

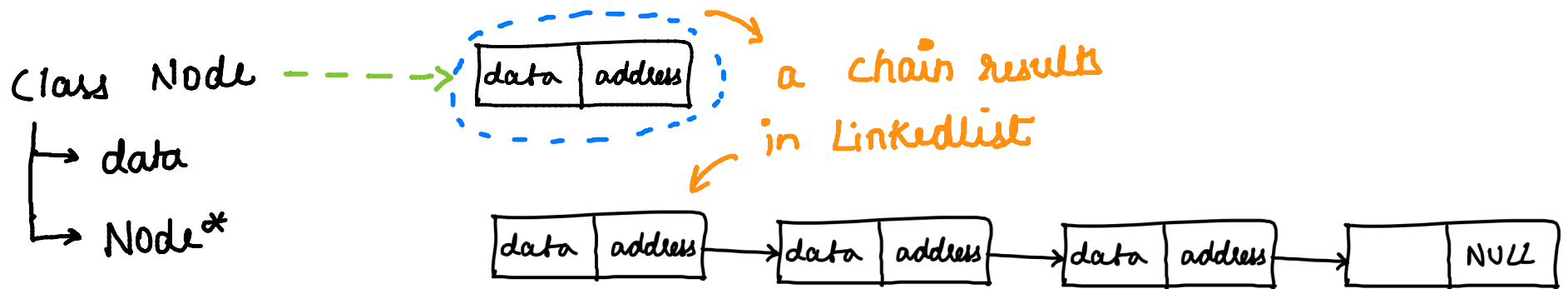
Linked List

Contents →

0. Introduction
1. Reverse a Linked List
2. Middle of Linked List
3. Delete node in a Linked List
4. Merge two sorted Lists
5. Add two numbers
6. Add two numbers II
7. Linked List Cycle
8. Linked List Cycle II
9. Remove Nth node from End of List
10. Palindrome Linked List
11. Remove duplicates from sorted List
12. Swapping nodes in Linked List
13. Odd Even Linked List
14. Swap Nodes in Pairs
15. Copy list with Random Pointer
16. Reverse Nodes in K-group
17. Design Linked List
18. Sort List

Linked List

LinkedList is linear data structure, which consists of a group of nodes in a sequence.



Advantages

1. Dynamic nature
2. Optimal insertion & deletion
3. Stacks and queues can be easily implemented
4. No memory wastage

Real life Applications

1. Previous & next page in browser
2. Image Viewer

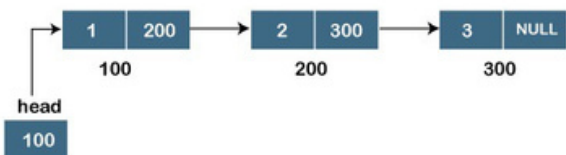
Disadvantages

1. More memory usage due to address pointer.
2. Slow traversal compared to arrays.
3. No reverse traversal in singly linked list
4. No random access.

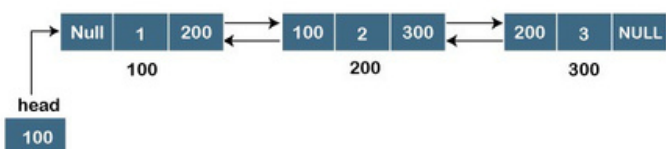
3. Music player

Type

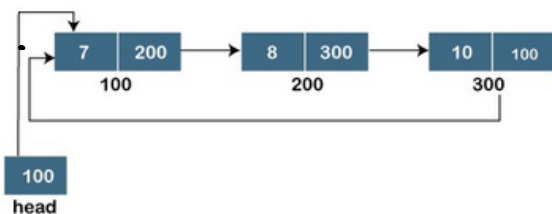
1. Singly linkedlist



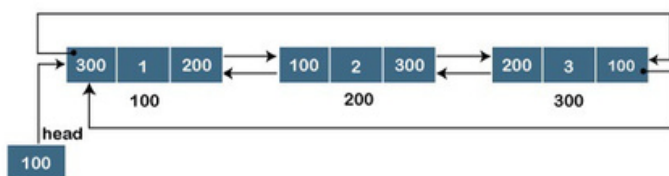
2. Doubly linkedlist



3. Circular linkedlist



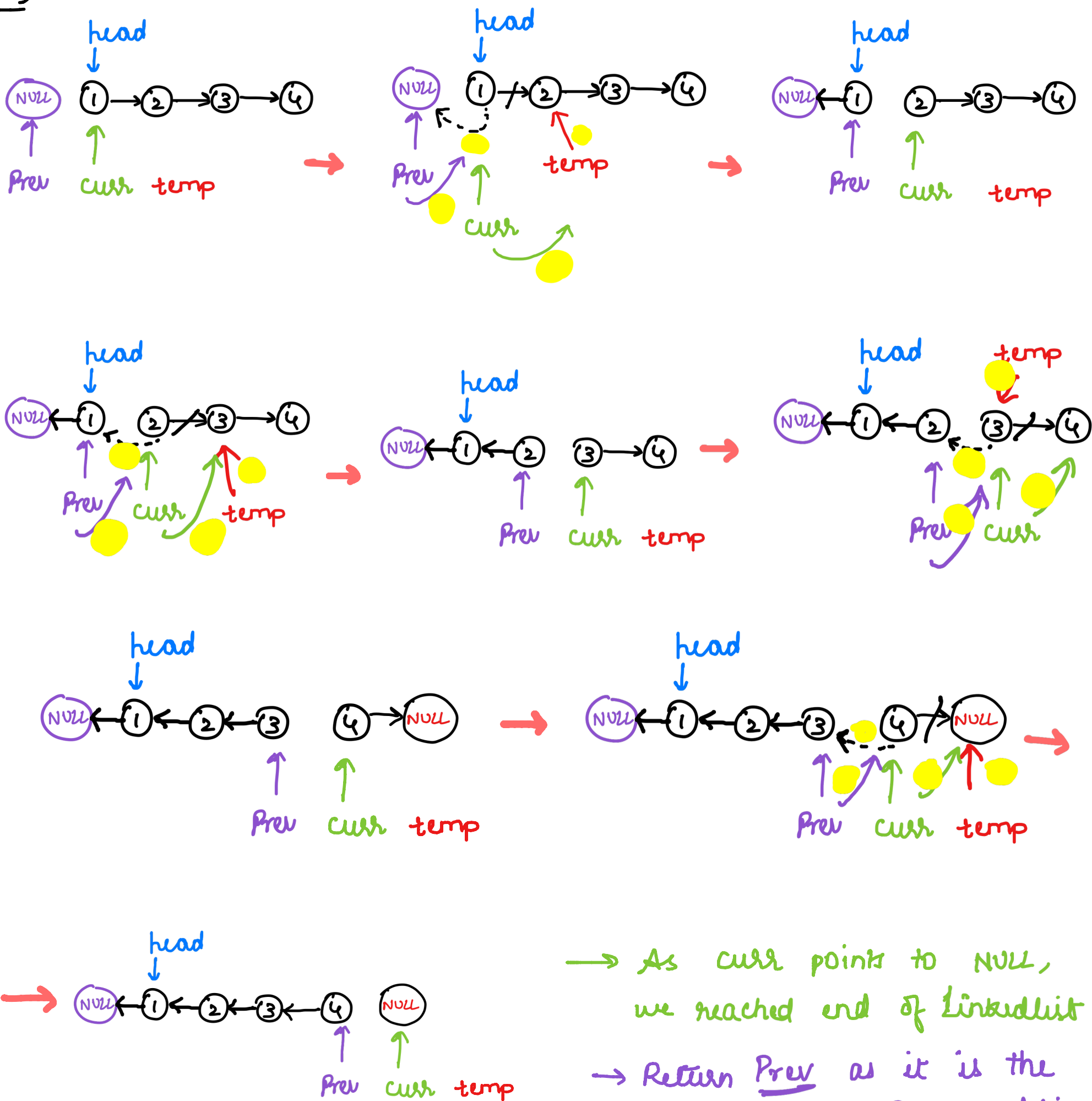
4. Doubly circular linkedlist



① Reverse a linkedlist → Given a linkedlist, return reversed list.

Eg $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \Rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$

Sol)



→ As `curr` points to `NULL`, we reached end of linkedlist
→ Return `prev` as it is the starting pointer of reversed list.