Don't wait for the right time.

Create it.

- → Why linked-linked ?
- → Practical use-case.
- → Insertion in linked list
  - → At start
  - → At end
  - → At Kth- index
- → Deletion in linked list (#idea)

Arrays.

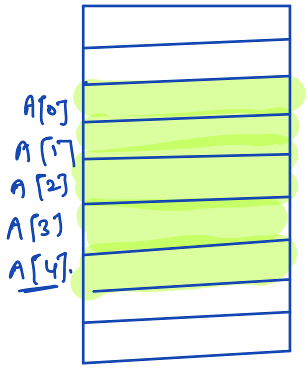
arr[N]

arr[5].

Time complexity to access any random element → arr[i] → $\boxed{O(1)}$

↓ ?

Elements stored are contiguous.



int[] arr = new int[3] X [Not possible]

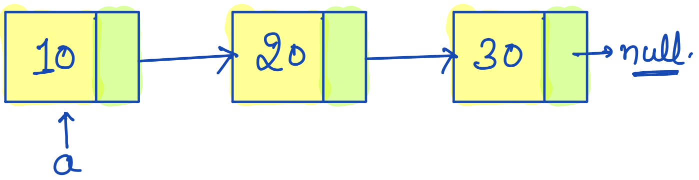
⇓

Contiguous space for 3 integers is not available.

	At start.	At end	k^{th}-index.
Insertion	O(N)	O(1)	O(N)
Deletion	O(N)	O(1)	O(N)

Linked-List.

```
class Node {
      int val;
      Node next;

      Node (int x) {
        this.val = x;
         this.next = null
      }
}
```



```
Node a = new Node (10);
 a.next = new Node (20);
a.next.next = new Node (30);
```

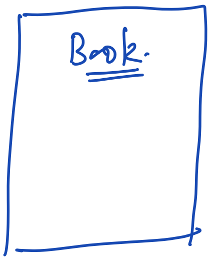
```
Node a = new Node (10);
[Node b = new Node (20);
     a.next = b.          ]
```

Music Player.

★ Favorite Songs-

→ Song-1
→ Song-2
|
|
|
→ Song-50

★ Search engines.

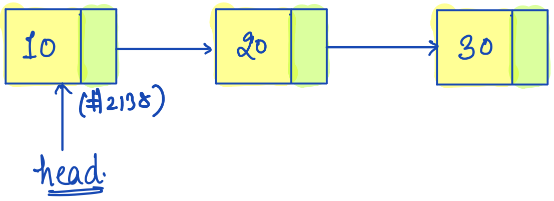


[Galaxy]

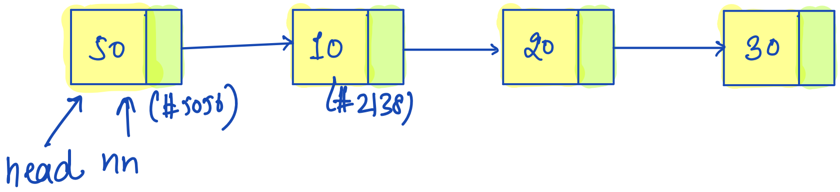
→ Glossary.
→ Appendix

galaxy → [-,-,-,-,--]

Insertion At Start



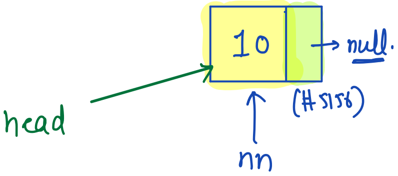
→ insert 50 at start

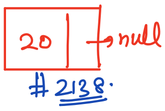


```
Node insertAtStart( head, val){
        Node  nn  =  new Node (val);
          nn.next = head ;
           head = nn ;
         return head.
}
```

[head→ object reference]
↳ pointer.

T.C → O(1)
S.C → O(1)

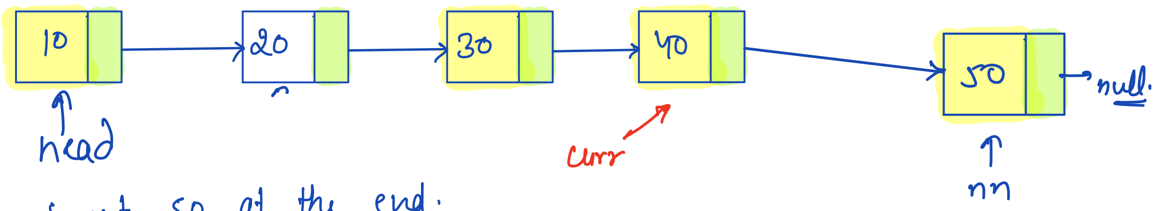




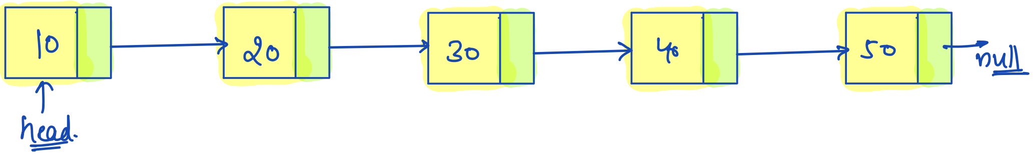
```
            (#2138)
Node  a  = new Node (20);
Node  b  =  a
        (#2138)   (#2138)
```

{ null ← a   a.val
               a.next] ⇒ NULL POINTER EXCEPTION }

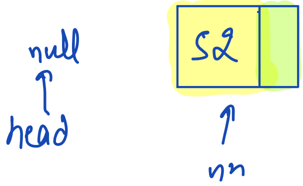
Insertion At End.



→ insert 50 at the end.



```
Node insertAtEnd ( head, val) {
    Node nn = new Node (val);
    if (head == null) {
        head = nn ;
    }
    else {
        Node curr = head ;
        while ( curr.next != null)
        {
            curr = curr.next ;
        }
        curr.next = nn ;
    }
    return head;
}
```

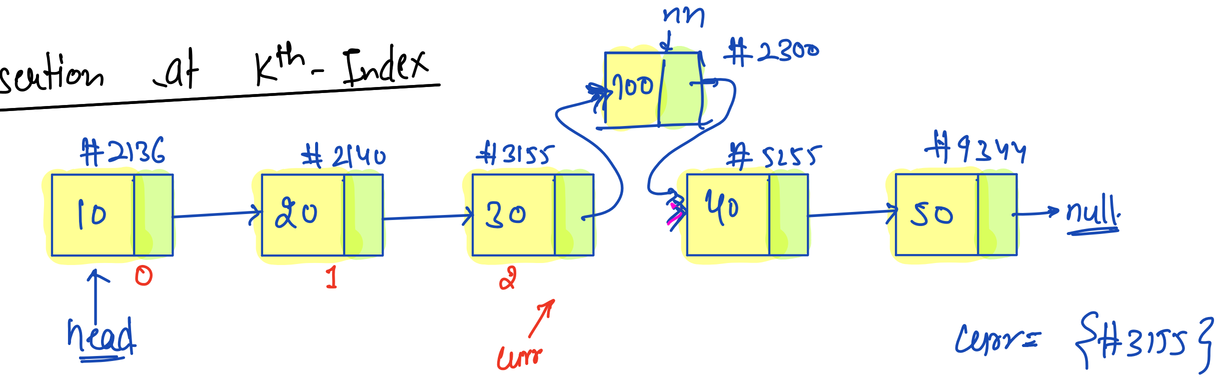


T.C → O(N) → O(1)
S.C → O(1)

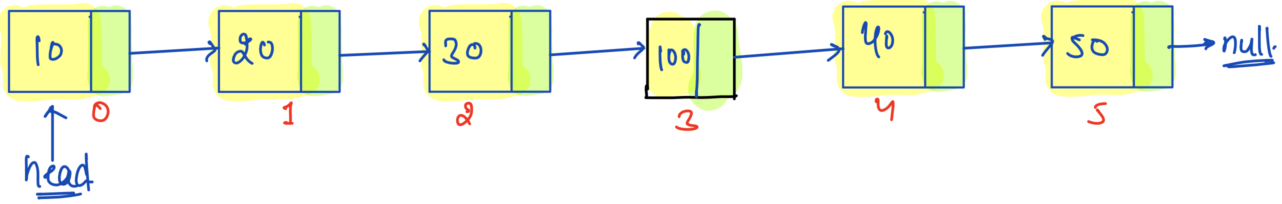
*.

//Create a new Node (nn)
tail.next = nn
tail = nn

Insertion at Kth- Index

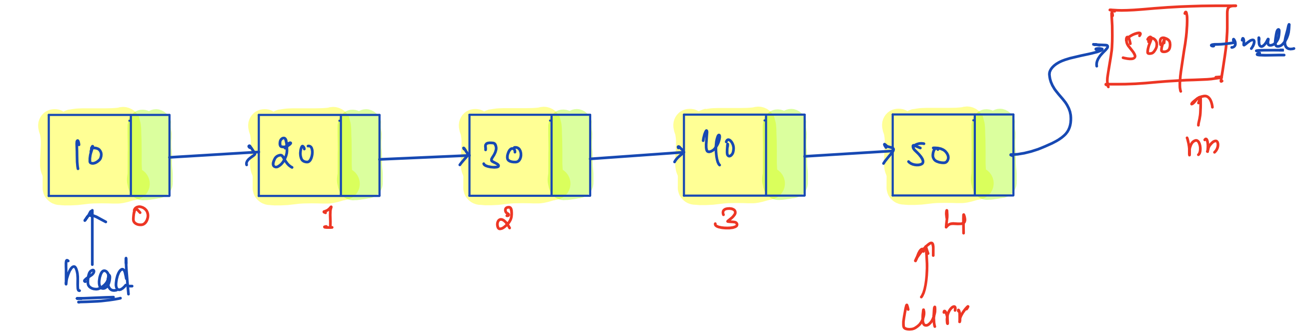


→ insert 100 at 3rd index



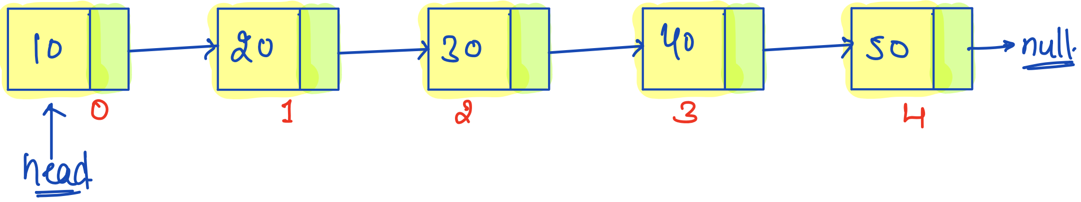
```
Node insertAtKthIdx (head, val, K){
        Node nn = new Node(val);
    if (K==0){
        return insertAtStart(head, val);
    }
    else{ Node curr = head;
        for( i=1 ; i<K ; i++){
                curr = curr.next;
        }
                        #5155
          nn.next = curr.next   // creating right connⁿ
                           #2300
          curr.next = nn        // creating left connⁿ.

        return head;
    }
}
```

T.C → O(N)
S.C → O(1)



insert 500 at 5th index.

✓ Correct ans if (k== n).



① Delete At First

head = head.next;

② Delete At Last

→ Traverse upto second last node.

→ curr.next = null

③ Delete At kth idx

[# todo]

[Mistakes → learn.]

1. Write the code on paper first.

2. Dry-run.

3. Never use "try & error"

4. Think about edge cases & Null Pointer Exception.