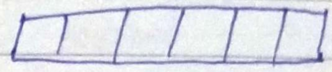
Data Structure and Algorithms.

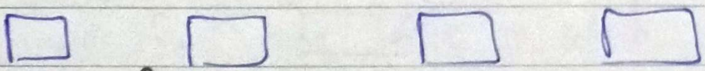
linked List:

*. linear data structure used to store a list of values.

single memory block with Partitions

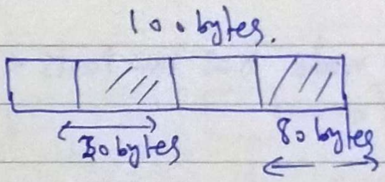


* memory blocks linked to each other



Challanges of an array:-

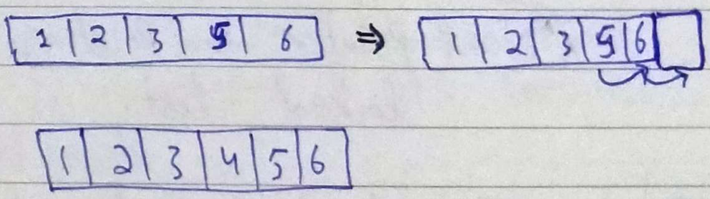
1. Static size e.g. we can't expand the array.
2. Contigous memory allocation



∴ we can't allocate 100 bytes because not single block present.

3. Inserting and deleting is costly (O(n))

e.g. Inserting 4

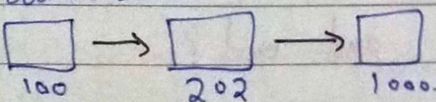


Advantages of linked list over an array:-

1. dynamic size allocation. (we can't specify at Start size)

2. Non-contigous memory allocation.

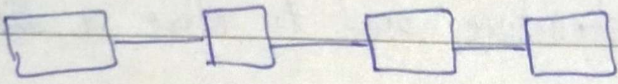
e.g



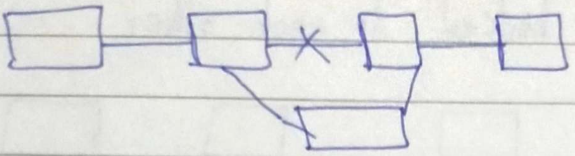
means that blocks are not to be placed at contigous fashion. they can be but not compulsory.

3. insertion and deletion is not expensive

e.g.



Suppose we have to add ☐ at 3rd Position so,
we Just remove link and place it

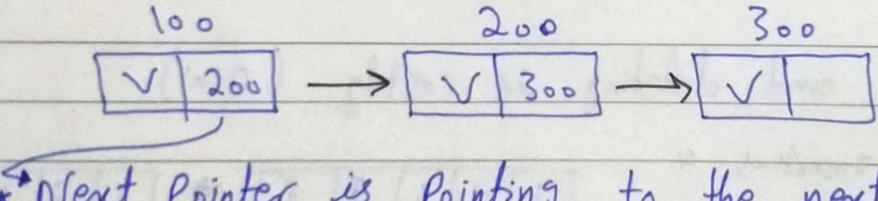


List node:
↳ two Parts
block of memory → node

list node has two Parts < Data / value storing
Next Pointer (this N.P will be pointing to the ~~list~~ next node)

Data	N.P

Example:



(100, 200, 300 are adresses)

* Next Pointer is Pointing to the next node in linked list.

* The adress of 2nd node will ~~d~~ be store in the 2nd Part of Ist node and so on.

How do we know my linked list from where it starts and end?

* For that we place a <u>head</u> Pointer

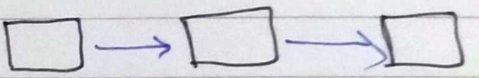
* this head Pointer Points to the first node of linked list

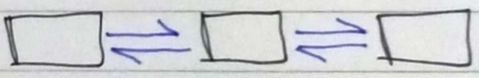
Q. If we have to Pass linked list to any function?

 * we Just Pass head note and that will be give me entire linked list.

Q to identify the ~~the~~ End Node?
 * As we find any node in which null Pointer is stored at 2nd Part ⇒ So, we identify the end node.
 Also called tail node.

Types of linked list:-

1) Singly linked list:

 ↳ every node points to it's successor node (next node)
 ↳ we can only move in forward direction.

2) Doubly linked list:

 ↳ every node is connected to it's Previous and next node.
 ↳ we can move in both forward and backward direction.

3) Circular linked list:
 ↳ the last node is Pointing to the head node