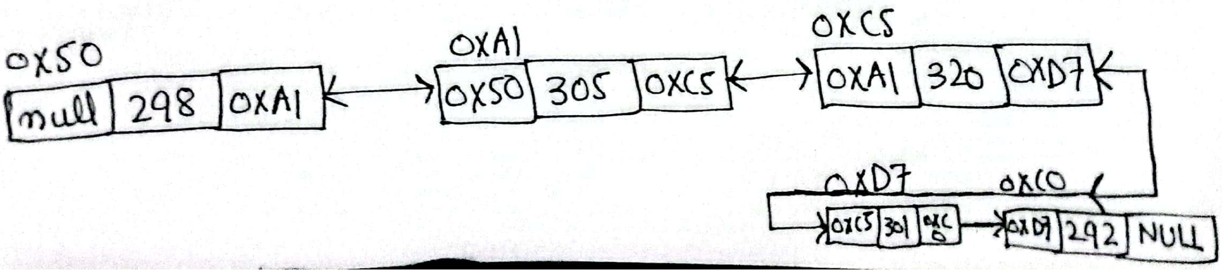
Benefits over Arrays:

① You will have just allocated that space which you need.

② Insertion is easier.

Linked list Traversal = O(n)
Accessing element by value = O(n)

Double Linked Lists:



↳ So in double linked lists we have address of the next element and also the address of previous element. So we can traverse in both directions.

→ One advantage array have over linked list is if you have index no you can access element directly but in linked list you have traverse through list.

↳ array → O(1)

linkedlists → O(n)

→ Inserting/Deleting in end of arrays → O(1)

↳ but if reaching limit you have to copy.

↳ Inserting/Deleting in end of linked list ⇒ O(n)