Linked List

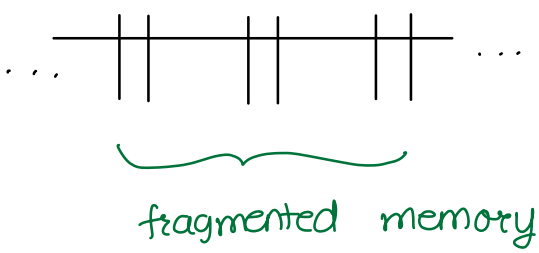
- Abhishek Sharma
- Akansh Nirmal
- Bhaveshkumar
- Burhan
- Divya P
- Gagan Kumar S
- Gowtham
- Nikhil Pandey
- Pankaj
- Prajwal Khobragade
- Purusharth A
- Rajat Sharma
- Rajendra
- Sanket Giri
- Saurabh Ruikar
- sharath r
- Shradha Srivastava
- Shrikanth
- Subhashini
- Subhranil Kundu
- Sumit Adwani
- Sushant Patil
- suyash gupta
- Vasanth
- Venkata Sribhavana Nandiraju
- Vimal Kumar
- Vishal Mosa

- → Use of linked list over arrays.
- → Basic functions { Access, Search }
- → Insertion and deletion
- → Reverse the linked list
- → Palindrome list.

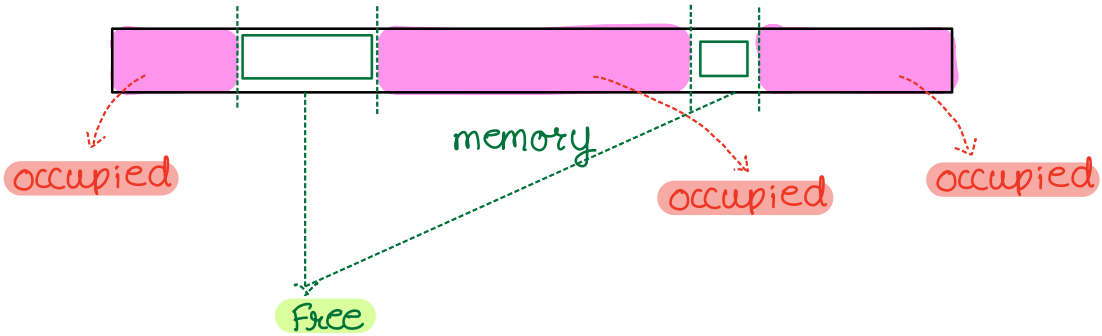
current PSP

63% ⟶ 65%

No continuous blocks of 100 mb is available



fragmented memory



If you create an array will you be able to utilize all of the above free space. ⟶ NO.

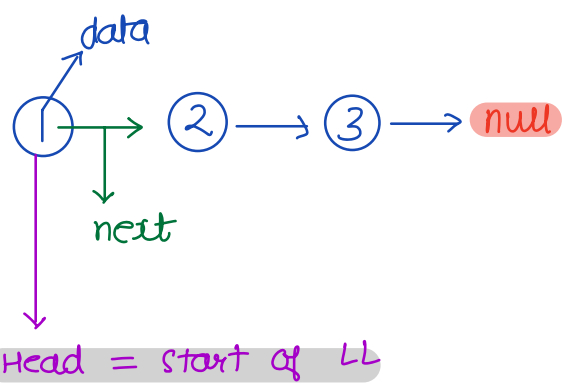
↑ Limitation of Array.

What is linked list ?

Linear DS which can utilize all of the available memory efficiently.

Representation


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



Operations

1> Access k^{th} element { k = 0 is the first element }

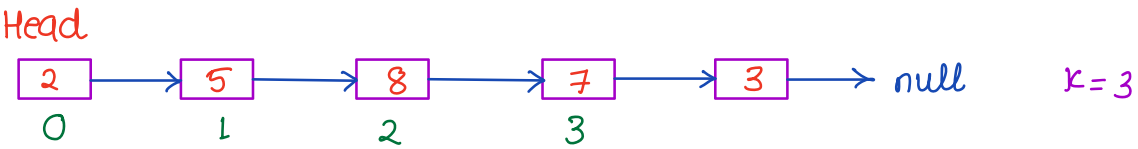
	0	1	2	3	4
A =	2	5	8	7	3

k = 3

A[k]

Tc to access k^{th} node

Tc : O(1)



k = 3

NOTE Never change Head unless required

```
Node temp = head
for i ⟶ 1 to k { // jump k times
    temp = temp.next
}
print ( temp.data) .
```

Tc : O(k)

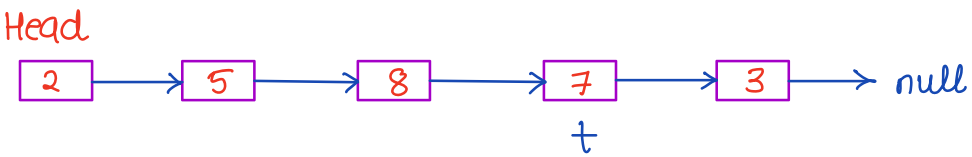
Sc : O(1)

2> check for value X { searching }

Array → Not sorted → Linear search TC: O(N)
→ Sorted → Binary search TC: O(log N)

Linked List → Linear search only

X = 7



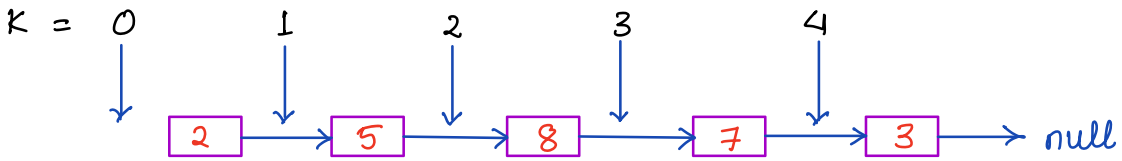

```
Node temp = head

while ( temp != null ) {
    if (temp.data == X)
        return true
    temp = temp.next
}
return false
```

TC: O(N)

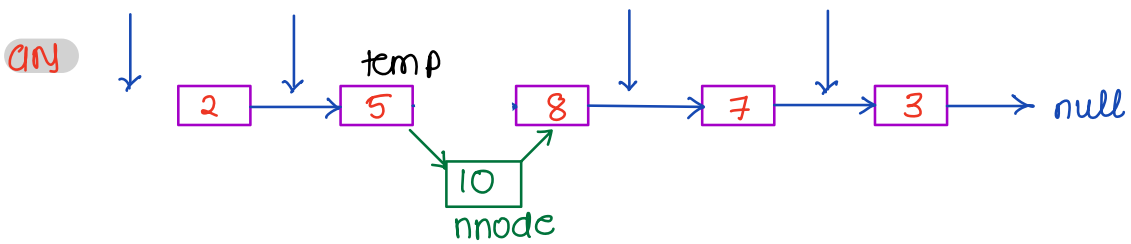
SC: O(1)

3> Insert a value X at k^{th} position {0-based} in a linked list.

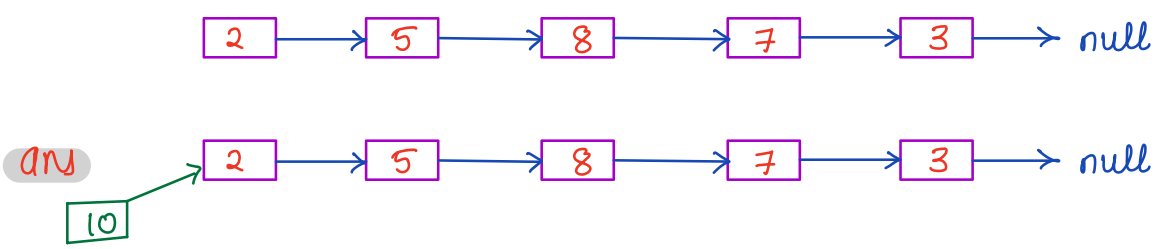
$0 <= K <= N$

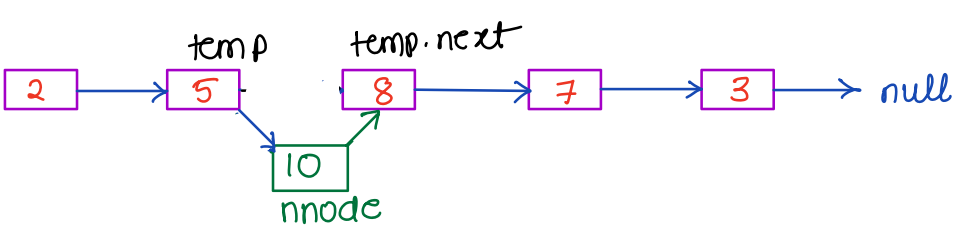


At K=2 insert X = 10



K = 0 , X = 10





nnode.next = temp.next

temp.next = nnode

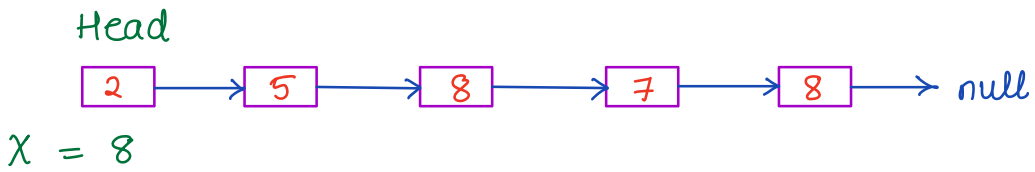
```
Node insertAtK ( Node head , int x , int k ) {
    if ( k == 0 ) {
        Node nnode = new Node(x)
        nnode.next = head
        head = nnode
        return head
    }

    temp = head

    // make k-1 jumps
    for i ⟶ 1 TO k-1 {
        temp = temp.next
    }

    Node nnode = new Node(x)

    nnode.next = temp.next
    temp.next = nnode

    return head
}
```

TC : O(K)

SC : O(1)

Q → Delete the first occurrence of value X in the given linked list. {If X is not present dont change}

Return head of linked list



X = 8

ans

Edge Cases

→ head == null

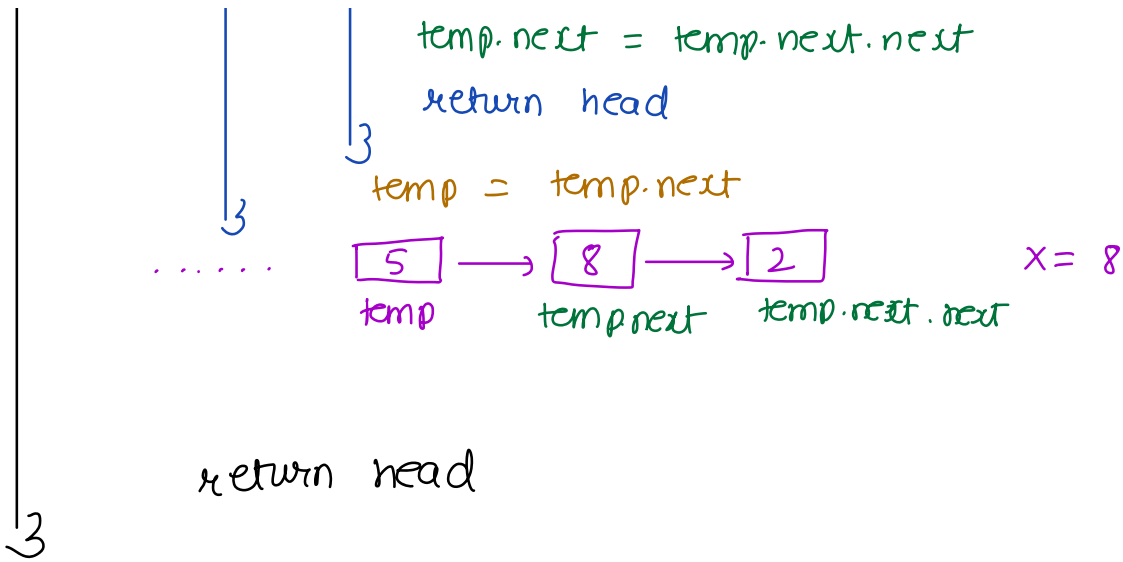
→ X is not present

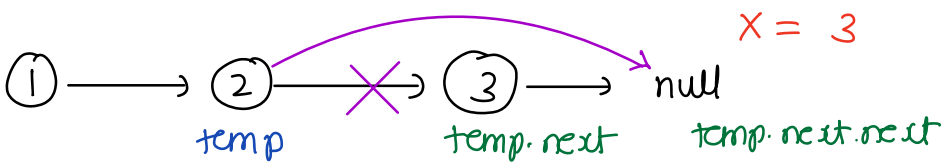
→ head.val == X

```
Node deleteFirstX ( Node head , int X ) {
    if (head == null) { return head }

    if ( head.data == x ) {
        return head.next
    }

    temp = head

    while ( temp.next != null) {
        if (temp.next.data == X ) {
```

TC : O(N)

SC : O(1)



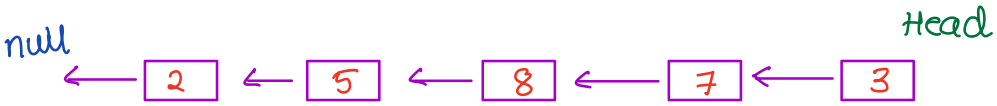


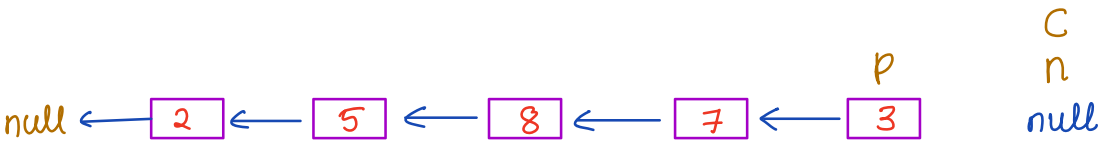
Break : 22 : 35

Pls revise the content till now and post Qs in Q tab.

Neuron
Cisco
Walmart
Globant
Credit Suisse

Reverse the Linked List *****






```
Node   reverseList (Node head) {
        prev = null
        curr = head

        while (curr != null) {
            nxt = curr.next
            curr.next = prev
            prev = curr
            curr = nxt
        }
        return prev
}
```

TC: O(N)

SC: O(1)

Find the length of linked list

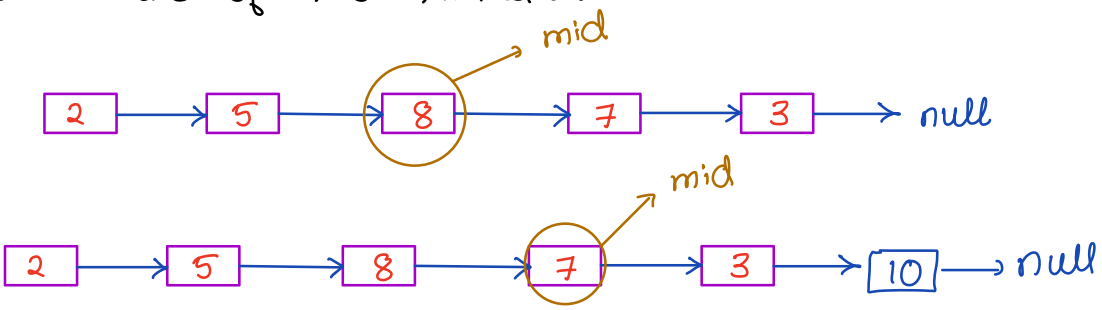
i/p

o/p ans = 5

TC: O(N)

```
temp = head        cnt = 0
while ( temp != null) {
        cnt++
        temp = temp.next
}
print(cnt)
```

Find middle of the linked list

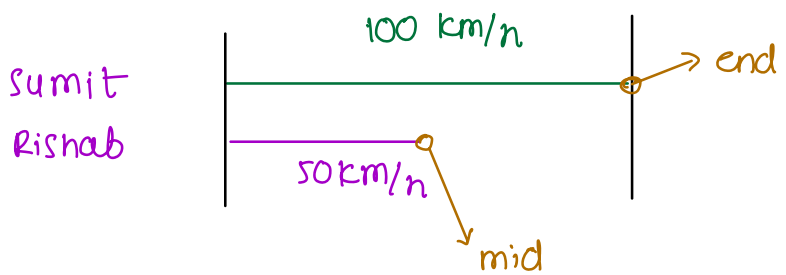


⟶ count size of LL

```
temp = head
mid = (size of LL from above code)/2
for i ⟶ 1 TO mid
        temp = temp.next
print(temp.data)
```

use the to cal mid in OA or coding platform on test

Calculate mid with only 1 traversal



Idea ⟶ Maintain two nodes slow and fast

slow moves at a speed of 1

fast moves at a speed of 2

POP

1 2 1

amma

madam

Q → check if the given linked list is a palindrome.

reads same forward & backward

ans = false

ans = true

Bruteforce

TC : O(N)

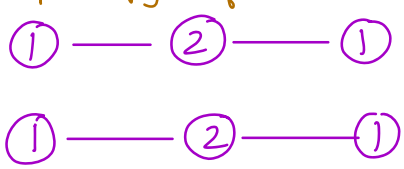
SC : O(N)

Convert LL into an array

Then check for palindrome

Brutcforce 2

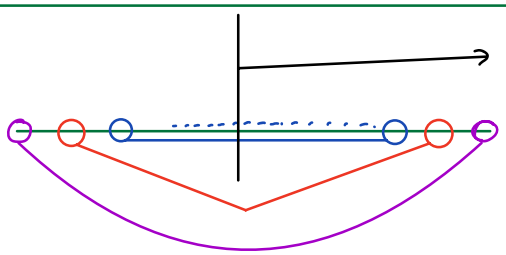
Make a deep copy of linked list

I/O → 

LL1

LL2 { deep copy }

reverse LL2 and compare all nodes of LL1 & LL2

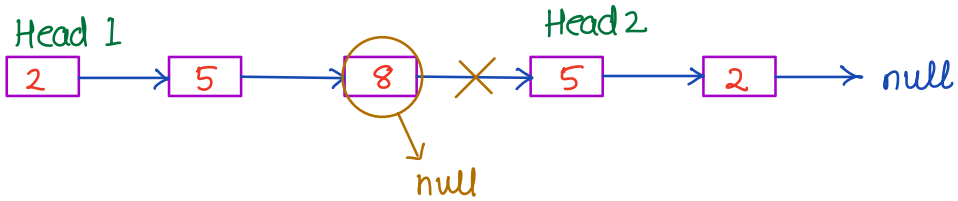


compare values

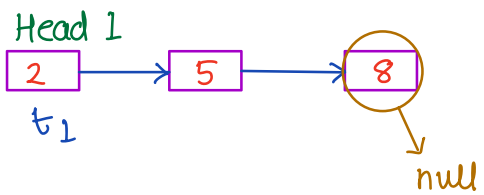
x^{th} first val == x^{th} last val

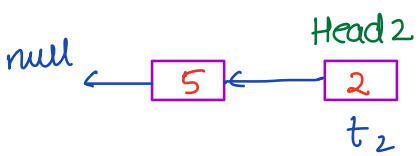
Find middle

Break the LL after middle



Reverse Head 2



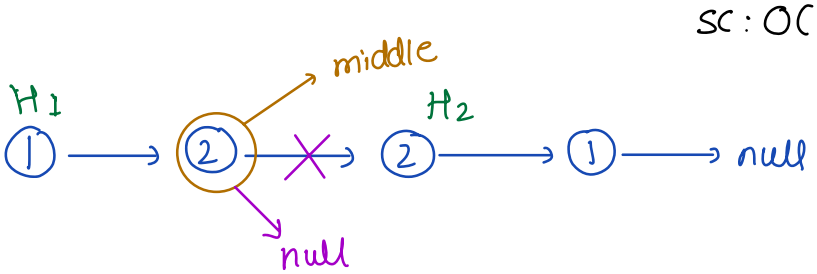


now compare the values and keep moving while temp 1 or temp 2 is not null.

TODO ⟶ write the above code

TC : O(N)

SC : O(1)




Head
1
null
am = true