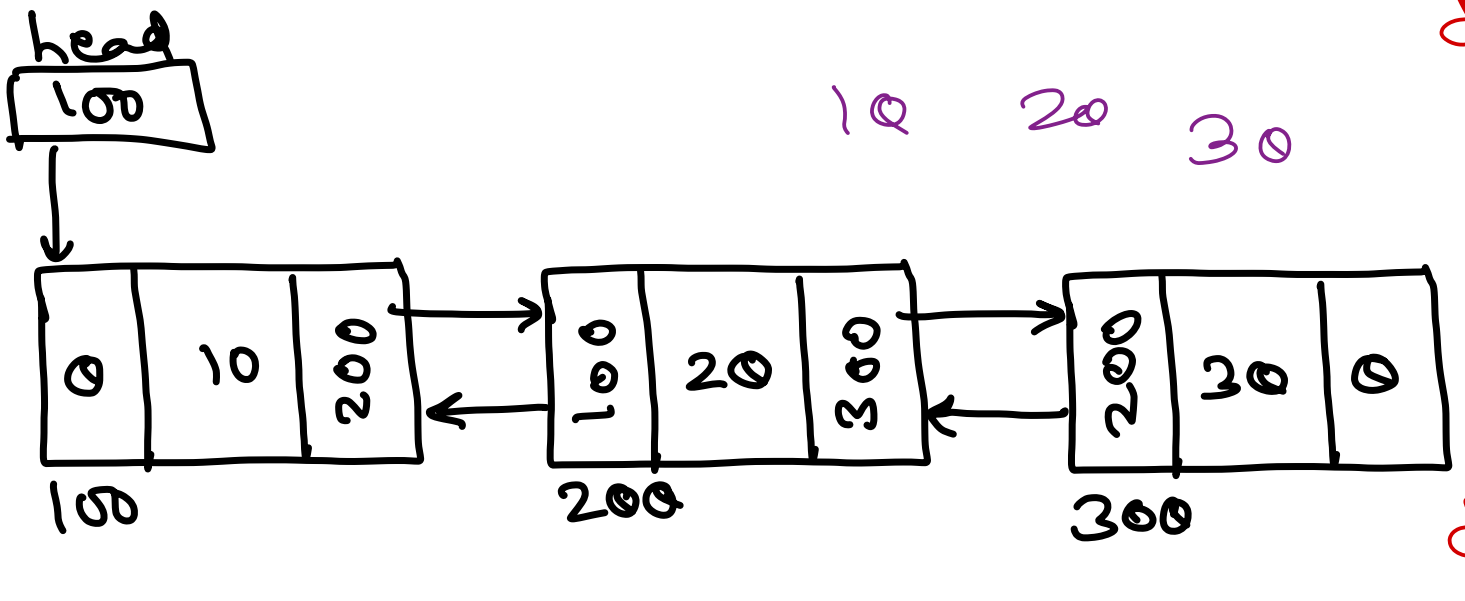
Data Structure & Algorithms

Sunbeam Infotech

doubly linear linked list → display()

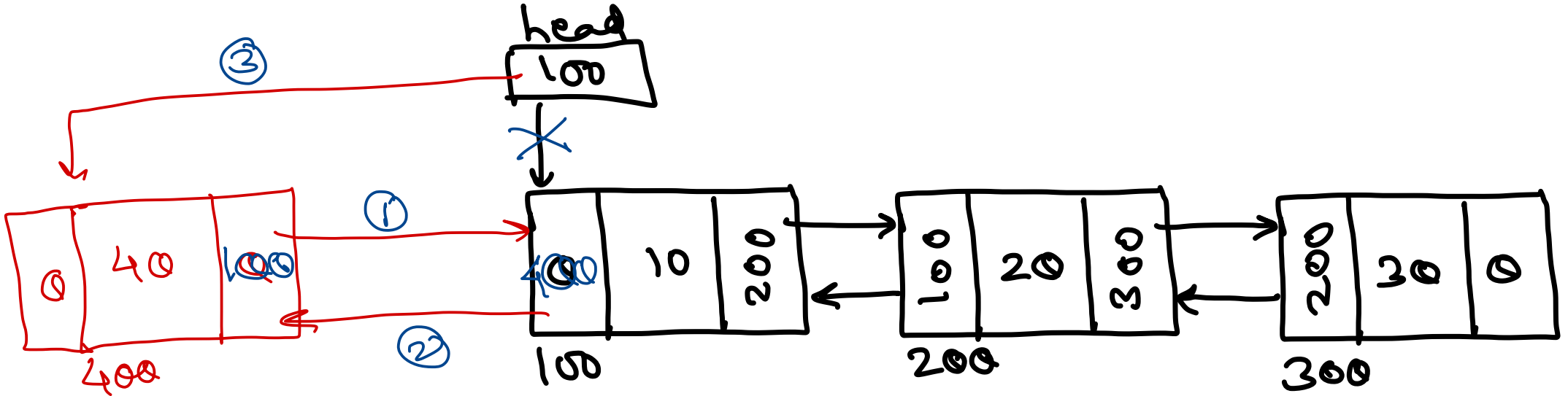



```
class node {
    int data;
    node *next;
    node *prev;
};
```

```
display_fwd():
    node *trav = head;
    while(trav != NULL){
        cout << trav->data;
        trav = trav->next;
    }
```

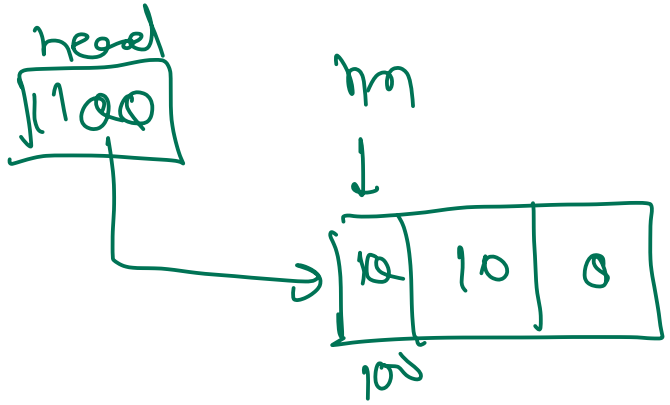
```
display_rev():
    node *trav;
    if(head != NULL)
    {   trav = head;
        while(trav->next != NULL)
            trav = trav->next;
    }
    while(trav != NULL)
    {   cout << trav->data;
        trav = trav->prev;
    }
```

doubly linear list – add first.



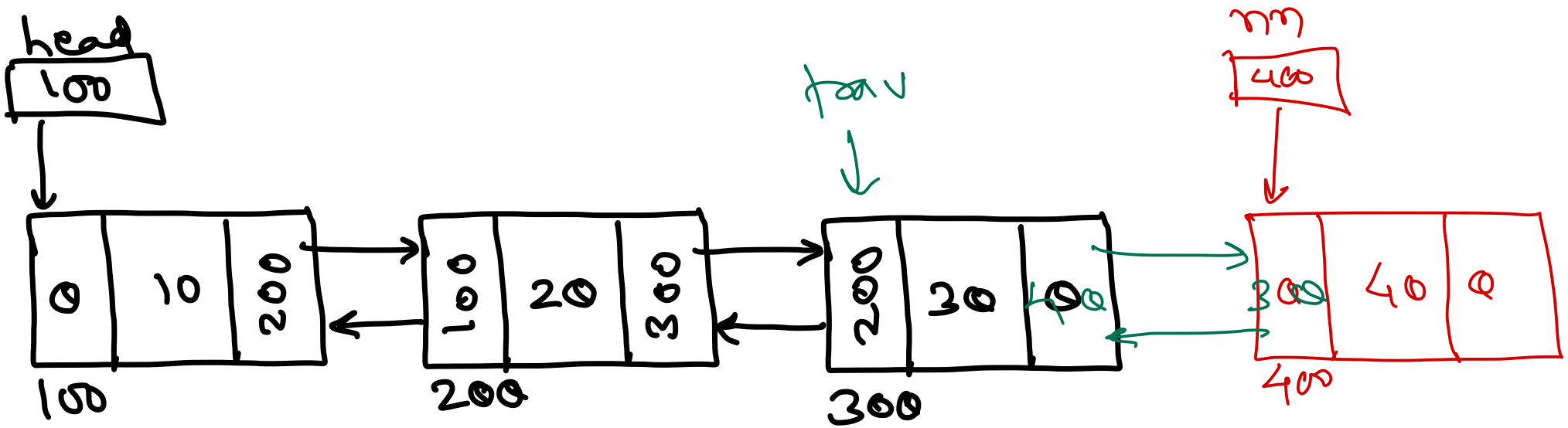
nn = new node(val);

Special 1: list is empty.




```
if(head == NULL)
    head = nn;
else
{
    1) nn->next = head;
    2) head->prev = nn;
    3) head = nn;
}
```

doubly linear list → add last



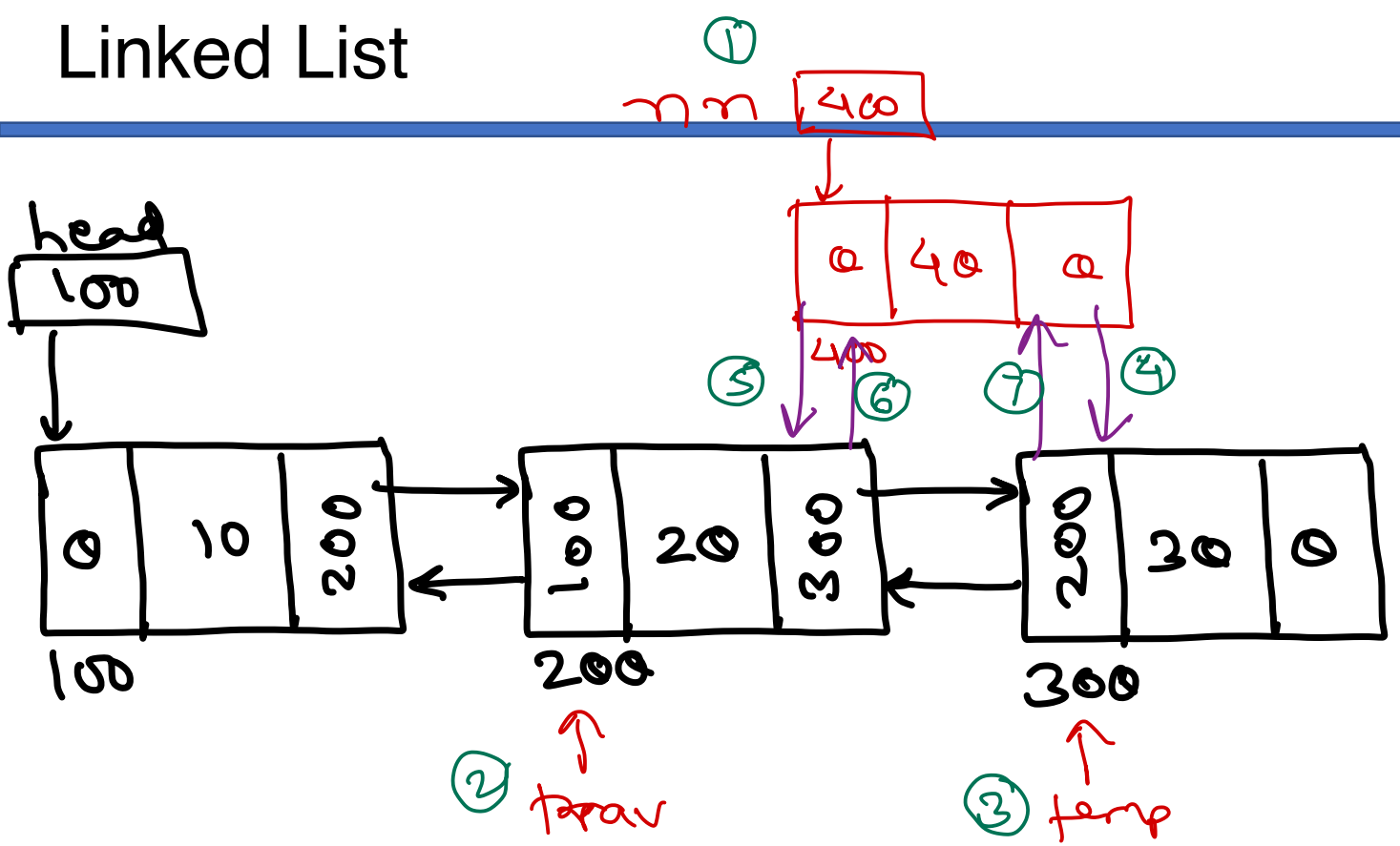
① alloc & init node
② traverse till last node. (trav)
③ nn → prev = trav;
④ trav → next = nn;

special: list empty, node is last node.

```
nn = new node(val);
if (head == null)
    head = nn;
else {
    trav = head;
    while (trav->next != NULL)
        trav = trav->next;
    nn->prev = trav;
    trav->next = nn;
}
```

Linked List

doubly list: add at pos



1. alloc & init node.
2. traverse till pos -1 (trav).
3. take addr of next node (temp).
4. nn -> next = temp;
5. nn -> prev = trav;
6. trav -> next = nn;

if(temp != NULL)

7. temp -> prev = nn;

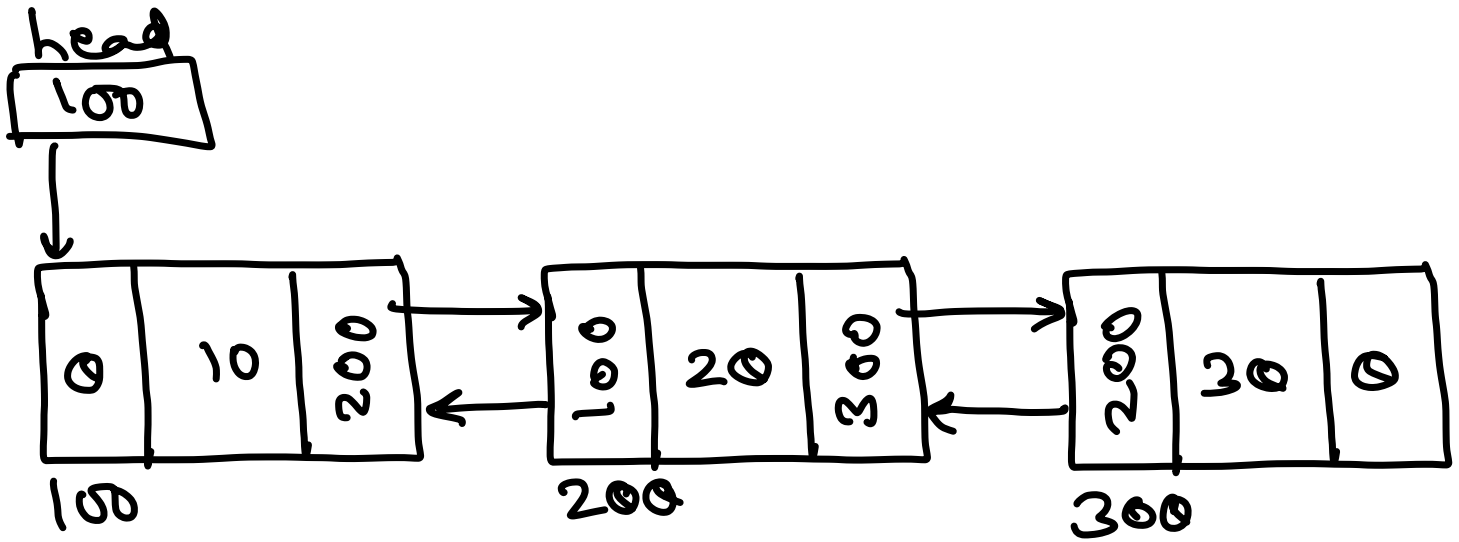
S1) if list empty & pos <= 1
add_first()

S2) pos is beyond size of list,
if (trav -> next == NULL)
break;

S3) pos is immediately after the size of list

check temp before setting its prev,

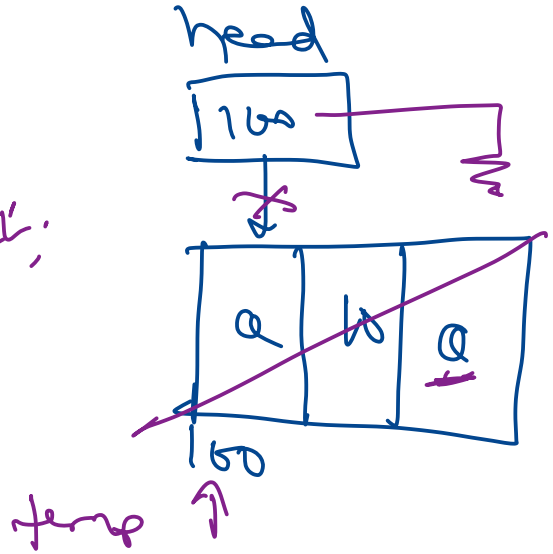
doubly linear list : del first.



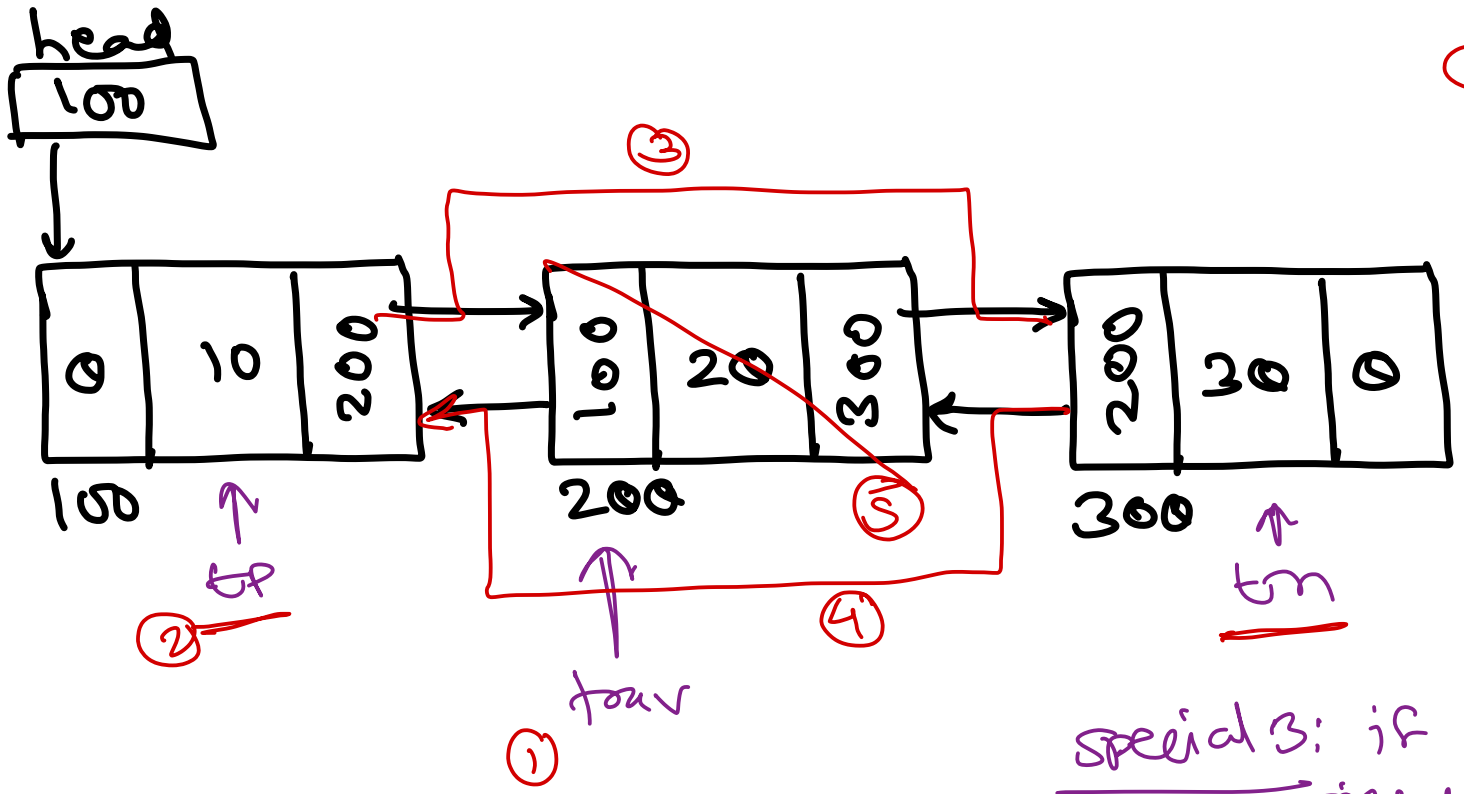
① take addr of first node into temp.
② take head to next node.
③ delete temp node.
④ new head's prev to NULL.

special 1:
if list is empty,
return;

special 2:
if (head != NULL)
head->prev = NULL;



doubly linear ll → del at pos-



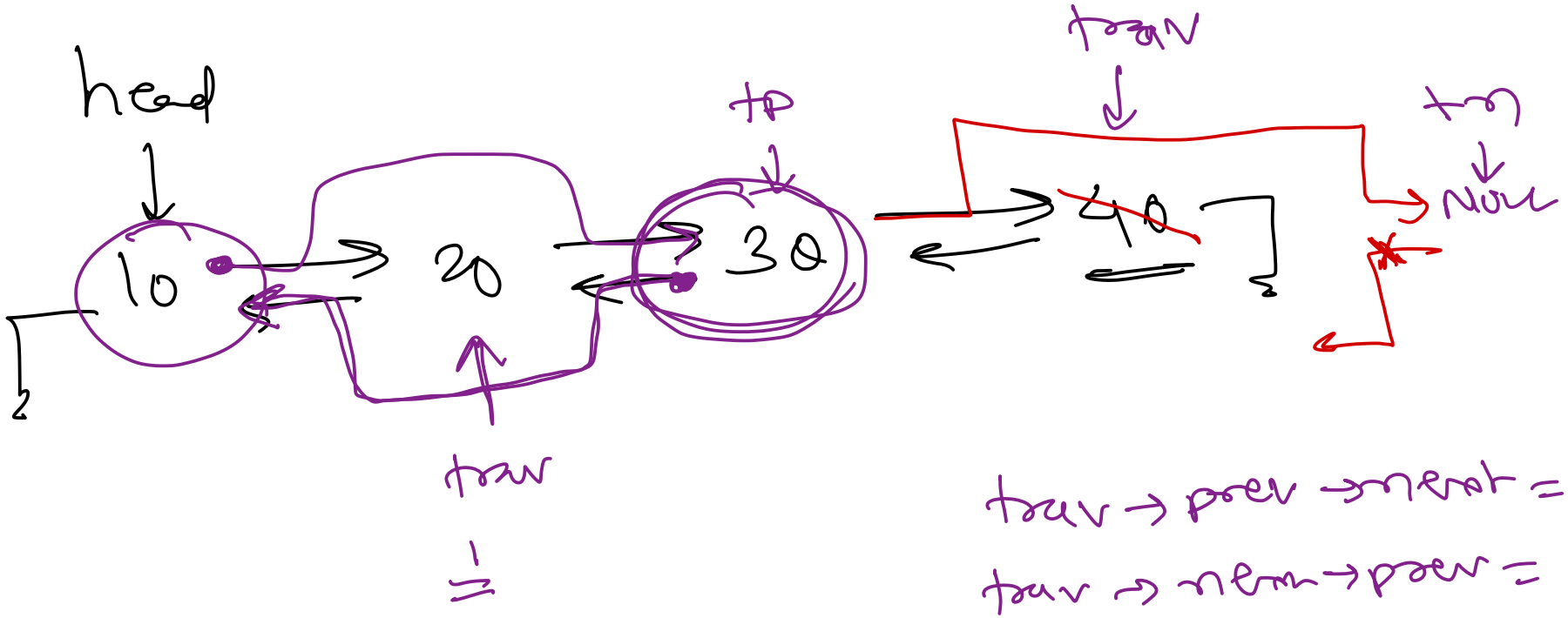
① trav till pos.
② take ptr to prev (tp) & ptr to next (tmp).
③ tp's next to tmp.
④ tmp's prev to tp
⑤ delete trav.

Special 1:
if list empty, return

Special 2:
if pos <= 1
return;

Special 3: if pos beyond size;
if (trav == NULL)
return;

Linked List




```
trav->prev->next = trav->next;
trav->next->prev = trav->prev;
delete trav;
```

doubly list → maintain count. [assign]

```
class doubly_list {
private:
    node *head;
    int cnt;
public:
    ① in ctor,
        cnt = 0;
    ② in each add,
        cnt++;
    ③ if each del,
        cnt--;
};
```

Advantages :

① list empty cond : cnt == 0.

② check last pos in add at pos

```
if (pos >= cnt)
    add_last(val);
```

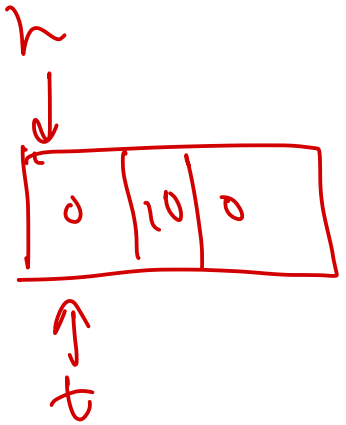
③ check last pos in del at pos.

```
if (pos > cnt)
    return;
```

Linked List

doubly linear ll : tail pointer. [assign].

```
class doubly_llist {
  private:
    node *head;
    node *tail;
  public:
    ① ctor :
        head = NULL;
        tail = NULL;
};
```




```
if(head==tail)
{ delete head;
  head=null;
  tail = null;
}
```

③ rev display will be fast.

Advantage :

① add last ; no need to trav filled
↳ O(1).

② del last : no need to trav till end.
↳ O(1)

```
void del_last() {
  if(tail != null) {
    temp = tail;
    tail = tail->prev;
    delete temp;
    if (tail != null)
      tail->next = NULL;
  }
}
```

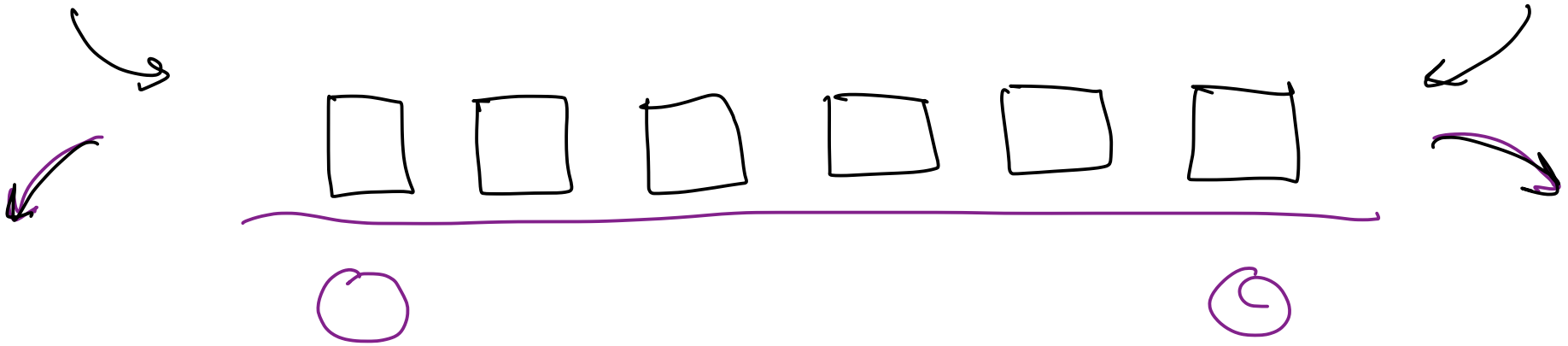
Double Ended Queue using Linked List (assign)

In deque, push & pop can be done from both ends.
Ideal time complexity should be O(1);

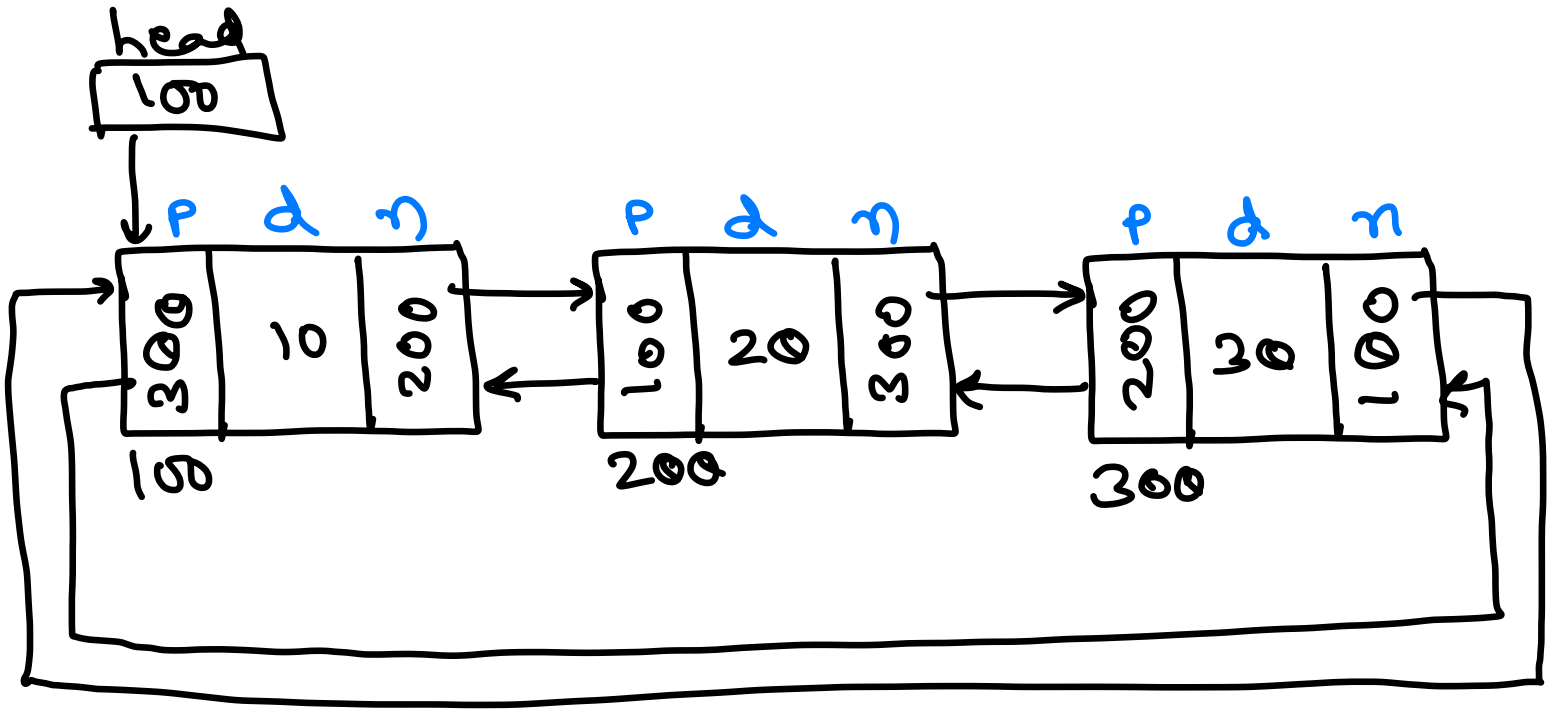
```
class doubly_list {
private:
    node * head;
    node * tail;
public:
    ctor()
    add_first()
    add_last()
    del_first()
    del_last()
    add_at_pos()
    del_at_pos()
    del_all()
    get_first()
    get_last()
    isempty()
};
```

```
class deque {
private:
    doubly_list dl;
public:
    push_rear()
    push_front()
    pop_rear()
    pop_front()
    isempty()
    peek_front()
    peek_rear()
};
```

Linked List



doubly circular linked list



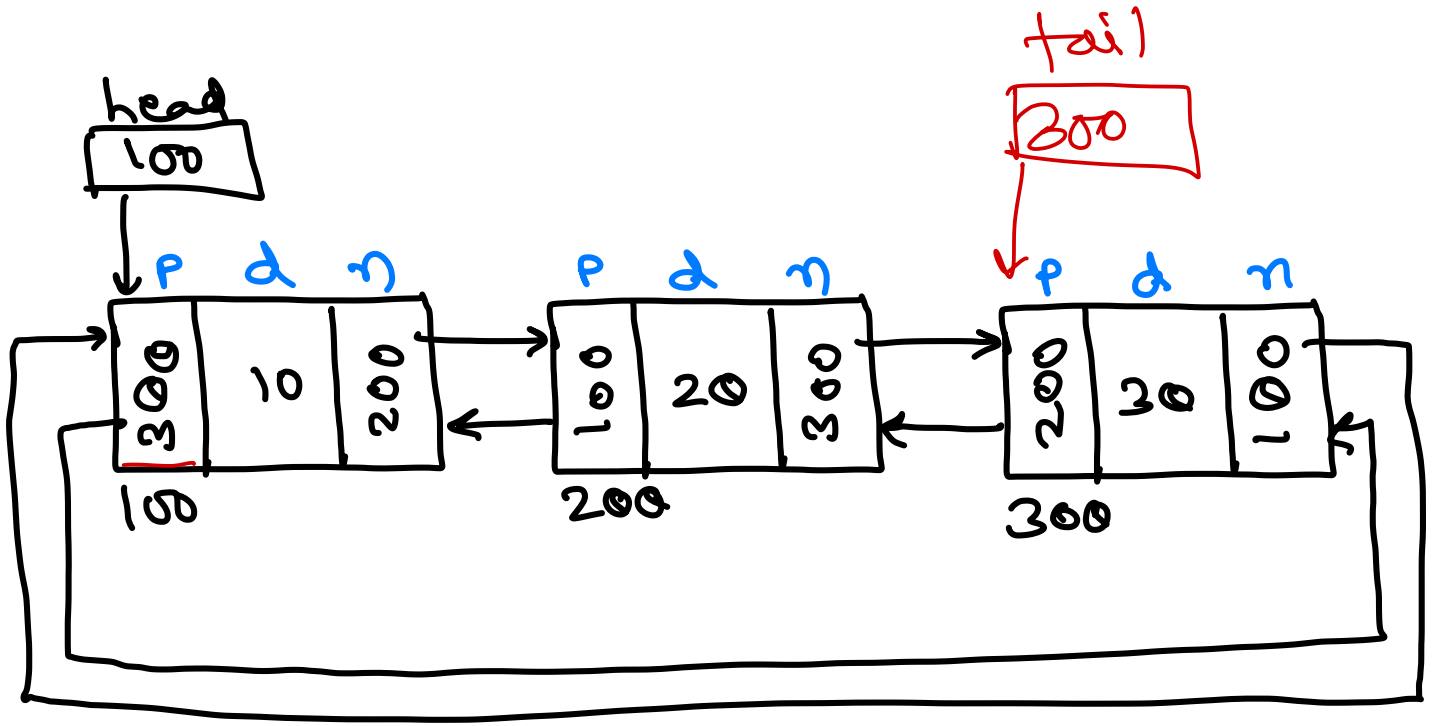
temp = head → prev;

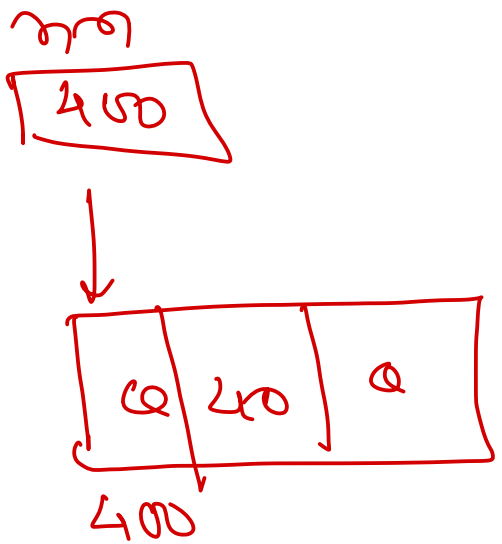
↓

Jump to last node.

↳ O(1).

doubly circular linked list - add last, assign





tail = head->prev;

display()
add_first()
add_last()
del_first()
del_last()

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>