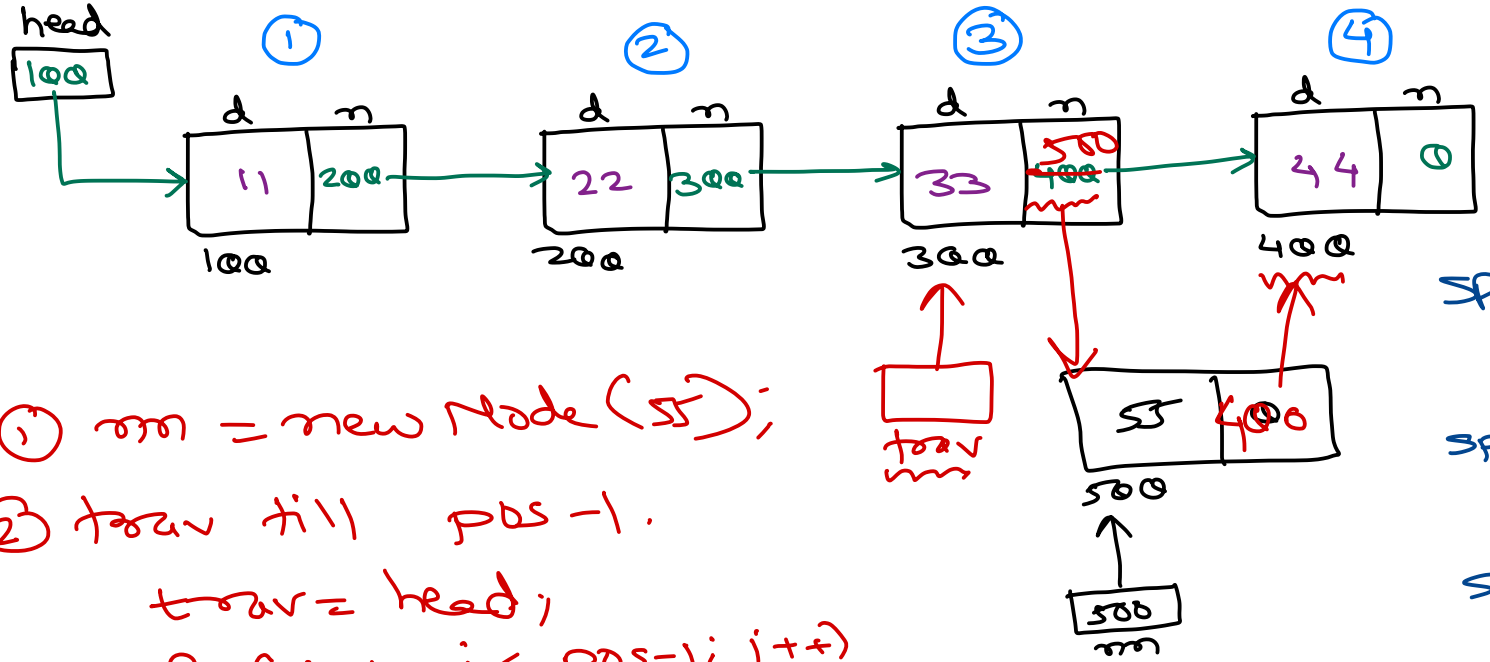
Data Structure & Algorithms

Sunbeam Infotech

Nilesh Ghule

Singly Linear Linked List – insert op → at a pos / after a node / before a node.



make before break

special 1 → if list is empty.
→ add node at start.

special 2 → if pos == 1,
→ add node at start.

special 3 → if pos < 1,
→ add node at start.

special 4 → if pos > max+1
→ add node at the end.

① nn = new Node(55);

② trav till pos-1.

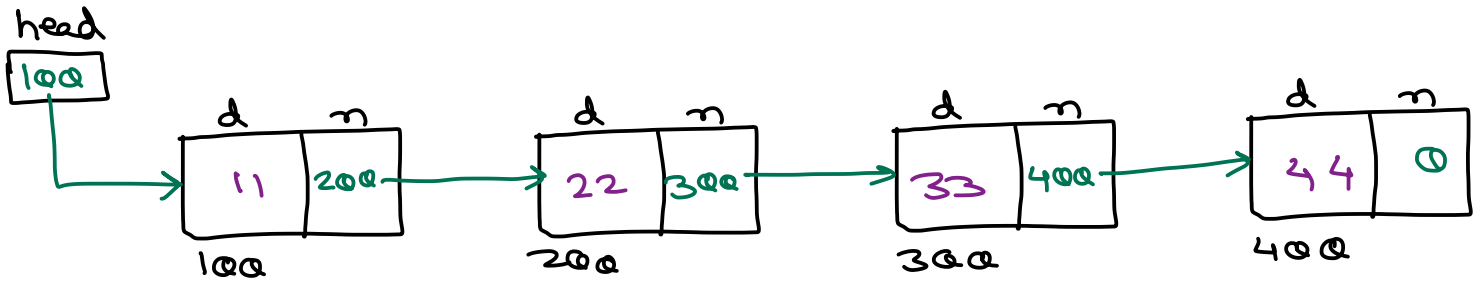
```
trav = head;
for(i=1; i<pos-1; i++)
    trav = trav.next;
```

```
if(trav.next == null)
    break;
```

③ nn.next = trav.next;

④ trav.next = nn;

Singly Linear Linked List – del first



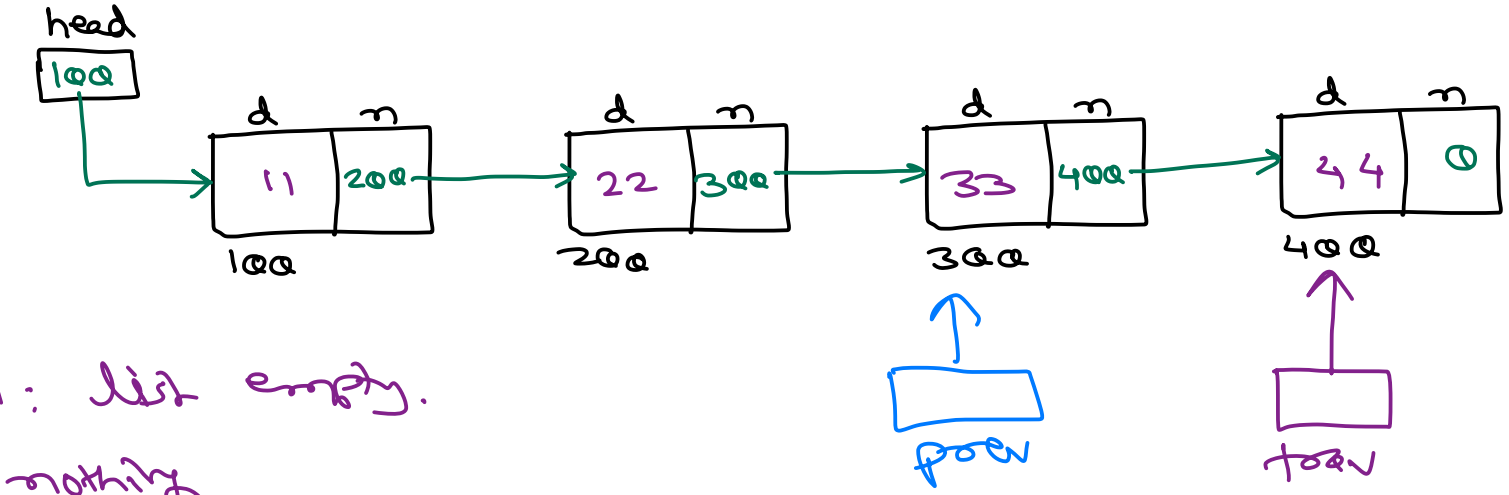
head = head.next

↓

first node (11) will be garbage collected.

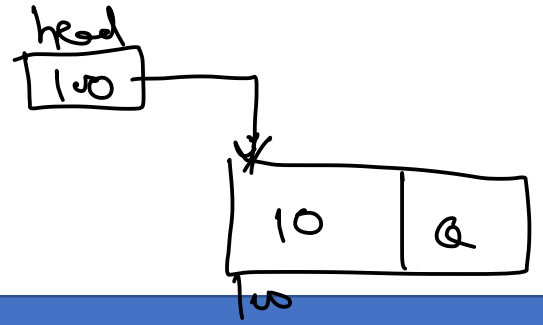
special: if list is empty,
↳ do nothing (return)

Singly Linear Linked List - del last



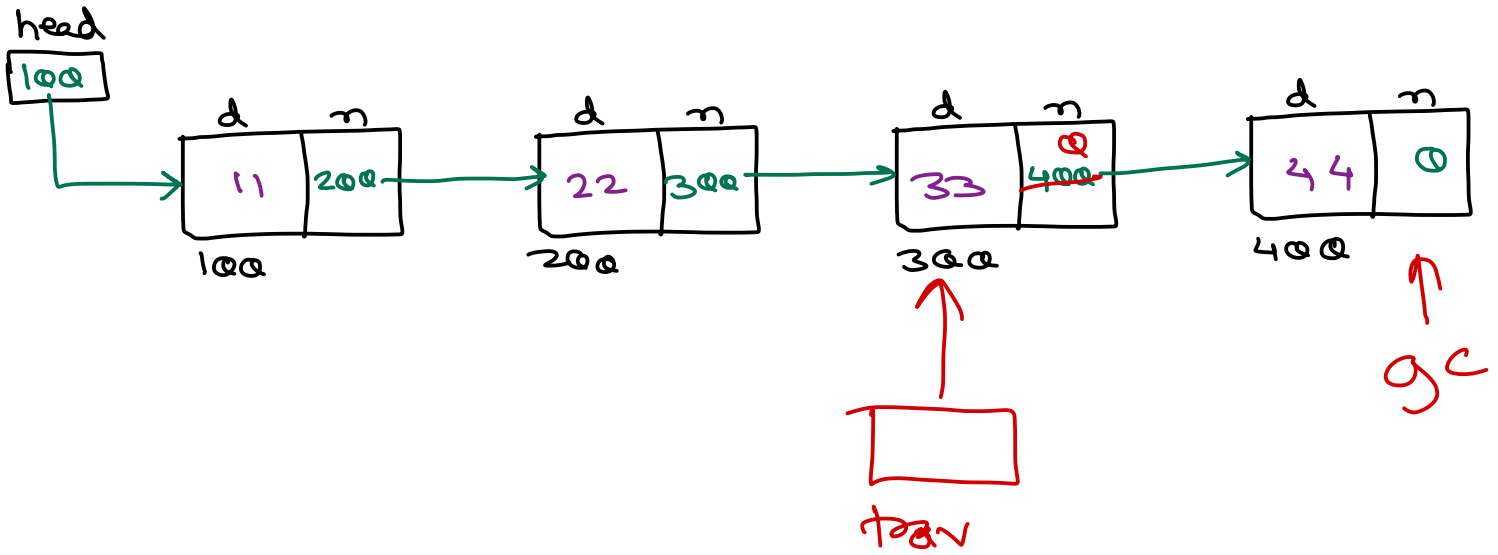
Special 1: list empty.
- do nothing.

Special 2: if list has only one node.
- del first node



1. trav till last node, also maintain a prev ptr
2. prev.next = null;
3. trav will be garbage collected.

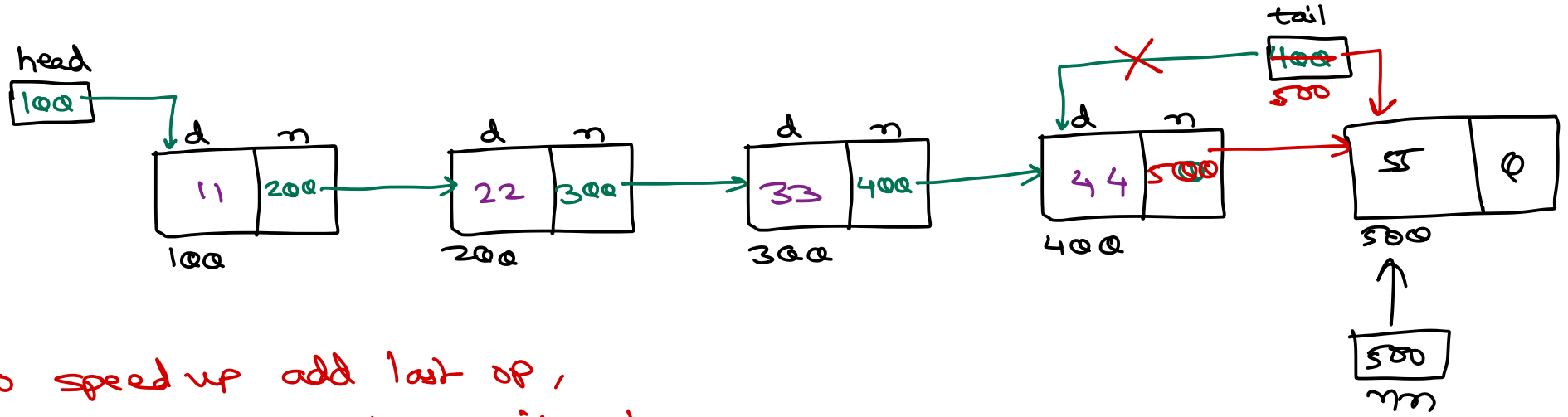
Singly Linear Linked List - del last.




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

trav.next = null;
```

Singly Linear Linked List




```
tail.next = nn;
tail = nn;
```

O(1)

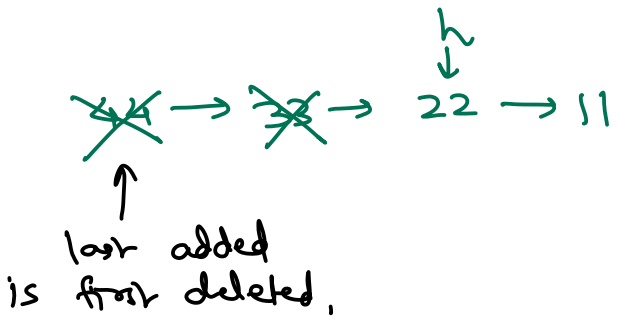
To speed up add last op,
we can maintain tail ptr
to keep addr of last node.

However, we should modify all
ops to change tail ptr
wherever required.

Stack / Queue using Linked List

- Stack can be implemented using linked list.
  - add first
  - delete first
  - is empty

OPS are done from same end i.e. head.
↓
LIFO

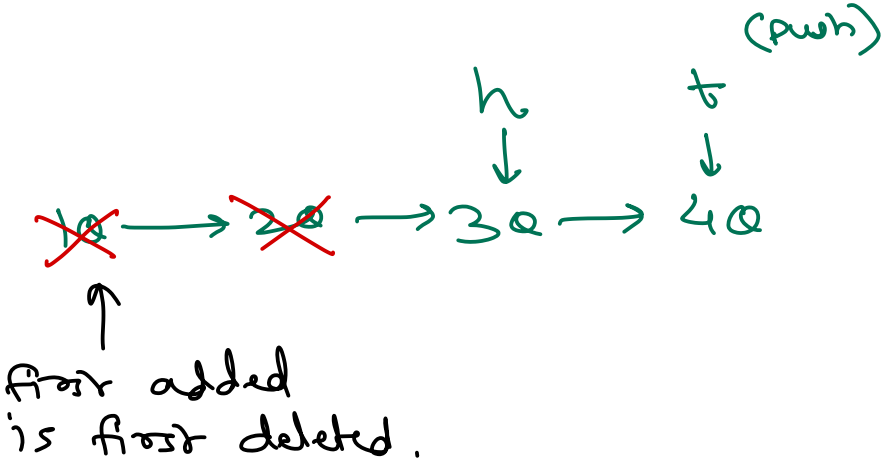


- Queue can be implemented using linked list.
  - add last
  - delete first
  - is empty

OPs are done from both ends. ie head & tail
(front) (rear)

O(1) if we try tail.

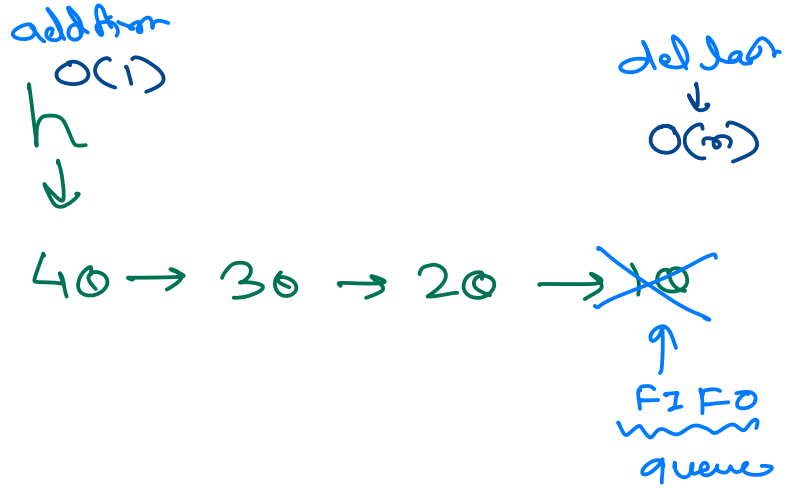
FIFO



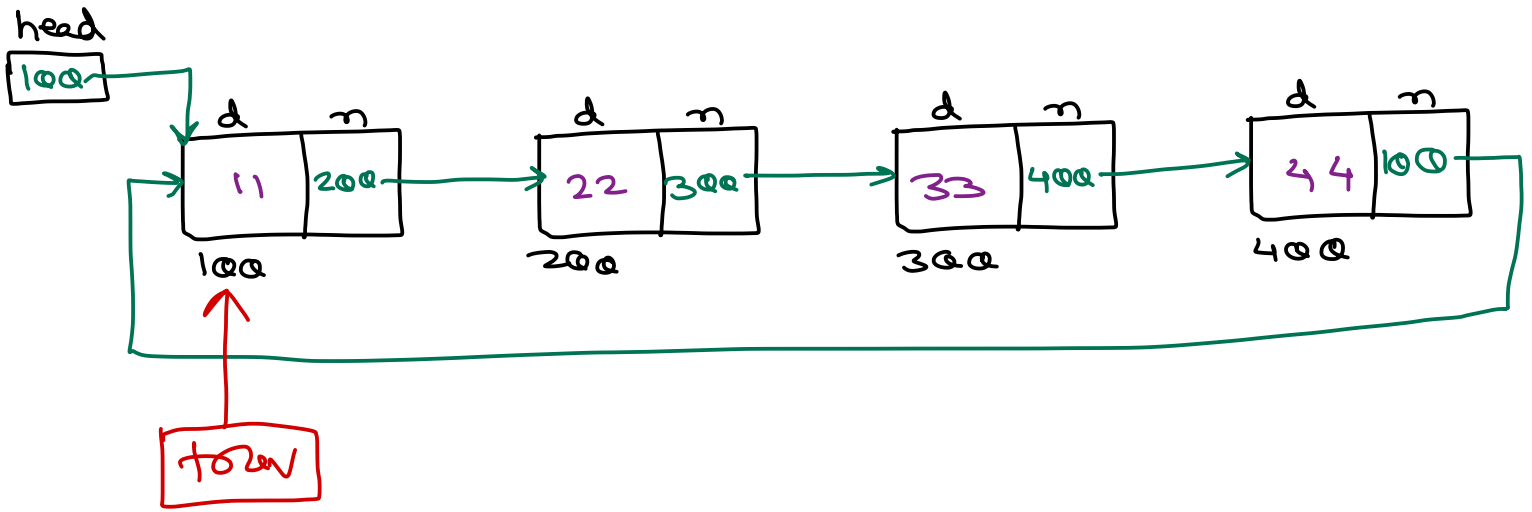
Stack / Queue using Linked List

- Stack can be implemented using linked list.
  - add first
  - delete first
  - is empty

- Queue can be implemented using linked list.
  - add ~~last~~ first
  - delete ~~first~~ last
  - is empty



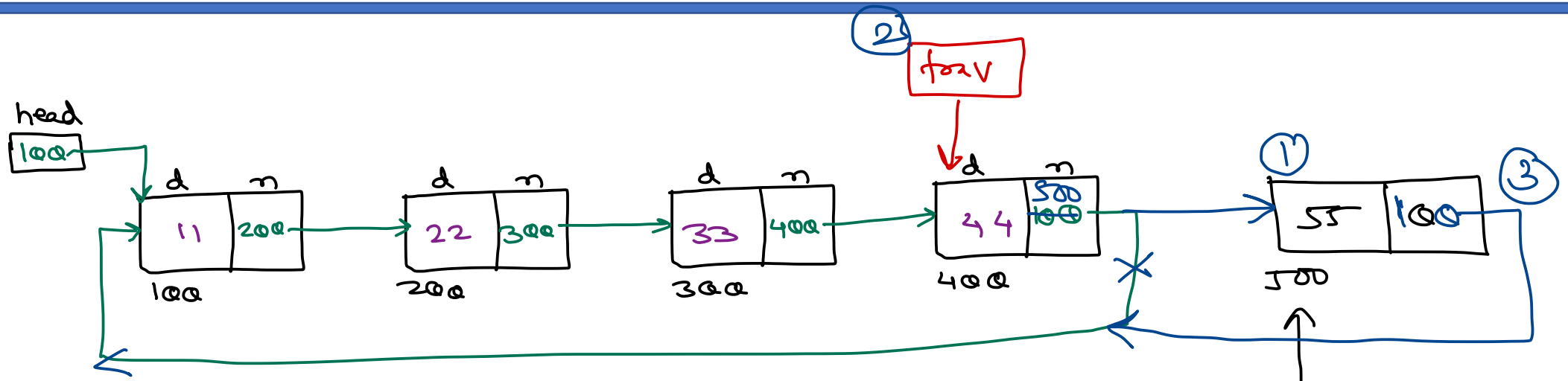
Singly Circular Linked List – last node contain addr of first node.




```
if(head != null) {
    trav = head;
    do
    {
        pf(trav.data);
        trav = trav.next;
    } while(trav != head);
}
```

11 22 33 44

Singly Circular Linked List – add last



① Create & init mm.

② traverse till last node.

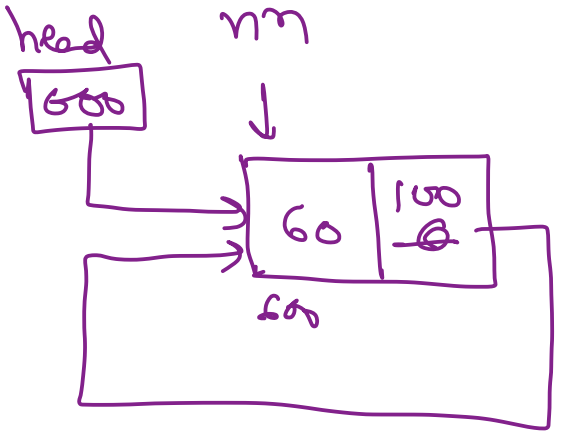
③ mm.next = head;

④ trav.next = mm;

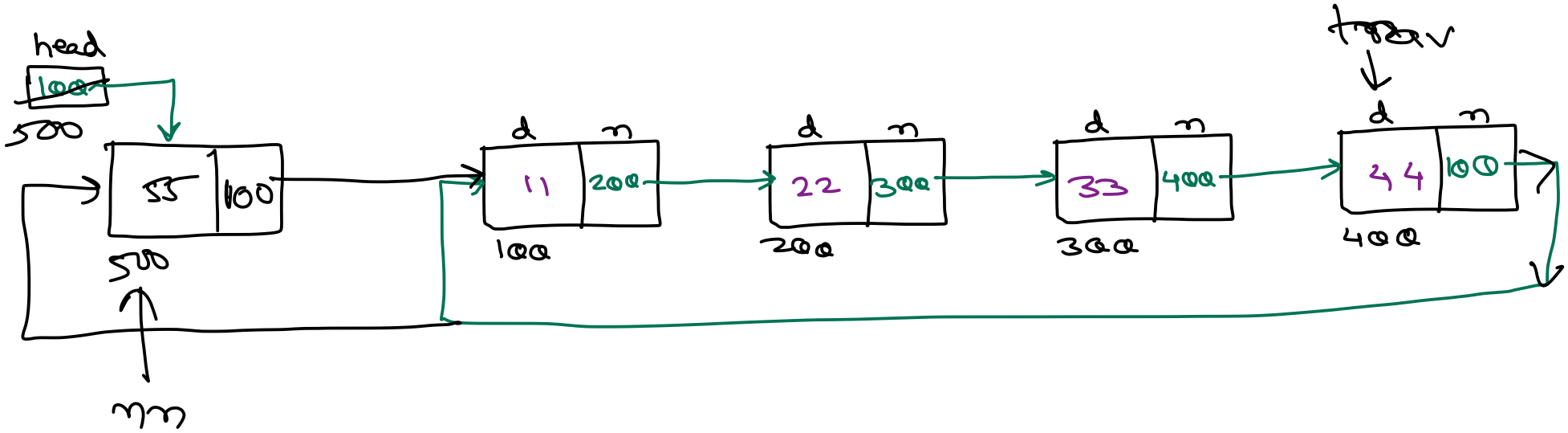
Special 1: list empty

head = mm;

mm.next = head;



Singly Circular Linked List



① create & init nn.

② traverse till last node.

③ nn.next = head;

④ trav.next = nn;

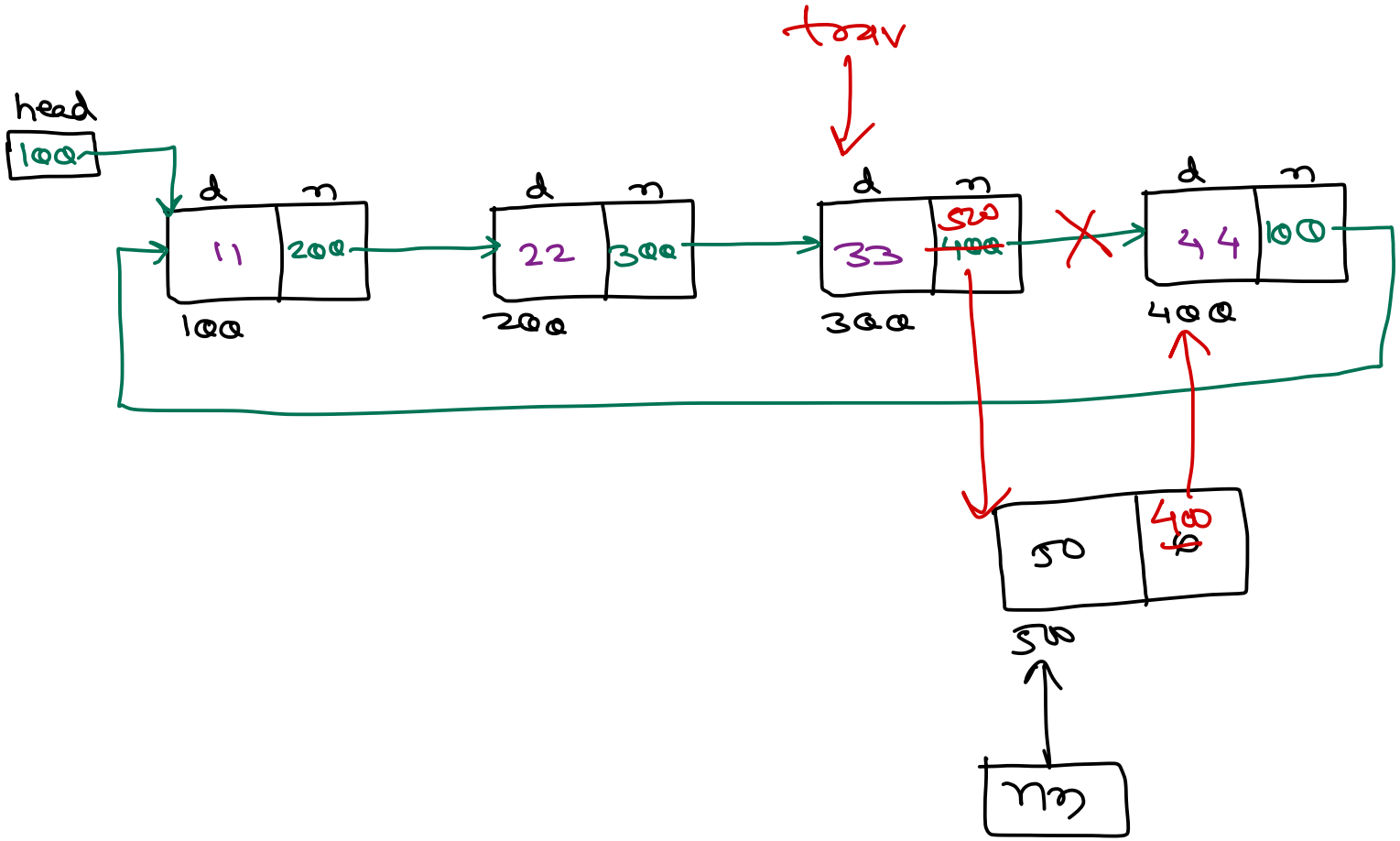
⑤ head = nn;

Specia l:. list empty

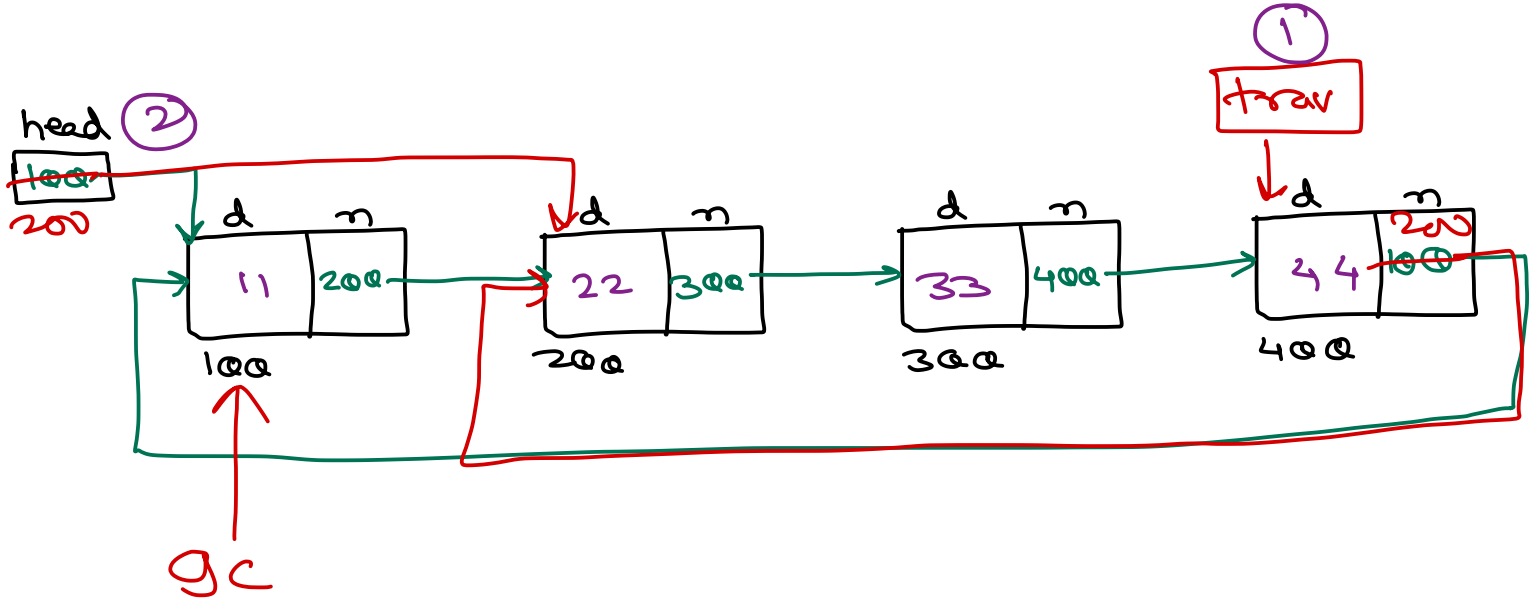
head = nn;

nn.next = head;

Singly Circular Linked List - insert at pos



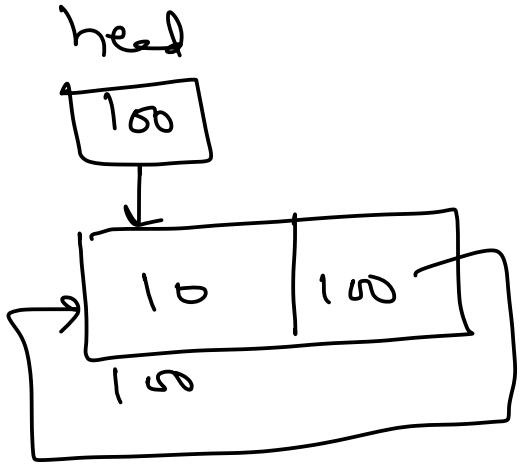
Singly Circular Linked List - delete first



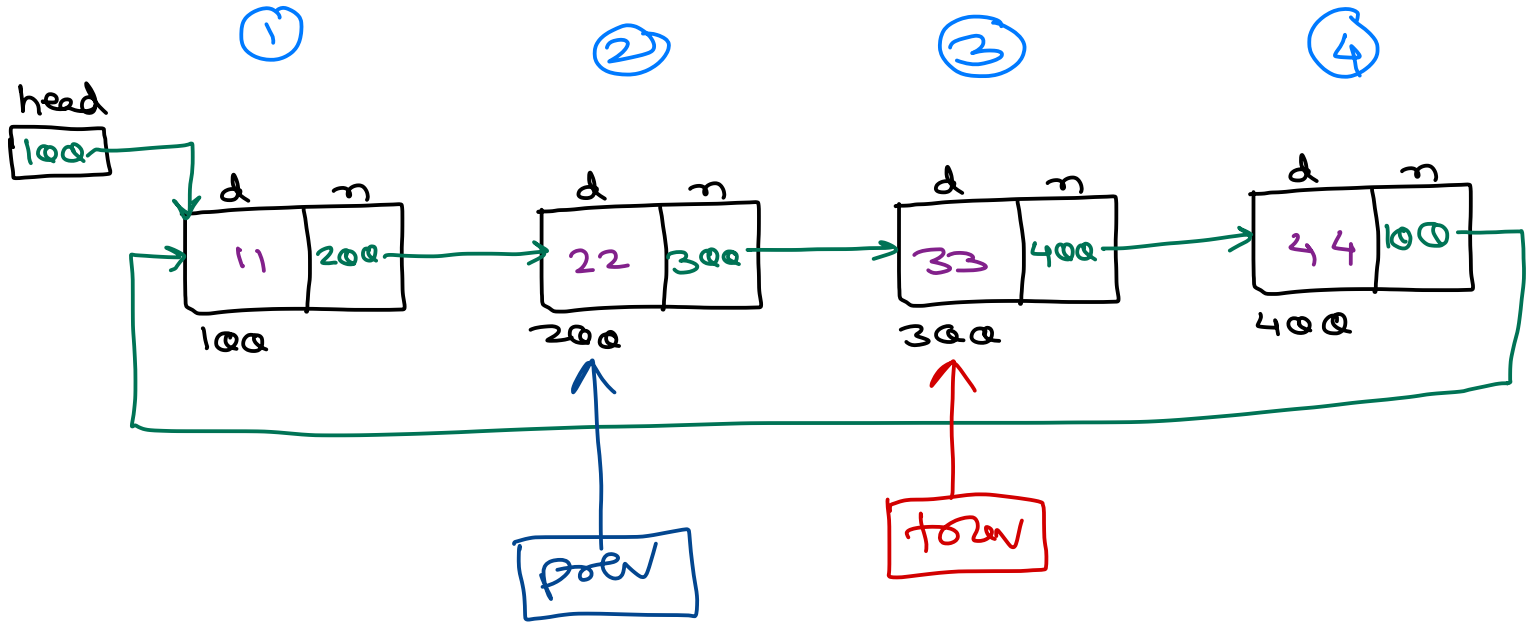
special 1: list empty
↳ do nothing.

special 2: single node
↳ head = null
↓
node will be gc.

1. trav till last node
2. take head to next node
3. trav next to new head.
4. old first node will be g.c.



Singly Circular Linked List - del at pos.



① trav till pos &
also maintain prev pointer.

② prev.next = trav.next;

③ trav node will be gc.

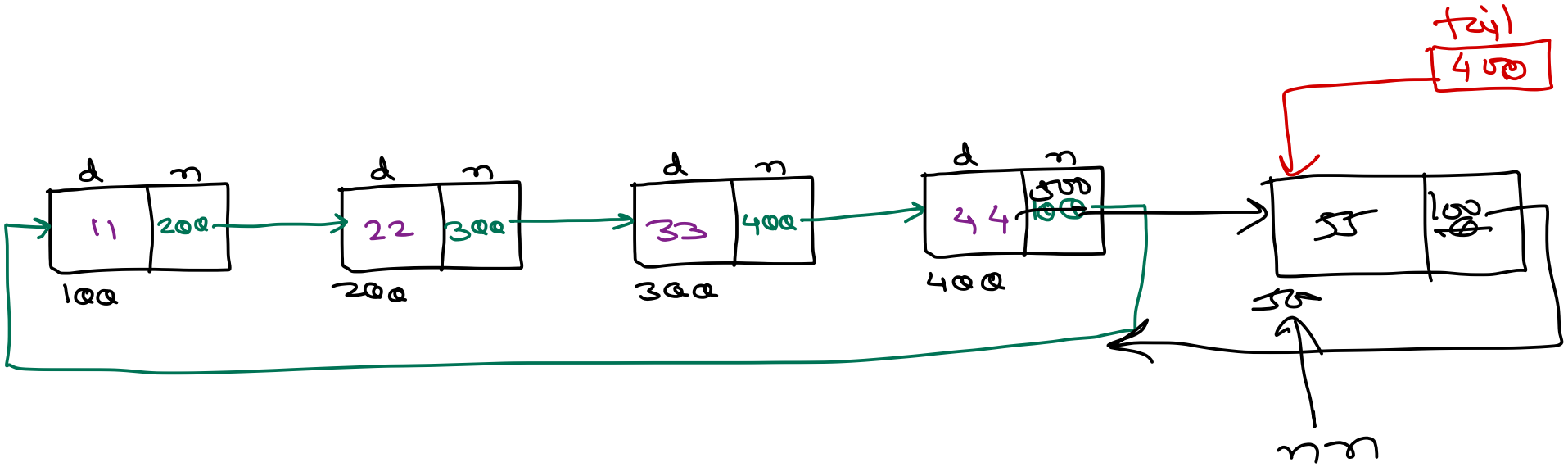
special 1: list empty
✓ do nothing.

special 2: if pos == 1,
✓ del first.

special 3: if pos < 1,
✓ do nothing

special 4: if pos > max,
✓ do nothing

Singly Circular Linked List → using tail



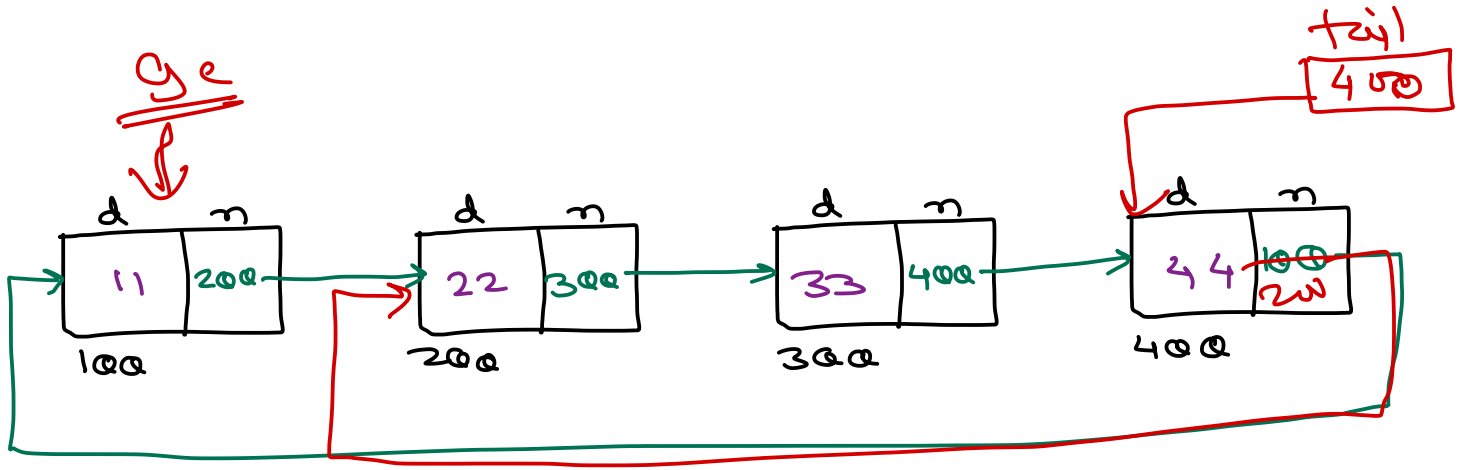
add last → O(1)

```
nn.next = tail.next;
tail.next = nn;
tail = nn;
```

add first → O(1)

```
nn.next = tail.next;
tail.next = nn;
tail = nn;
```

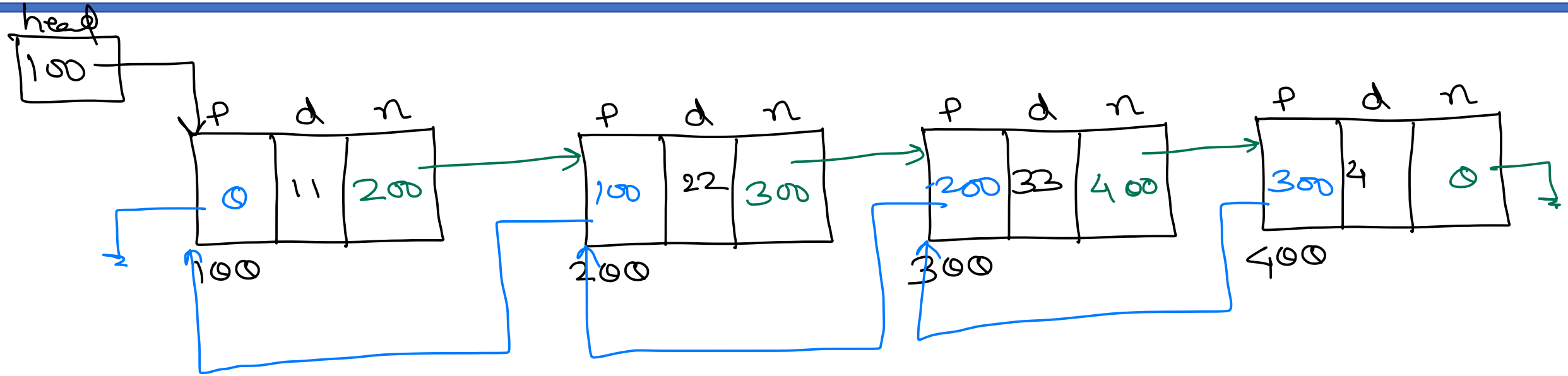
Singly Circular Linked List – using tail



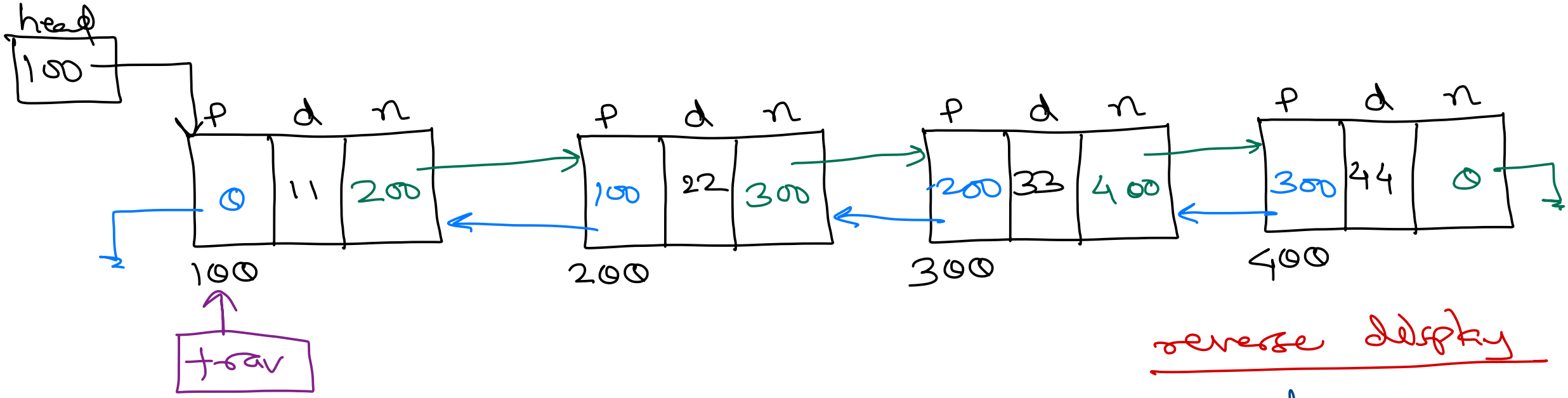
del-first – O(1)

```
tail.next = tail.next.next;
```

Doubly Linear Linked List



Doubly Linear Linked List → display



forward display

```
trav = head;
while (trav != null) {
    Pf (trav.data);
    trav = trav.next;
}
```

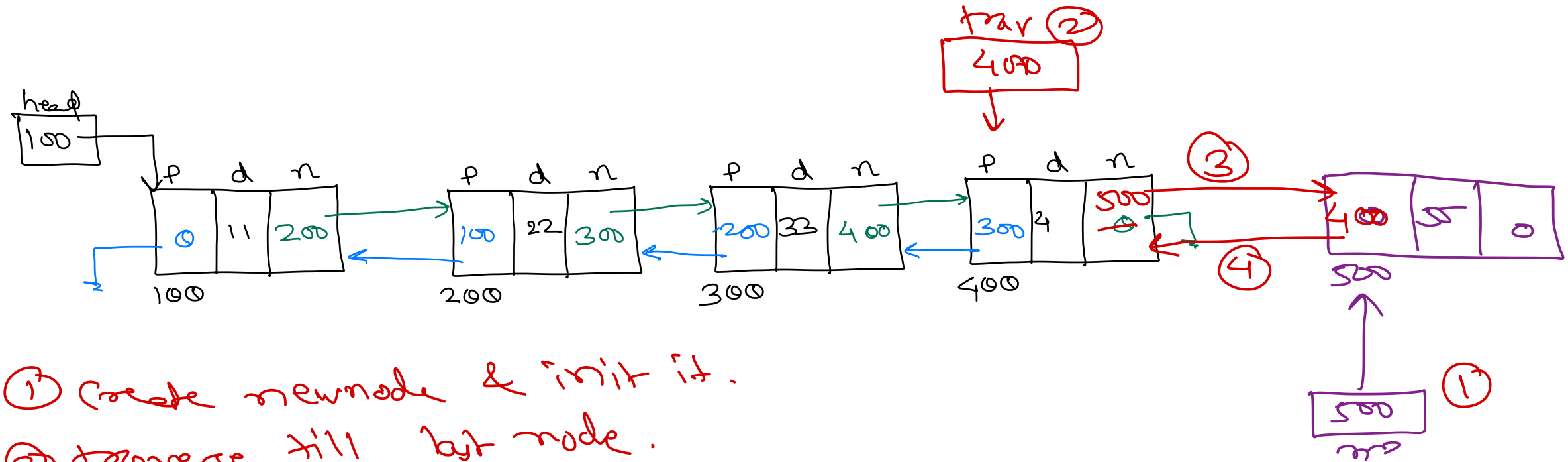
T(n)
O(n)

T(2n)
O(n)

reverse display

```
trav = head;
while (trav.next != null)
    trav = trav.next;
while (trav != null) {
    Pf (trav.data);
    trav = trav.prev;
}
```

Doubly Linear Linked List → add last



1. Create newnode & init it.
2. Traverse till last node.
3. trav's next = nn;
4. nn's prev = trav;

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>