Lecture : Linked List - 1

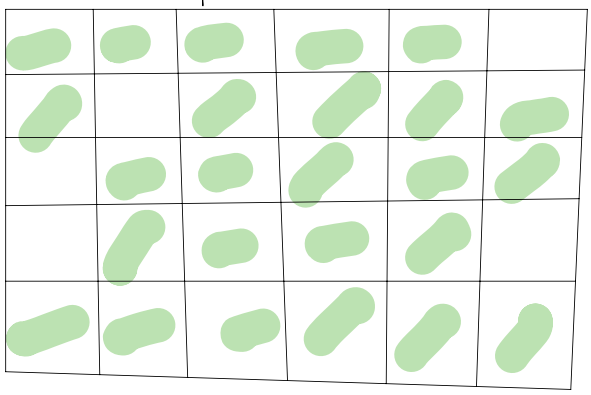
Agenda

- Introduction
- Insert | Delete | find | Access operations
- Reverse LL
- Palindrome LL.

Issues in array

only stores the elements in continuous manner

heap location



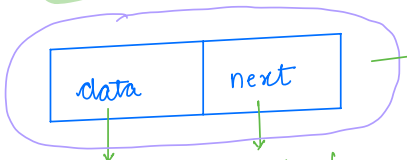
Green – Used memory.

arr[5] ⇒ Not possible

Linked list

- Linear data structure that can utilize all free memory.
- non-continuous memory allocation.

Representation of Linked List



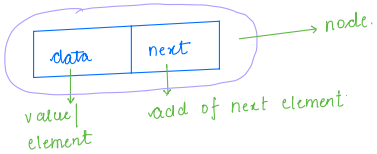
node.

value | element

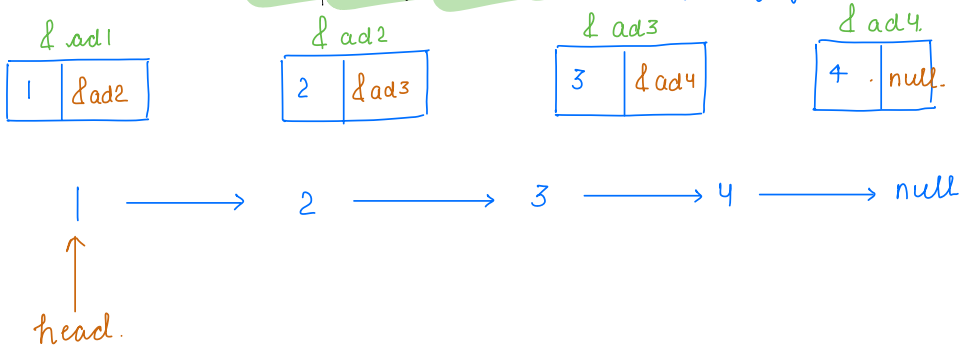
add of next element.

Structure of Linked List

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



Examples of Linked List { singly LL }



Operations on linked list

{ Though indexing doesn't exist in LL, assume node1 as 0th idx, node2 as 1st idx ------- and so on.

1.) Access kth element.

18 ⟶ 16 ⟶ 23 ⟶ 1 ⟶ 4 ⟶ null

k = 1 16

k = 3 1

k = 4 4

18 ⟶ 16 ⟶ 23 ⟶ 1 ⟶ 4 ⟶ null , K = 3.

↑ head.

head = 18 , temp = 18

temp = temp next 16

temp = temp next 23

temp = temp next 1

```
Node temp = head;
for( i = 1; i <= k; i++) {
    temp = temp.next;
}
return temp.data
```

{ k = 4, Do dry run }

Search for a value in Linked List

18 → 16 → 23 → 1 → 4 → null

K = 16 true

K = 24 false

K = 23 true

```
boolean search( Node head, int x) {
        Node temp = head;
        while( temp != null) {
                if ( temp.data == x) {
                        return true;
                }
                temp = temp.next;
        }
        return false;
}
```

Insert new node

Though indexing doesn't exist in LL, assume node1 as 0th idx, node2 as 1st idx ------- and so on.

ip

0	1	2	3	4	5	
10 →	6 →	4 →	8 →	9 →	14 →	null

val = 60
idx = 3

op

0	1	2	3	4	5	6	
10 →	6 →	4 →	60 →	8 →	9 →	14 →	null
		prev	newNode	agla			

agla = prev.next // 4.next = 8.

prev.next = newNode; // 4.next = 60

newNode.next = agla // 60.next = 8

ip

0 → 1 → 2 → 3 → 4 → 5 → null

10 → 6 → 4 → 8 → 9 → 14 → null

val = 80
idx = 4

op

0 → 1 → 2 → 3 → 4 → 5 → 6 → null

10 → 6 → 4 → 8 → 80 → 9 → 14 → null

prev newNode agla.

agla = prev.next || agla = 9.

prev.next = newNode || 8 → 80

newNode.next = agla || 80 → 9.

0 → 1 → 2 → 3 → 4 → 5 → null

10 → 6 → 4 → 8 → 9 → 14 → null

idx = 6
val = 90.

0 → 1 → 2 → 3 → 4 → 5 → null

10 → 6 → 4 → 8 → 9 → 14 → 90 → null

prev. n. agla.

agla = prev.next; || null

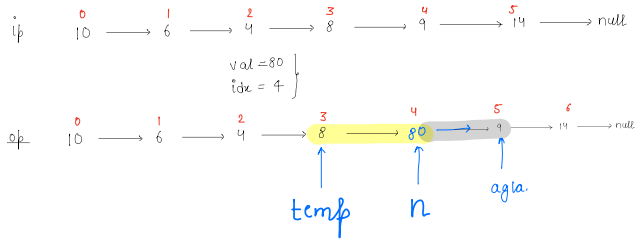
prev.next = n; 14 → 90

n.next = agla; 90 → null

Pseudocode

idx. , val.

```
temp = head;
for(i=1; i< idx; i++) {
        temp = temp.next;
}
Node n = new Node(80);
agla = temp.next;
temp.next = n;
n.next = agla;
}
```



Edge case:

ip $10 \longrightarrow 6 \longrightarrow 4 \longrightarrow 8 \longrightarrow 9 \longrightarrow 14 \longrightarrow null$

val = 80
idx = 0

op $80 \longrightarrow 10 \longrightarrow 6 \longrightarrow 4 \longrightarrow 8 \longrightarrow 9 \longrightarrow 14 \longrightarrow null$

n

n.next = head;
head = n;

final pseudocode

```
void insert( Node head, int val, int idx)  {
        Node n = new Node(val);
        temp = head;
        if (idx==0) {
            n.next = head;
            head = n;
            return;
        }
        for (i=1; i< idx; i++) {
            temp = temp.next;
        }

        agla = temp.next;
        temp.next = n;
        n.next = agla;
}
```

Qu Delete first occurence of a value in linked list.

ip $1 \longrightarrow 8 \longrightarrow 4 \longrightarrow -2 \longrightarrow 12$, $x = -2$.

op $1 \longrightarrow 8 \longrightarrow 4 \longrightarrow \cancel{-2} \longrightarrow 12$

ip $1 \longrightarrow 8 \longrightarrow \cancel{4} \longrightarrow -2 \longrightarrow 4 \longrightarrow 12$ $x = 4$

op $1 \longrightarrow 8 \longrightarrow -2 \longrightarrow 4 \longrightarrow 12.$

Cases

1.) Empty list head==null.

null.

$x = 4$.

ans: null.

2.) Delete head. , head.data = x

LL :- $4 \longrightarrow$ null

$x = 4$.

ans: null

return null.

$4 \longrightarrow 8 \longrightarrow 16 \longrightarrow 20 \longrightarrow$ null

del: 4

ans: $8 \longrightarrow 16 \longrightarrow 20 \longrightarrow$ null

3) x is somewhere in b/w the list

$1 \longrightarrow 8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 4 \longrightarrow 12.$

del = 4

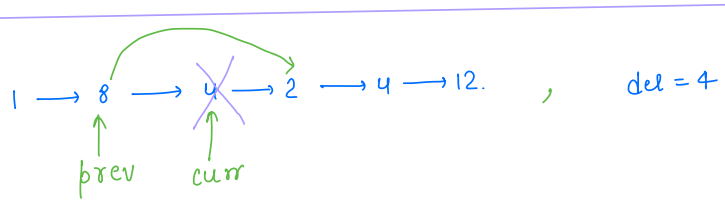
ans: $1 \longrightarrow 8 \longrightarrow 2 \longrightarrow 4 \longrightarrow 12$

4) x is not in the list

$1 \longrightarrow 8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 4 \longrightarrow 12.$

del = 108

ans: $1 \longrightarrow 8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 4 \longrightarrow 12.$

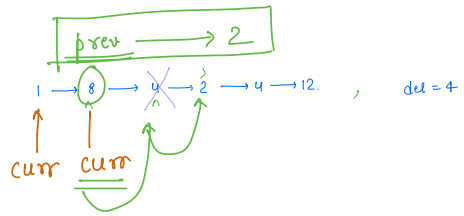


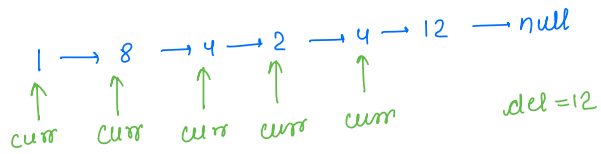
, del = 4

prev.next = curr.next;

Pseudocode

```
Node delete first occ( Node head, int x) {
    if ( head == null) {
        return null;
    }
    if (head.data == x) {
        head = head.next;
        return head;
    }
    curr = head;
    while(curr.next != null) {
        if ( curr.next.data == x)
            prev = curr;
            prev.next = curr.next.next;
            return head;
        }
        curr = curr.next;
    }
    return head;
}
```





Break: 8:34–8:44

Qu Reverse the linked list. [Do not use extra space. Manipulate the pointers only]

ip $2 \rightleftarrows 5 \rightleftarrows 8 \rightleftarrows 7 \rightleftarrows 3 \longrightarrow null$

op $3 \longrightarrow 7 \longrightarrow 8 \longrightarrow 5 \longrightarrow 2 \longrightarrow null$

Intuition

$2 \longrightarrow 5 \longrightarrow 8 \longrightarrow 7 \longrightarrow 3 \longrightarrow null$

$2 \longleftarrow 5 \longleftarrow 8 \longrightarrow 7 \longrightarrow 3 \longrightarrow null$

prev

curr

7.next = 8 [$7 \longrightarrow 8$]

7.next = 8

curr.next = prev

Approach

ip 2 → 5 → 8 → 7 → 3 → null

prev = null
curr = 2
agla = null

Itr1

2 → 5 → 8 → 7 → 3 → null

prev curr

agla = curr.next; // 5
curr.next = prev [2 → null] ✓
prev = curr
curr = agla; // 5

Itr2

null ← 2 5 → 8 → 7 → 3 → null

prev curr

agla = curr.next; // 8
curr.next = prev // 5 → 2
prev = curr // 5
curr = agla; // 8

Itr3

null ← 2 ← 5 8 → 7 → 3 → null

prev curr

agla = curr.next // 7
curr.next = prev // 8 → 5
prev = curr // 8
curr = agla // 7

Itr 4

null ← 2 ← 5 ← 8 ← 7 → 3 → null

prev: curr

agla = curr.next || 3

curr next = prev || 7 → 8

prev = curr || 7

curr = agla || 3

Pseudocode

```
Node rev( Node head) {
    if (head == null) {
        return head;
    }
    if (head.next == null) {
        return head;
    }
    prev = null;
    curr = head;
    while(curr != null) {
        agla = curr.next;
        curr.next = prev;
        prev = curr;
        curr = agla;
    }
    head = prev;
    return head;
}
```

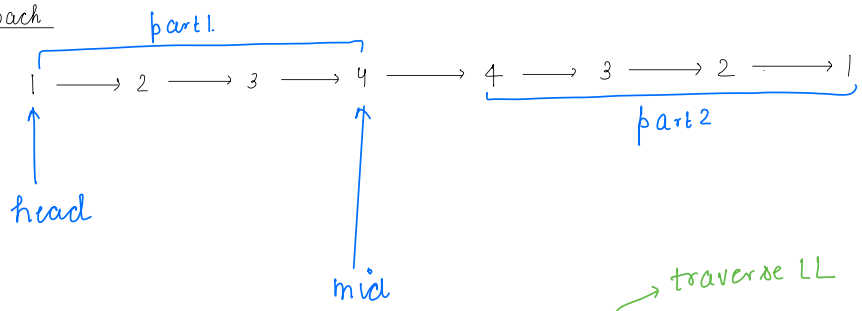
Qu Check if linked list is palindrome.

2 → 5 → 8 → 7 → 3 → null false

2 → 5 → 8 → 5 → 2 → null true

Expected SC : O(1)

Approach



1.) find mid. ⟶ access $\frac{len-1}{2}$ element. (traverse LL)

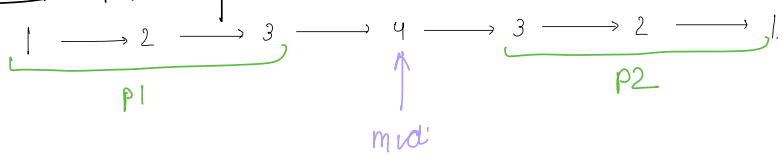
2.) rev(mid.next) // rev second half of LL

3.) Compare fir & rev(second half)

fir = 1 → 2 → 3 → 4

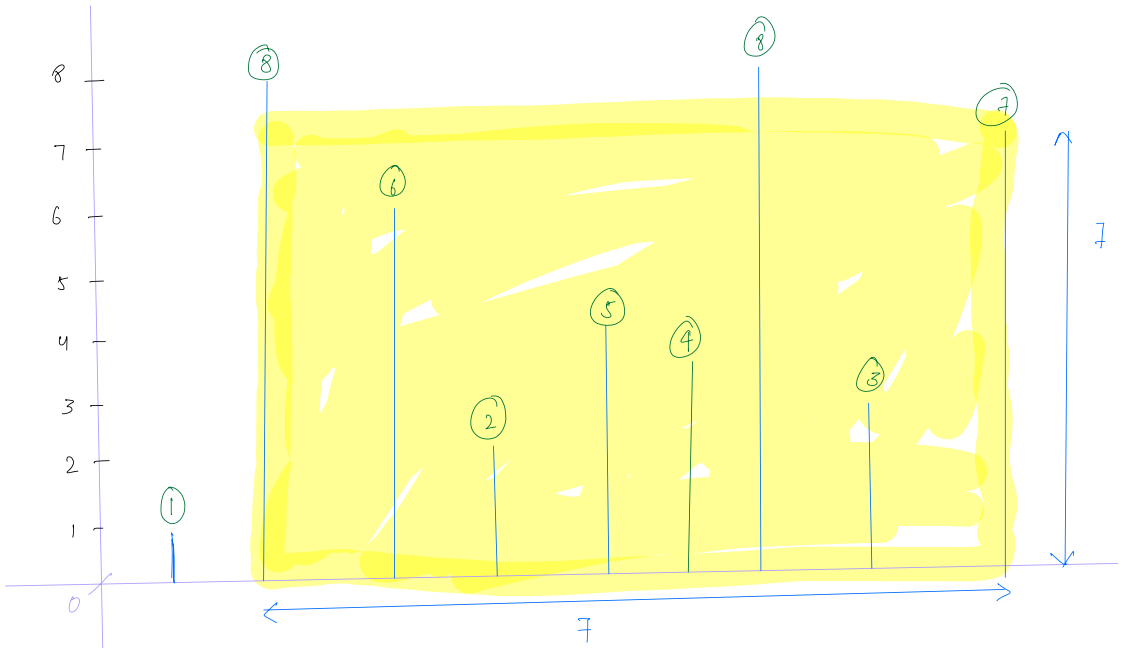
sec = 1 → 2 → 3 → 4

Catch: Think for odd length LL.



Thankyou ☺

Doubts [1, 8, 6, 2, 5, 4, 8, 3, 7]



ans = 7 * 7 = 49

0	1	2	3	4	5	6	7
4	2	10	6	8	2	6	2

ans = 24