Data Structure & Algorithms

Sunbeam Infotech

Bubble Sort

6 4 2 8 3 1

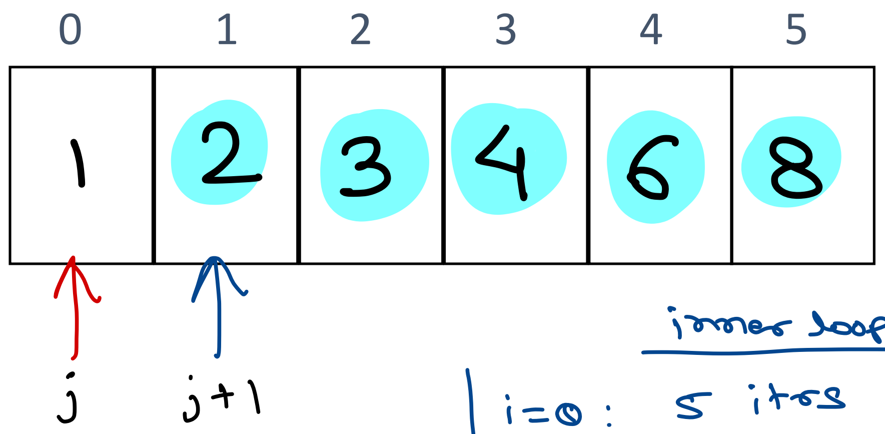
```
for(i=0; i<n-1; i++) {
    for(j=0; j<n-1; j++) {
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]);
    }
}
```

$i = (n-1) * (n-1)$

$T \propto n^2 - 2n + 1$

$T \propto n^2$

$\boxed{O(n^2)}$



inner loop

$(n-1)$ Passes

i=0 :	5 itrs
i=1 :	5 itrs
i=2 :	5 itrs
i=3 :	5 itrs
i=4 :	5 itrs

$+ \ \boxed{25}$

$(n-1)$ comp

Bubble Sort - improved Bubble Sort

```
for(i=0; i<n-1; i++) {
    for(j=0; j<n-1-i; j++) {
        if(a[j] > a[j+1])
            swap(a[j], a[j+1]).
```

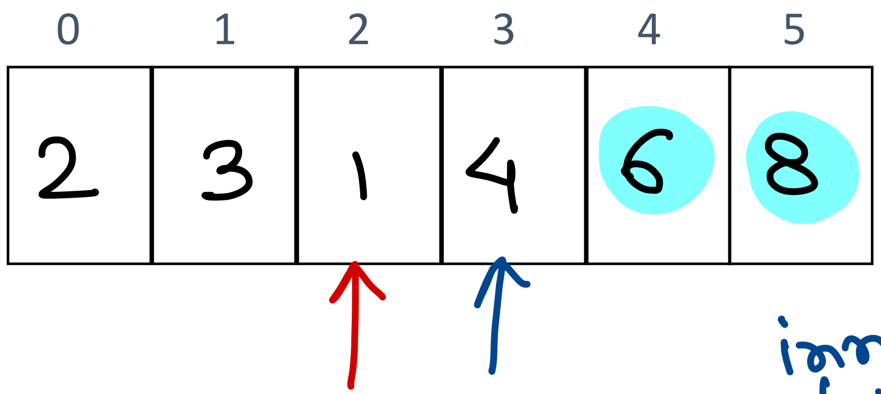
3

3

$i = (n-1) + (n-2) + \dots + 1$

$T \propto n^2 - n$

$\boxed{O(n^2)}$

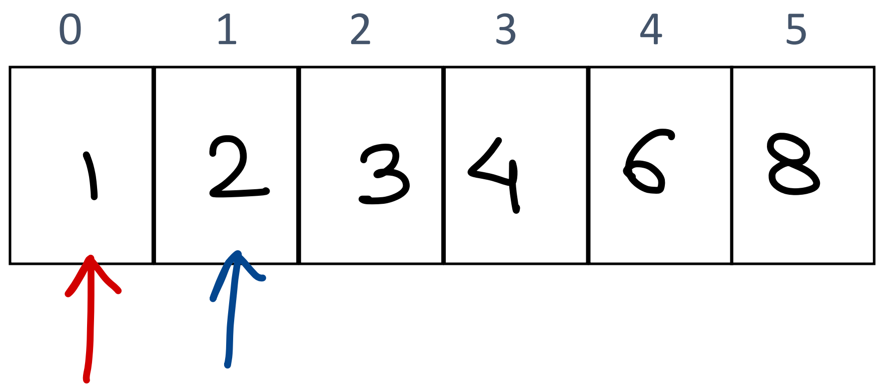


inner loop	
i=0 :	5
i=1 :	4
i=2 :	3
i=3 :	2
i=4 :	1

+ 15

Bubble Sort - more improved Bubble Sort

```
for(i=0; i<n-1; i++) {
    swapFlag = false;
    for(j=0; j<n-1-i; j++) {
        if(a[j] > a[j+1]) {
            swap(a[j], a[j+1]).
            swapFlag = true;
        }
    }
    if(swapFlag == false)
        break;
}
```



Best case : array already sorted.

* inner loop is executed only once.

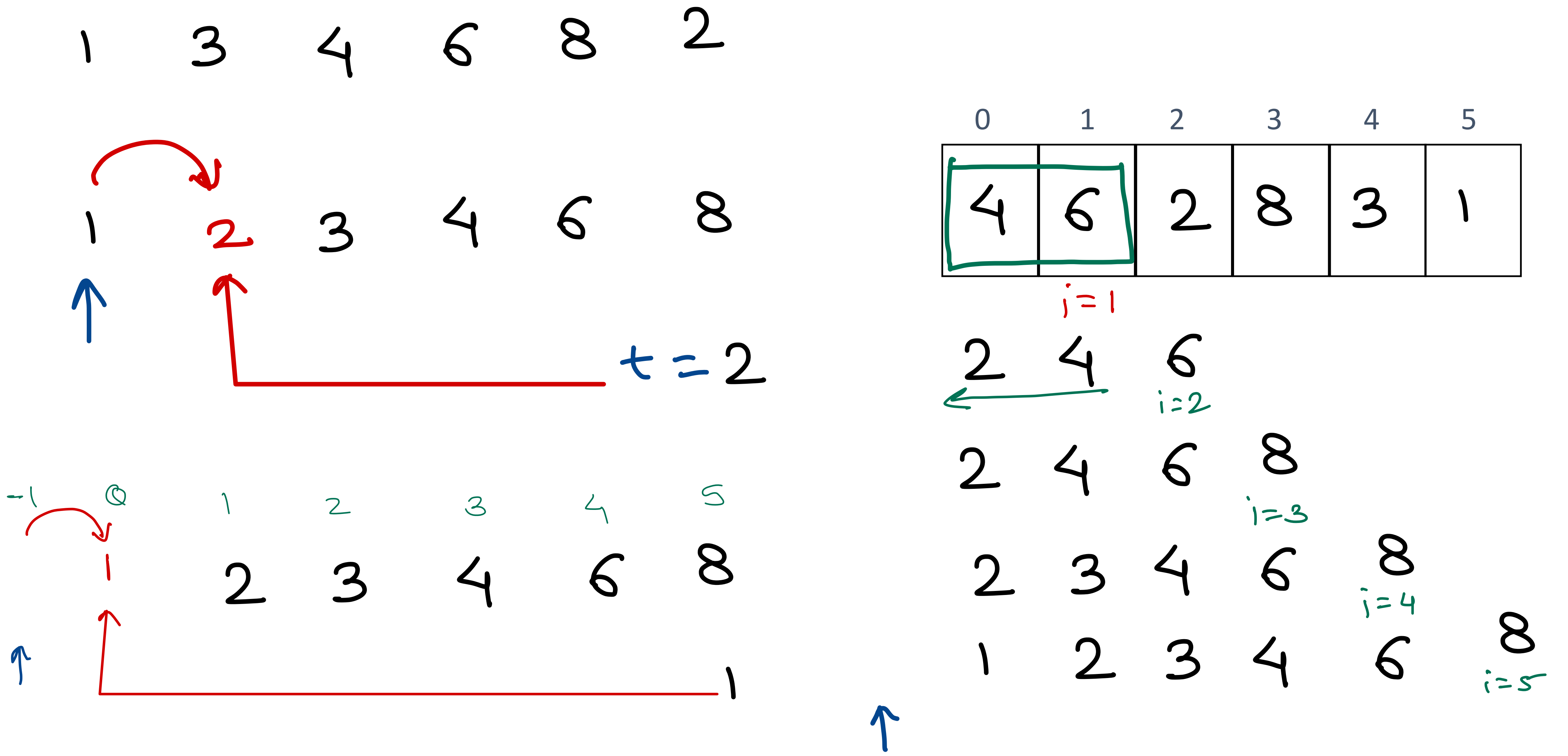
$i = n-1$

$T \propto n-1$

$\boxed{O(n)}$

Insertion Sort

6 4 2 8 3 1



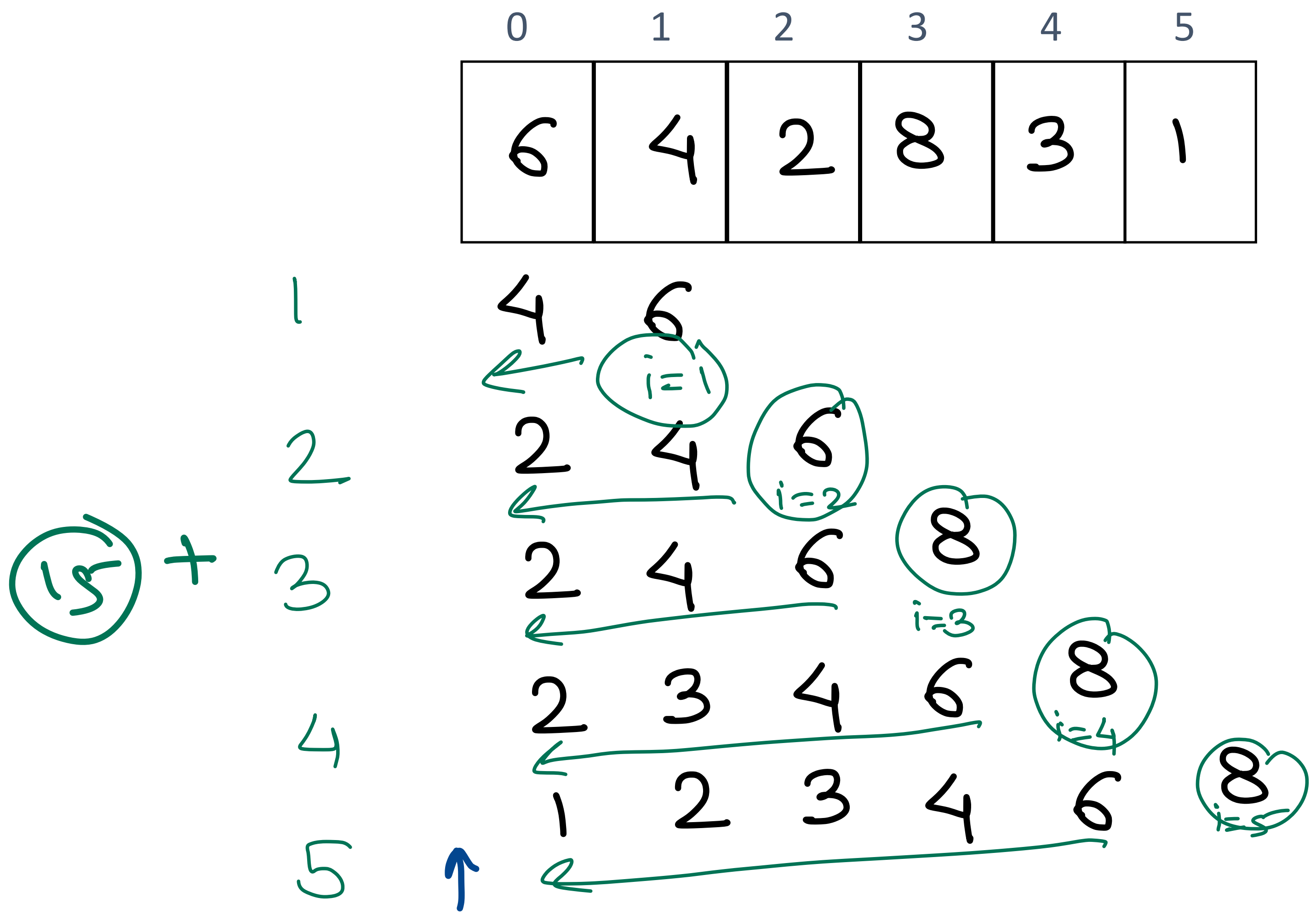
Insertion Sort

$i = 1 + 2 + \ldots + (n-1)$

$i = n(n-1)/2$

$T \propto n^2 - n$

$\boxed{O(n^2)}$



Insertion Sort

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

Best Case | $T \propto n$

$O(n)$

Linked List

- Linked List is list of items linked together.
- Each item in linked list is called as Node.
- Each node contains data and pointer/reference to the next node.
- Linked list is linear data structure.

→ can grow/shrink at runtime
→ sequential access only.
→ overheads
→ not contiguous alloc.

Arrays

1. random access → O(1)
2. fixed size
   - cannot grow/shrink at runtime.
3. contiguous allocation
4. no overheads
   - no extra space required other than data elements.

- Linked list ADT
  - addFirst() ✓
  - addLast() ✓
  - addAtPos() ✓
  - deleteFirst() ✓
  - deleteLast() ✓
  - deleteAsPos() ✓
  - deleteAll() ✓
  - traverse() or display()

Linked List

- There four types of linked list.
  - Singly linear linked list
  - Singly circular linked list
  - Doubly linear linked list
  - Doubly circular linked list

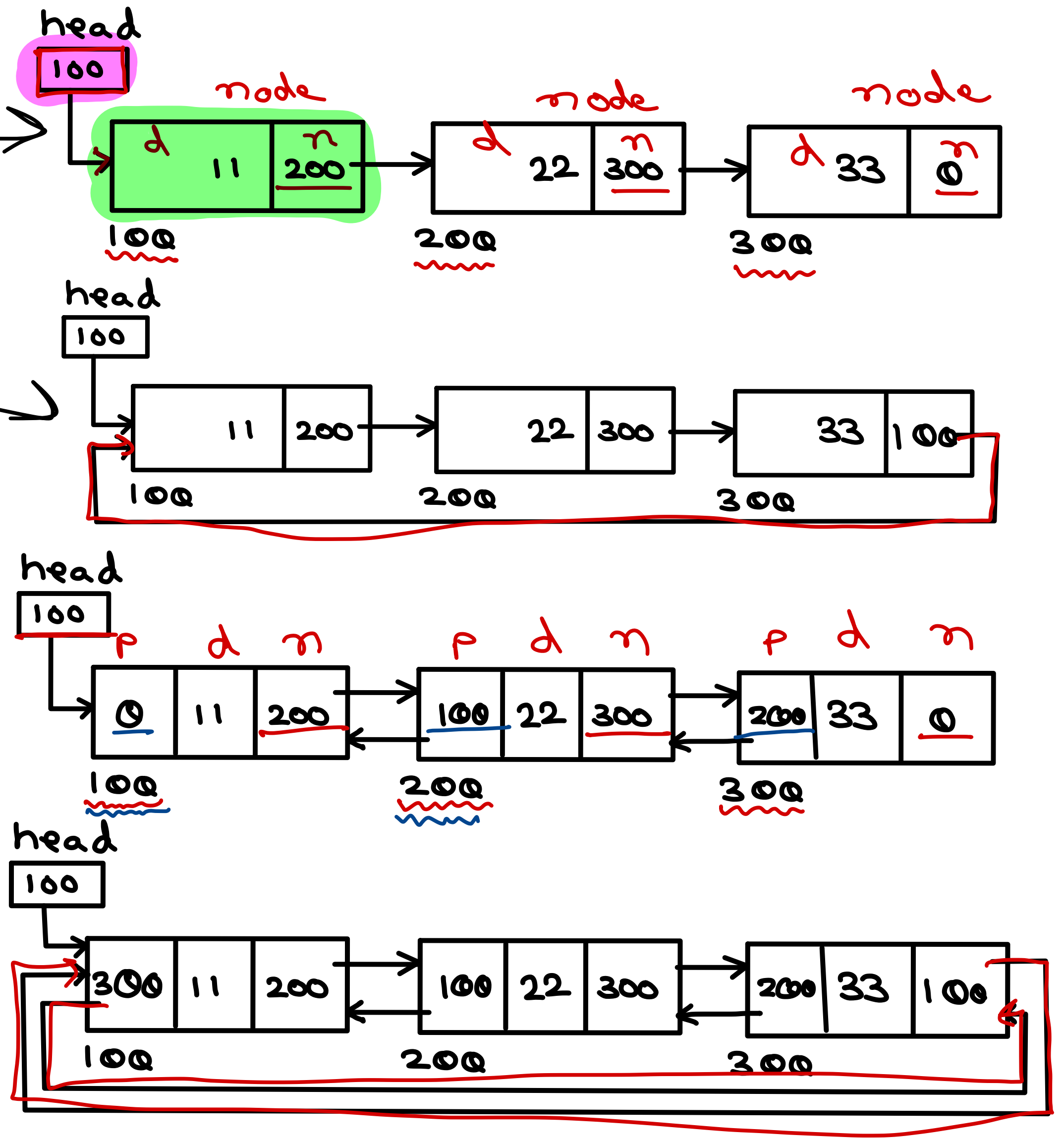
```
class MyList {
data { Node head;

public
ops { display()
      addFirst()
      delFirst()
};
```

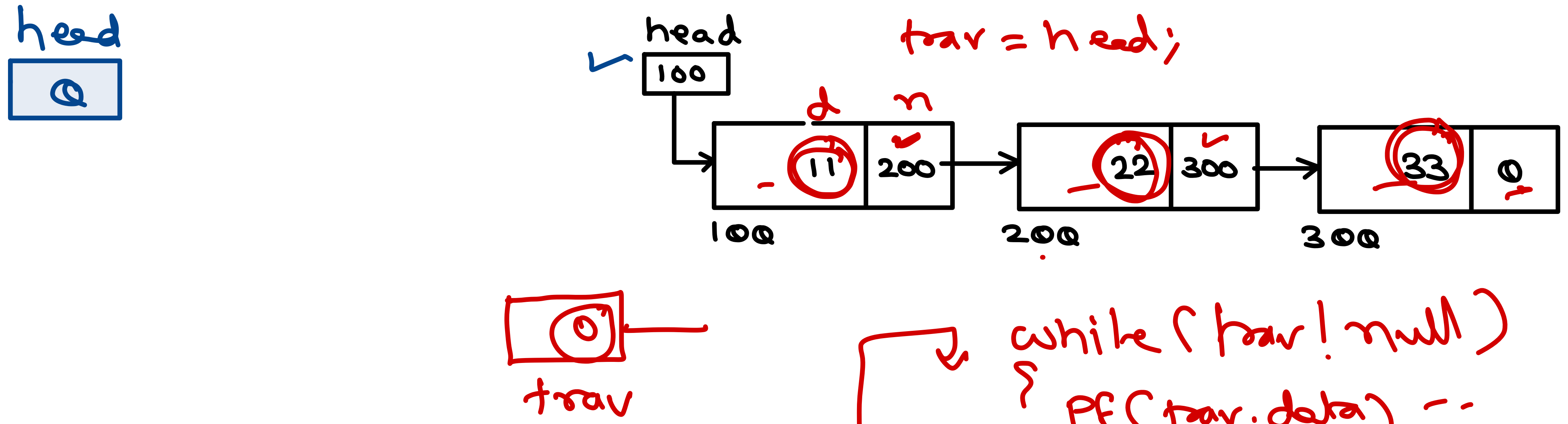
Java

static member class

```
class Node {
data { type data;
       Node next;
ops { ctor()
      get/set
}
```

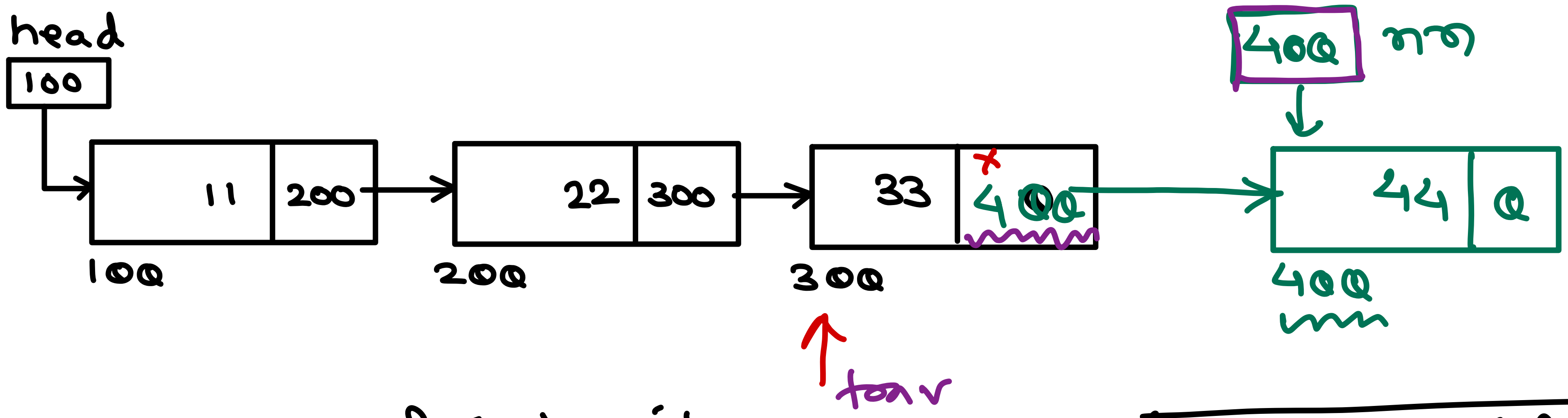


Linked List – display



```
while (trav != null)
{
    pf(trav.data) --
    trav = trav.next;
}
```

Linked List - add Last()



① create new node & init it.

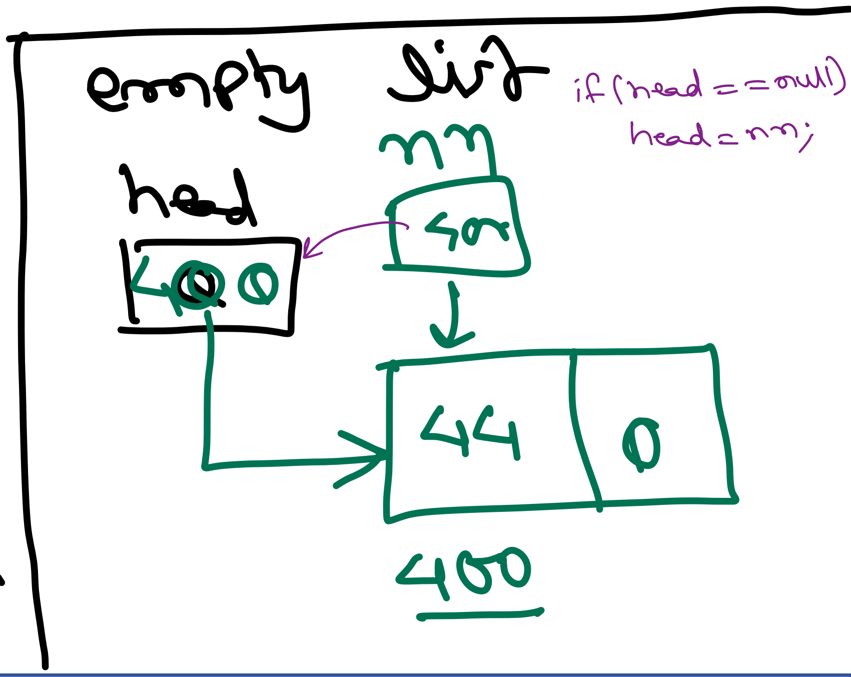
```
Node nn = new Node(val);
```

② traverse till last node.

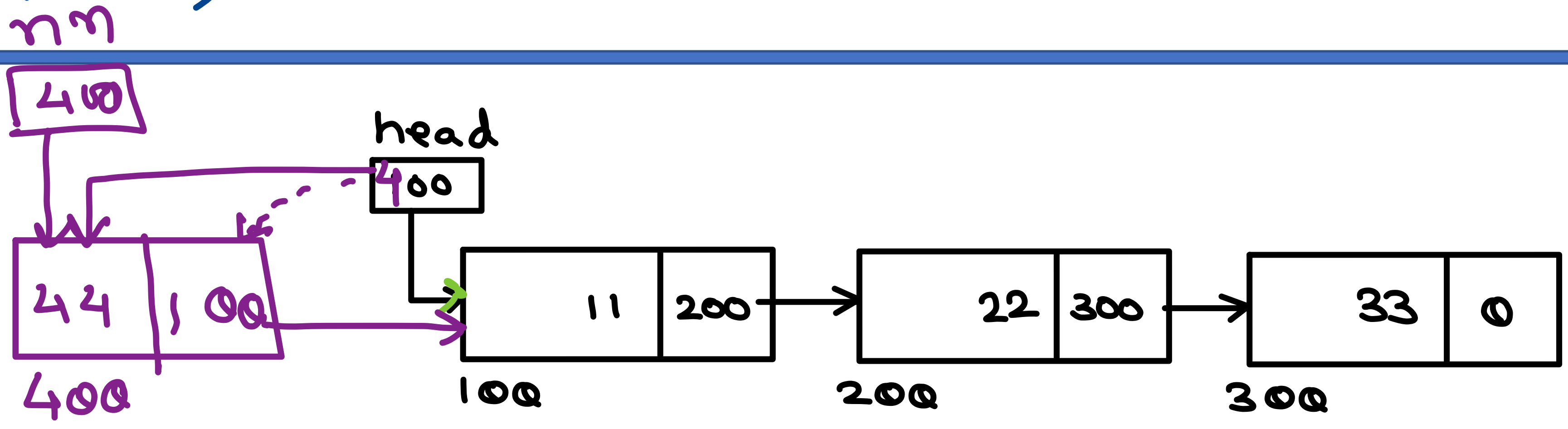
```
Node trav = head;
while(trav.next != null)
    trav = trav.next;
```

③ add new node into next of last node.

```
trav.next = nn;
```



Linked List - addFirst()



① alloc & init new node

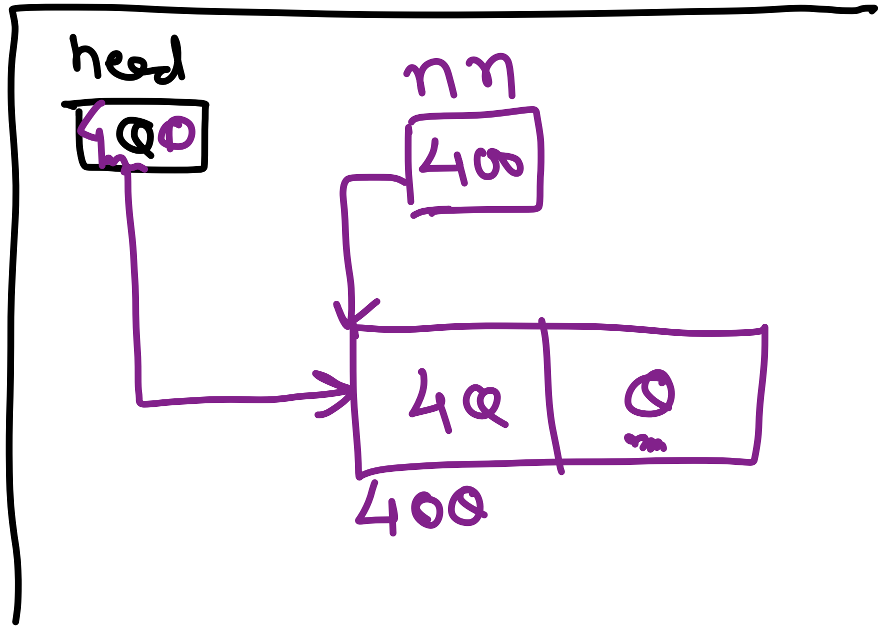
Node nn = new Node(val);

② new node next should first node addr

nn.next = head;

③ head should be new node.

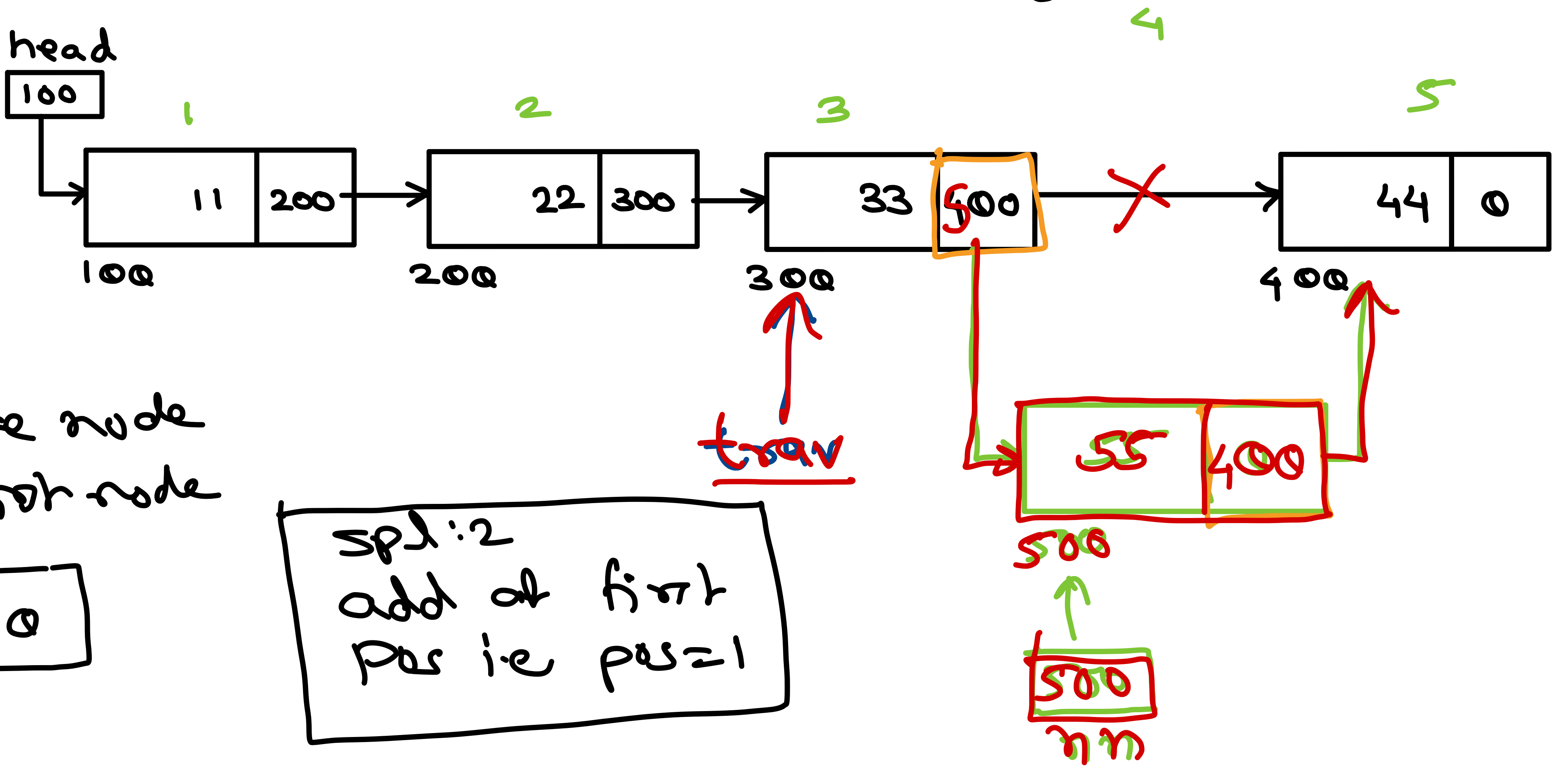
head = nn;



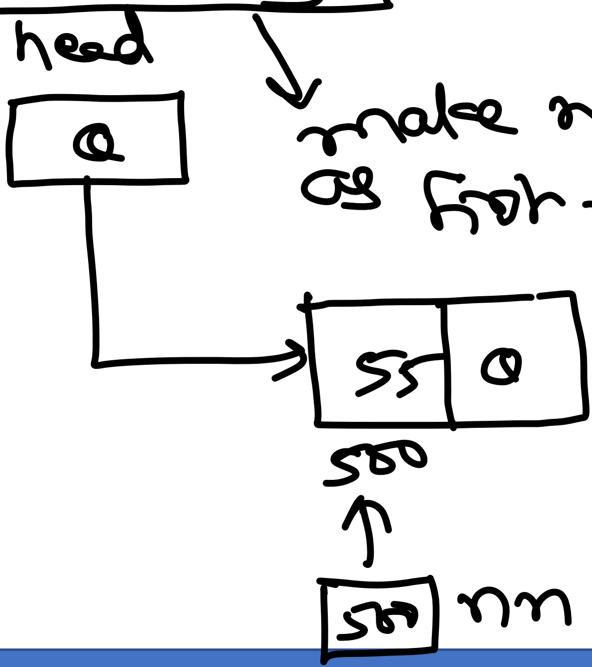
Linked List - addAtPos()

make before break

spl:3 add node beyond pos.



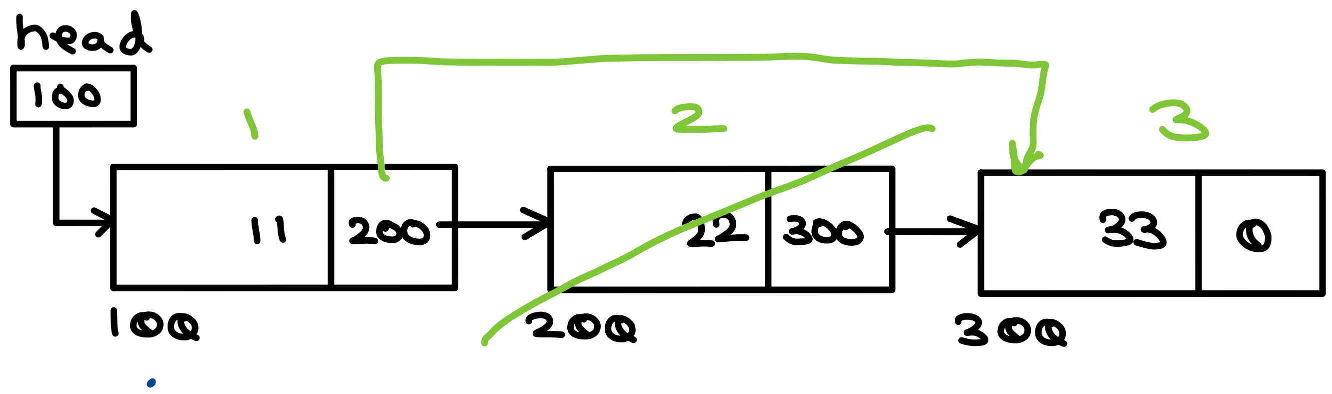
spl:1
list empty

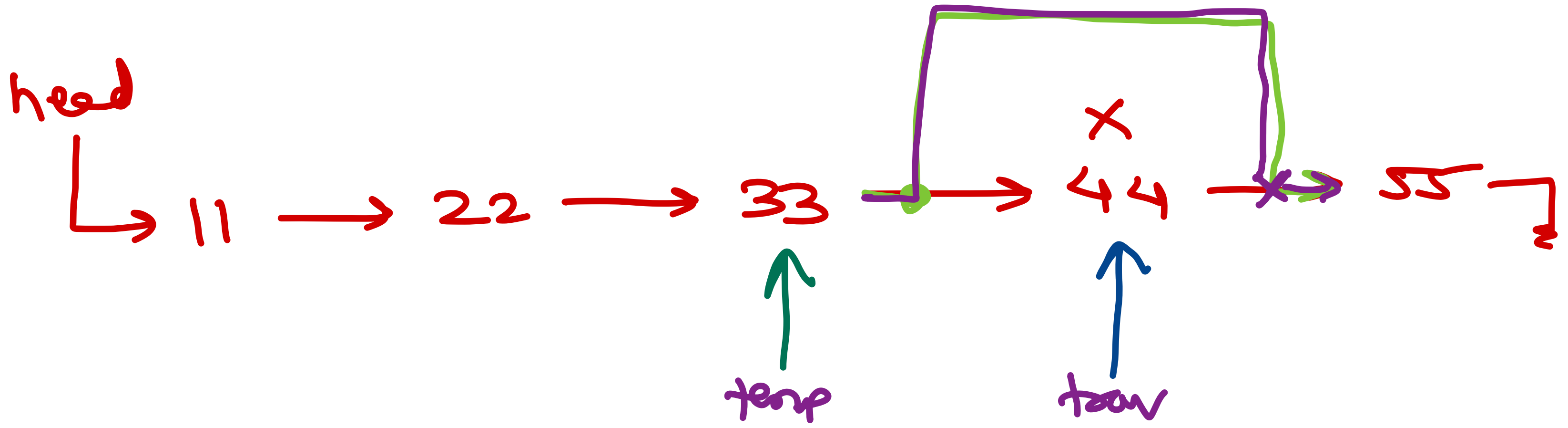


make node as first node

spl:2
add at first pos i.e pos=1

Linked List


head
100
1
2
3
11
200
22
300
33
0
100
200
300


head
11
22
33
44
55
temp
trav

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>