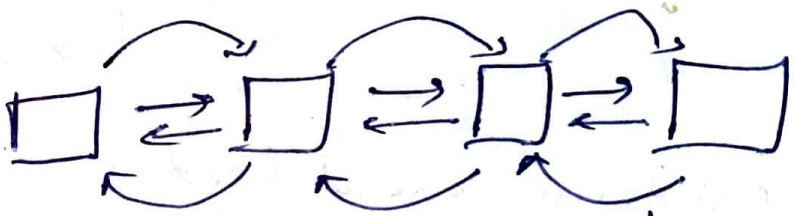
Doubly Linked List

- What is DLL
- Representation of DLL in C++ / Java
- Array → DLL
- Deletion of a Node
- Insertion of a Node



move in both directions

SinglеLL Code

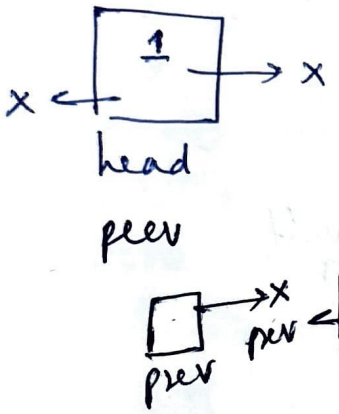
```
struct Node {
    int data;
    Node* next;
    Node(int val, Node* next1 = nullptr) {
        data = val;
        next = next1;
    }
};
```

Doubly LL Code

```
struct Node {
    int data
    Node* next;
    Node* back;
    Node(int val, Node* next1 =
        nullptr, Node* back1
            = nullptr) {
        data = val;
        next = next1;
        back = back1;
    }
};
```

Convert array to DLL

arr[] = | 1 | 3 | 2 | 4 |

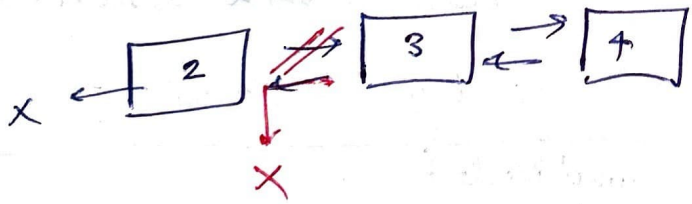



```
Node* head = new Node(arr[0]);
Node* prev = head
for (i = 1 → n-1) {
    Node * temp = new Node(arr[i], nullptr, prev);
    prev->next = temp;
    prev = prev->next; // prev = temp;
}
return head;
```

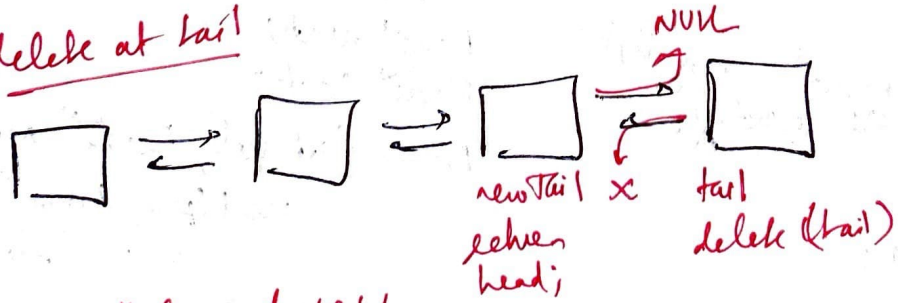
Deletion of a Node

- → at head
- → at tail
- → at kth position
- → at a particular Node.

(a) delete node at head



(b) delete at tail



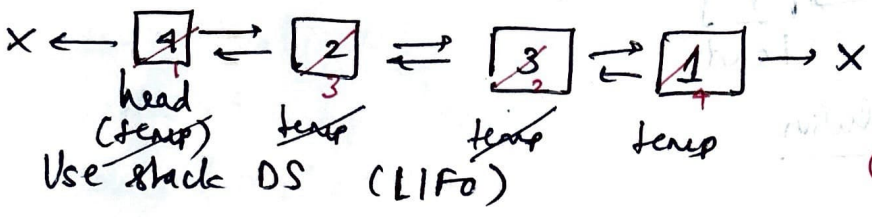
(c) delete kth element of DLL

(d) delete a node of DLL

Insertion of a Node

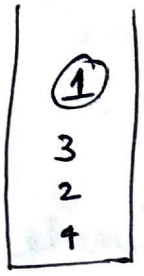
- ↳ before Head
- ↳ before Tail
- ↳ before a kth node
- ↳ before a particular node value.

Reverse a DLL



Use stack DS (LIFO)

(I) Method 1 — Swapping nodes using stack



1) In first traversal, push the nodes into stack
2) Again, in second traversal, now take the top element of the stack → pop → replace in the node.

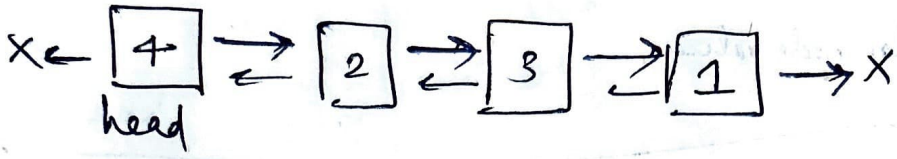
```
Node* temp = head;        stack<int>
while (temp != NULL) {
    st.push(temp->data);          step 1
    temp = temp->next;            O(N)
}

temp = head                       step 2  O(N)
while (temp != NULL) {
    temp->data = st.top(); // taken the top element & stored it in temp->data
    st.pop() -> popped that element from the stack
    temp = temp->next;
}
```

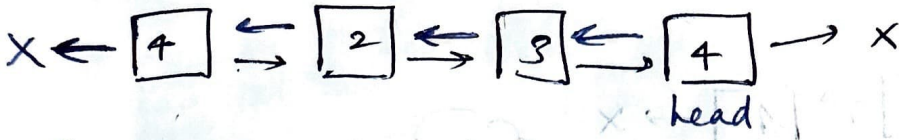
TC : O(2N)

SC : O(N) → N sized external stack is used.

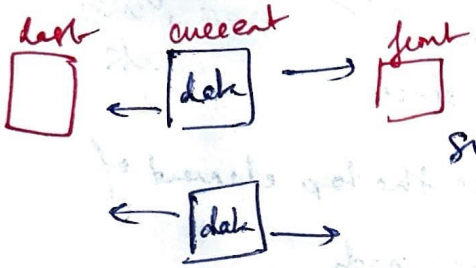
Method 2

Try in single traversal - to reverse the LL i.e not O(N) TC but O(N) TC





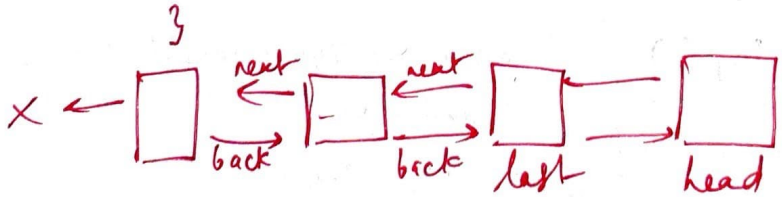
(Reverse the links) → Solution



Swap → temp = a;
a = b;
b = temp;

current → next = last;
current → prev = front;

last = NULL, current = head;
while (current != NULL) {
last = current → back;
current → back = current → next;
current → next = last;
} SWAPPED
current = current → back } to go to the next.



head = last → back;