Advantage of Array

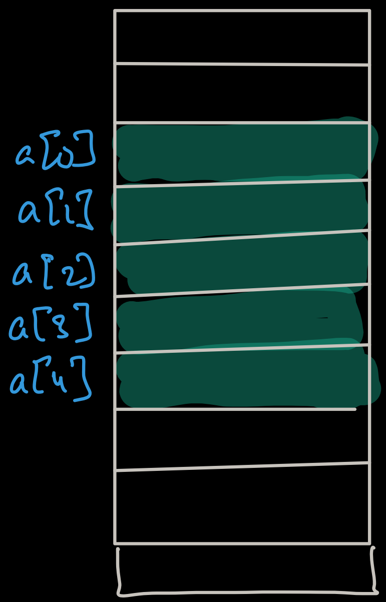
↳ array / arraylist

$A[i] \rightarrow O(1)$

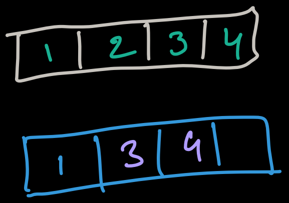
```
int [ ]  a = new int [5];
int [ ]  b = new int [5] x
```

Don't have enough memory

O(1) random access



	At start	At end	At random (kth pos)
Insertion	O(n)	O(1)	O(n)
Deletion	O(n)	O(1)	O(n)
Update	O(1)	O(1)	O(1)



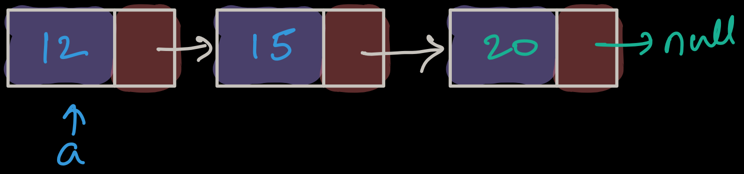
word → page No.

inverted index

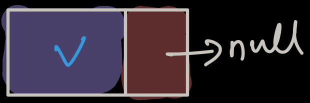
milk → 47
phone – [- -]
iphone – [- - - -]
iph – [. - - - .]

Linked List

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



```
Node a = new Node (12)
a.next = new Node (15)
a.next.next = new Node (20)
```



* this : reference variable that refers to current object.

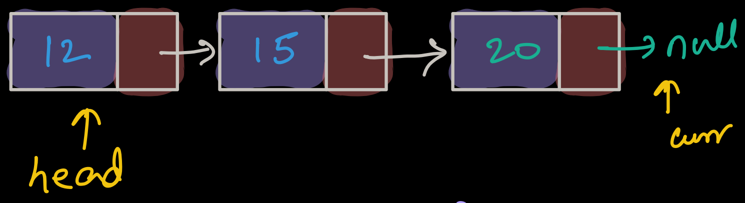
```
class pair {
    int x,y;
    pair(int x, int y) {
        x = x
        y = y
    }
}
x, y = 0,0
```

```
class pair {
    int x, y
    pair (int x, int y) {
        this.x = x
        this.y = y
    }
}
```

```
pair ( int a, int b) {
    this.x = a
    this.y = b
}
```

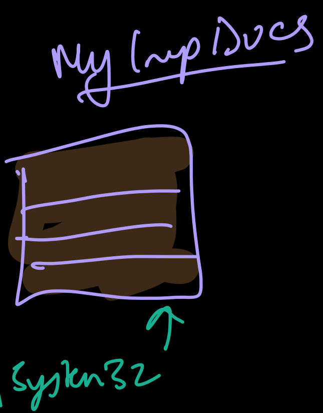
Print the Linked List

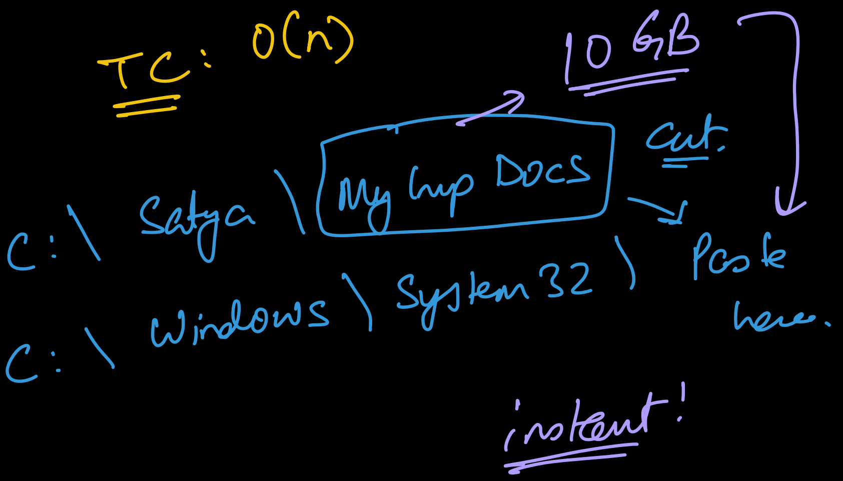
```
void printLL(Node head) {
    Node curr = head
    while(curr != null) {
        print(curr.val)
        curr = curr.next
    }
}
```



12 15 20

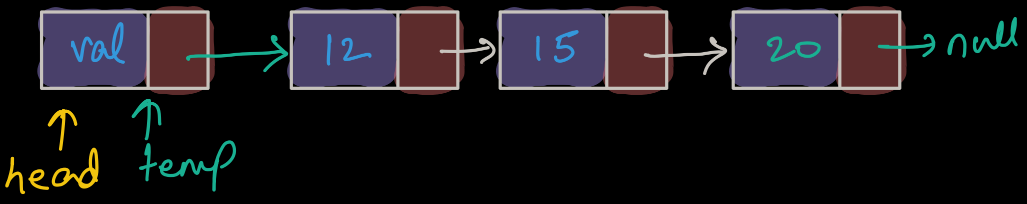
TC: O(n)



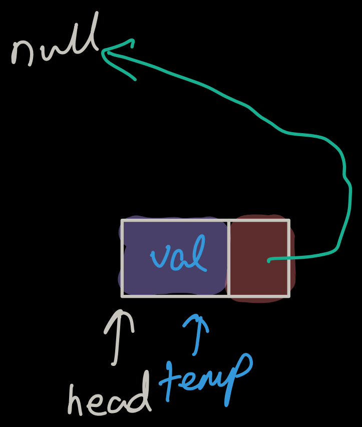


Insertion at start in LL

```
Node insertAtStart(head, val) {

    Node temp = new Node(val)

        temp.next = head

    head = temp

    return head

}
```

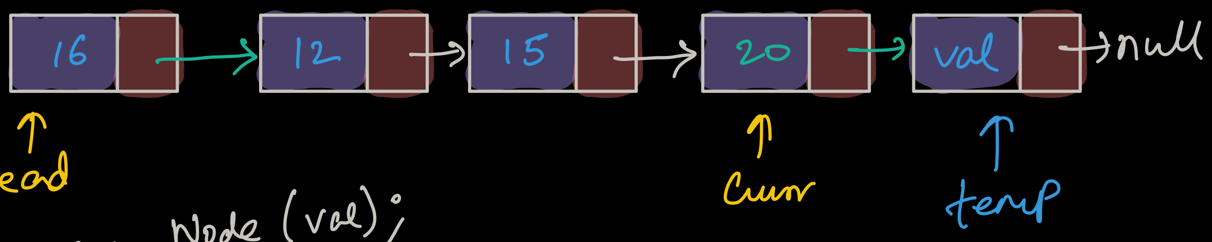


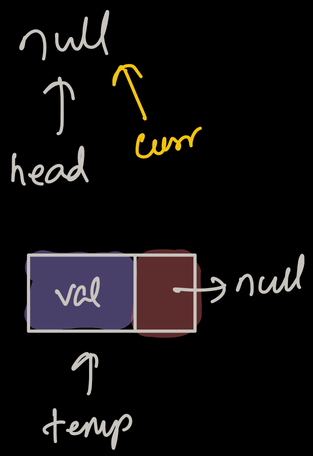
TC: O(1)



Insert at the end

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

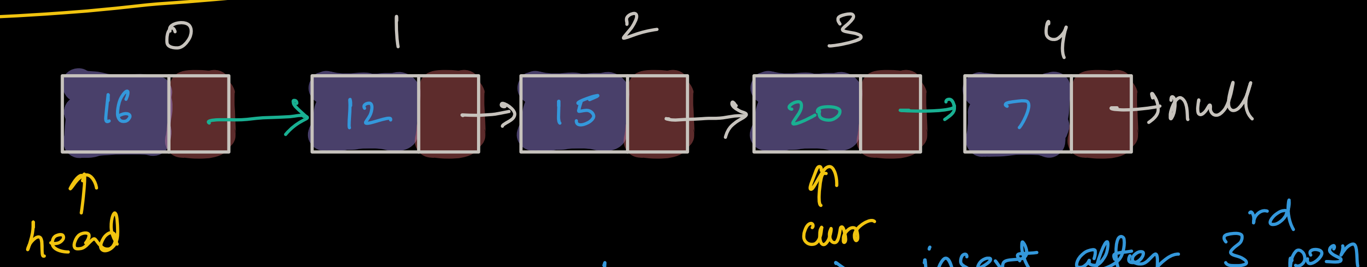




TC: O(n)

Break: till 8:43 am.

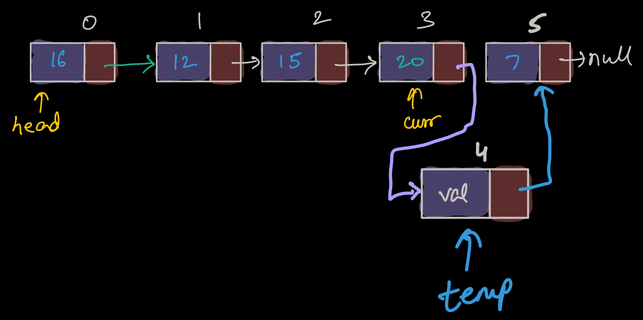
Insert at k th pos

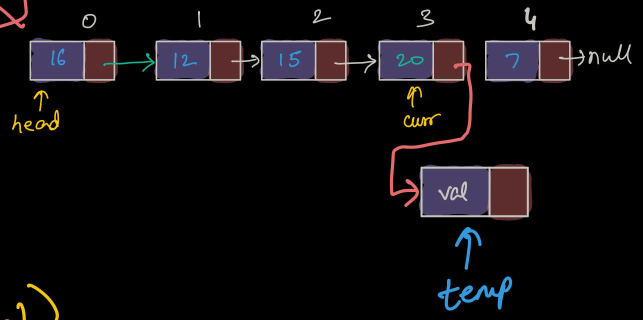


insert @ 4th pos ⟹ insert after 3rd posn

K=4

```
Node insertAtKPos(head, val, K) {        // K=N , code works.
    if(K==0) {
        head = InsertAtStart(head,val)
        return head
    }
    else {
        count=0
        Node temp= new Node(val);
        Node curr = head

        while(count < K-1) {
            curr= curr.next
            count++
        }
        ~~curr.next= temp~~ X
        temp.next = curr.next
        curr.next= temp
    }
}
```

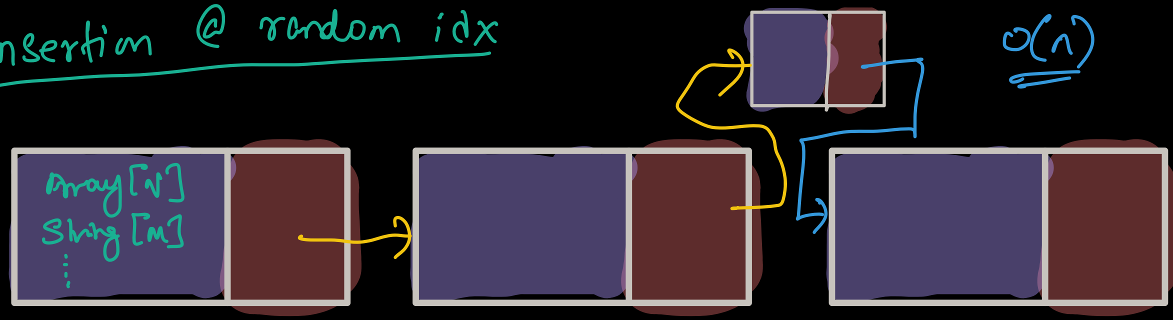




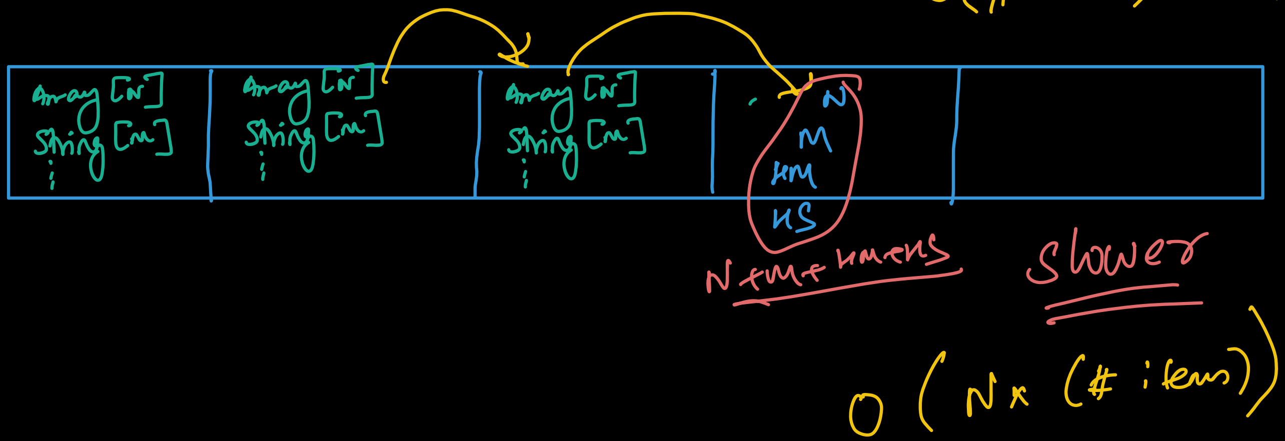
TC: O(N)

Delete: Todo

Insertion @ random idx



O(# items) is for 1 shift



→ Write code in paper

→ Dry run your code

→ Never do trial & error. Try to figure out the mistake.

→ Check your edge cases

* null
* size 1
* size 2/3
* Problem specific cases.