Newton School of Technology

18

Linked List – Part I

by Gladden Rumao

CSA 221 : DSA

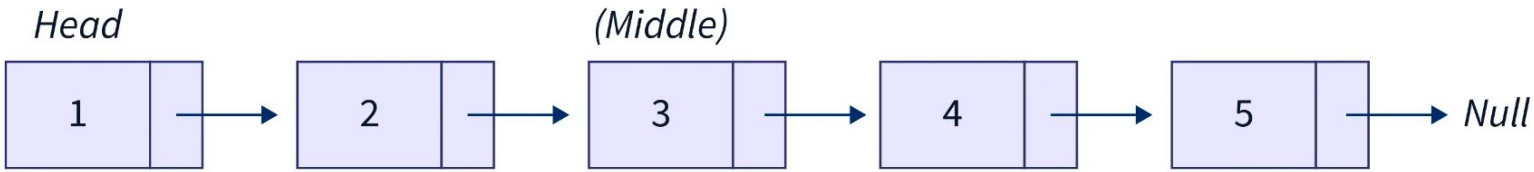
Newton School of Technology

Linked List

Newton School of Technology

What is Linked List ?

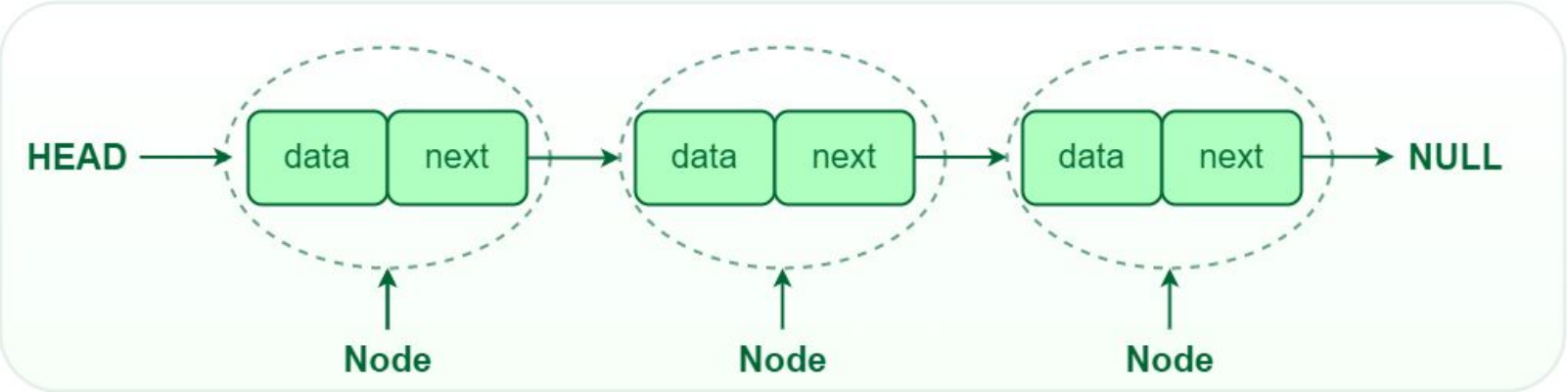
A linked list is a linear data structure where elements, known as nodes, are stored in a non-contiguous manner in memory.



Newton School of Technology

What is Linked List ?

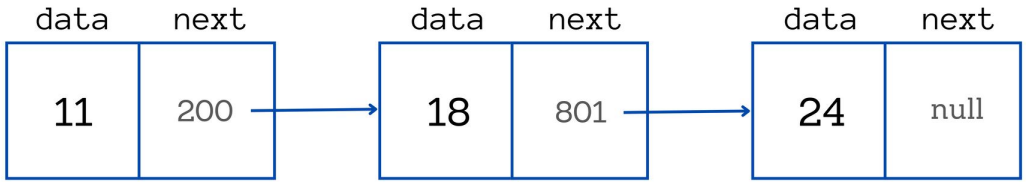
Each node contains the data and a reference (or link) to the next node in the sequence.



Newton School of Technology

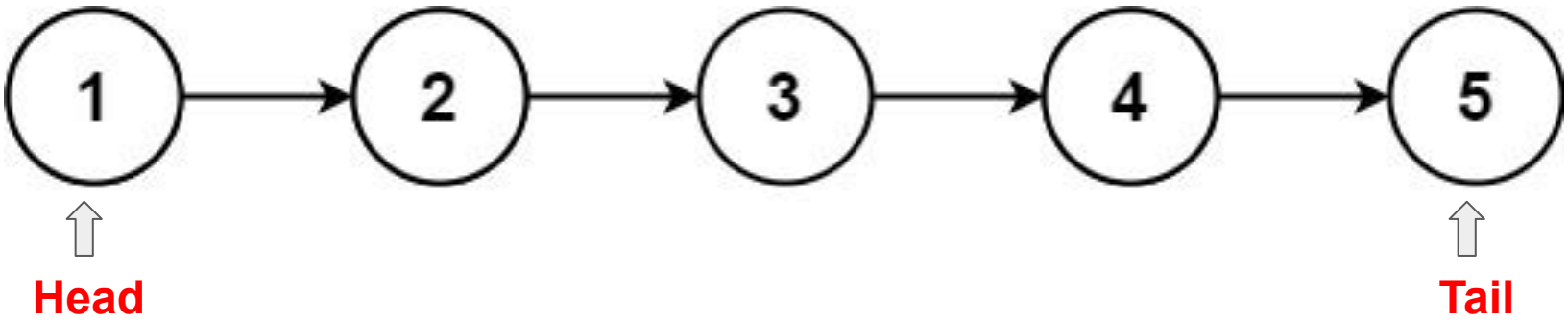
What is Linked List ?

Each node contains the data and a reference (or link) to the next node in the sequence.



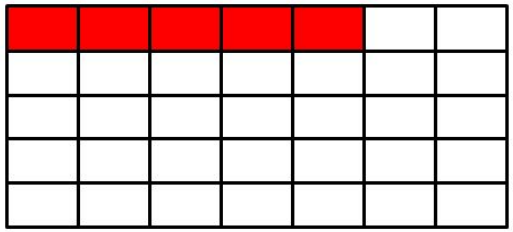
Newton School of Technology

Head and Tail References:



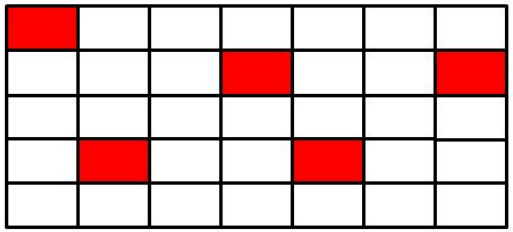
Static Vs Dynamic Memory Allocation :

STATIC



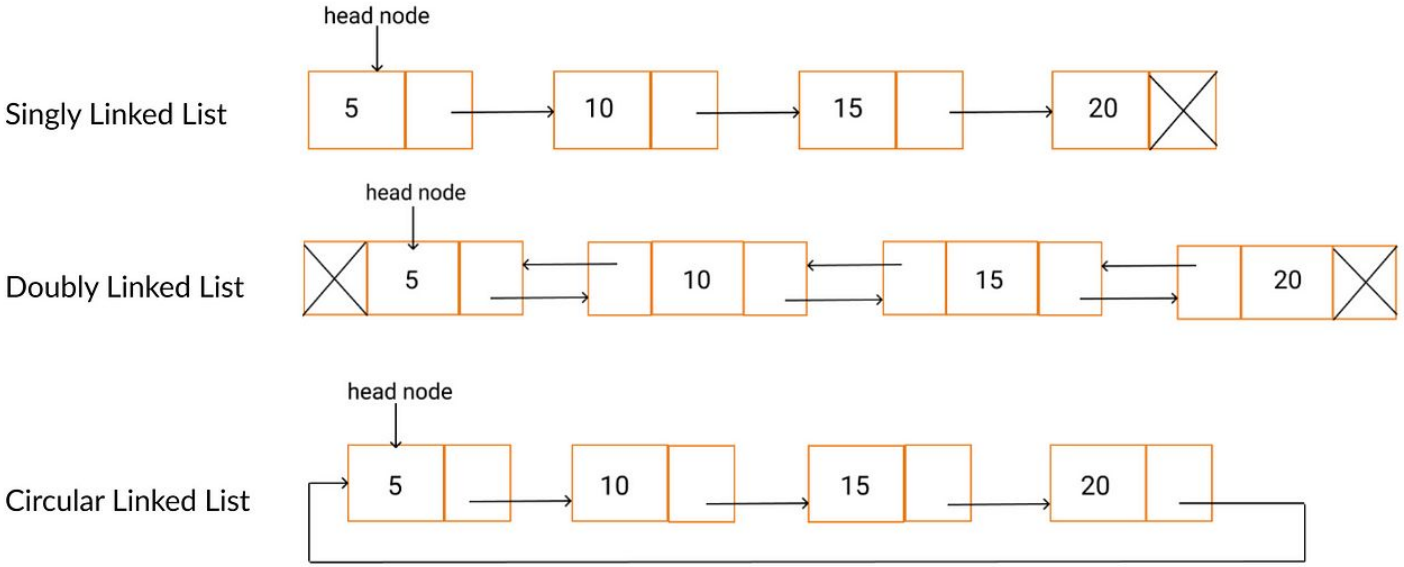
Arrays need a contiguous
Block of memory.

DYNAMIC



Linked Lists don't
need to be contiguous
In memory. They can grow
Dynamically.

Types of Linked List:



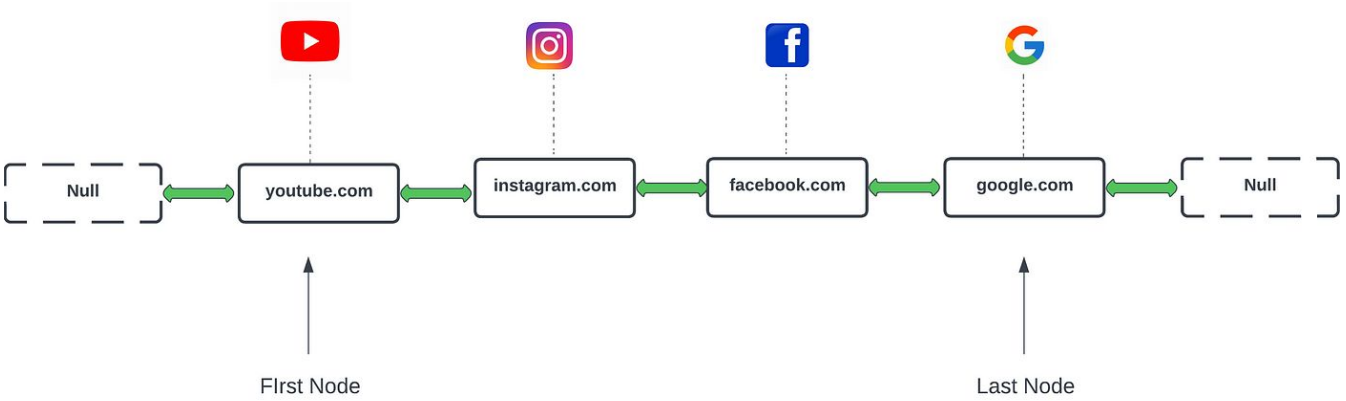
Newton School of Technology

Advantages Of Linked List:

1. **Dynamic Size:** Linked lists can grow or shrink in size dynamically, allowing efficient memory utilization without the need for resizing.
2. **Ease of Insertion/Deletion:** Inserting or deleting elements in a linked list is more efficient because it only involves changing the next pointers of nodes, whereas arrays require shifting elements.
3. **No Memory Wastage:** Arrays can lead to wasted memory space if the allocated array size is larger than the needed capacity. Linked lists allocate memory as and when it is required.

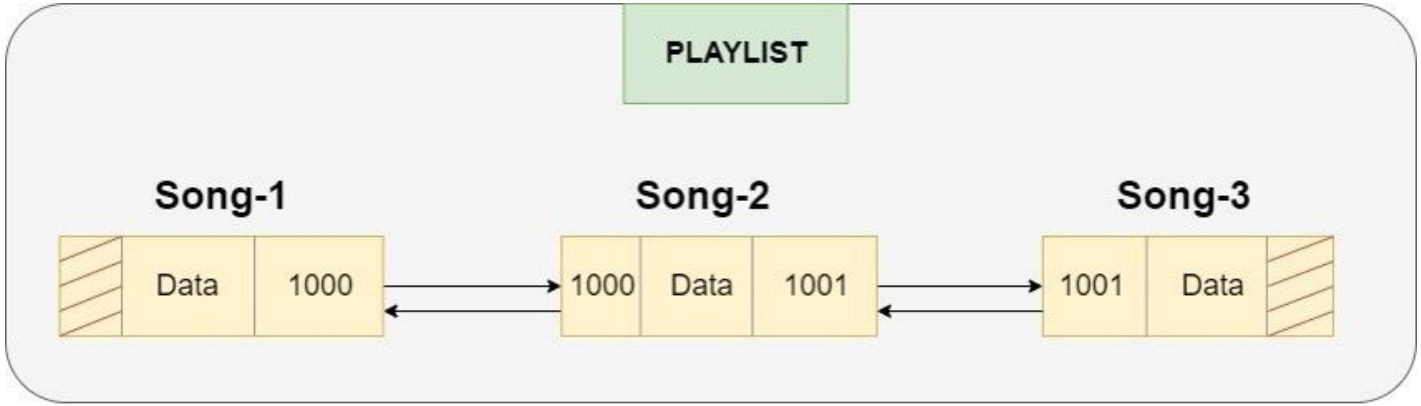
Real World Applications:

- **Browser History:** Modern web browsers use linked lists to keep track of visited URLs. This allows users to easily navigate forward and backward.



Real World Applications:

- **Music Playlists:** Apps like Spotify or Apple Music may use linked lists to manage playlists, where each song is a node with a link to the next song.



Newton School of Technology

Real World Applications:

- **Image Viewer Applications:** Linked lists can be used to implement image viewers with next and previous functionality. Each node represents an image, and navigation through images is facilitated by traversing the list.

Newton School of Technology

Basic Node Structure in Linked List :

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

Newton School of Technology

Basic Operations on Linked List :

1. Inserting a new Node
2. Deleting a Node

Newton School of Technology

Insertion

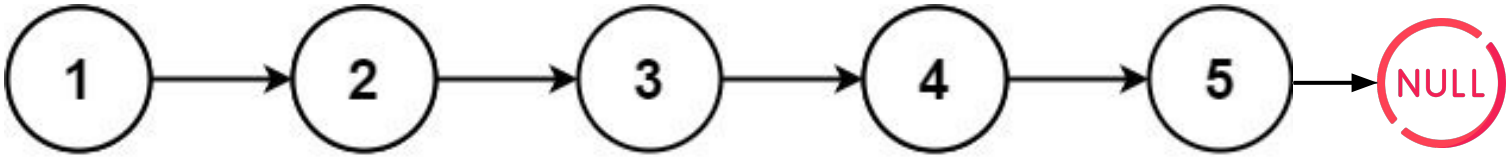
Newton School of Technology

Insert in Linked List :

1. Insert Node at the beginning
2. Insert Node at the end

Newton School of Technology

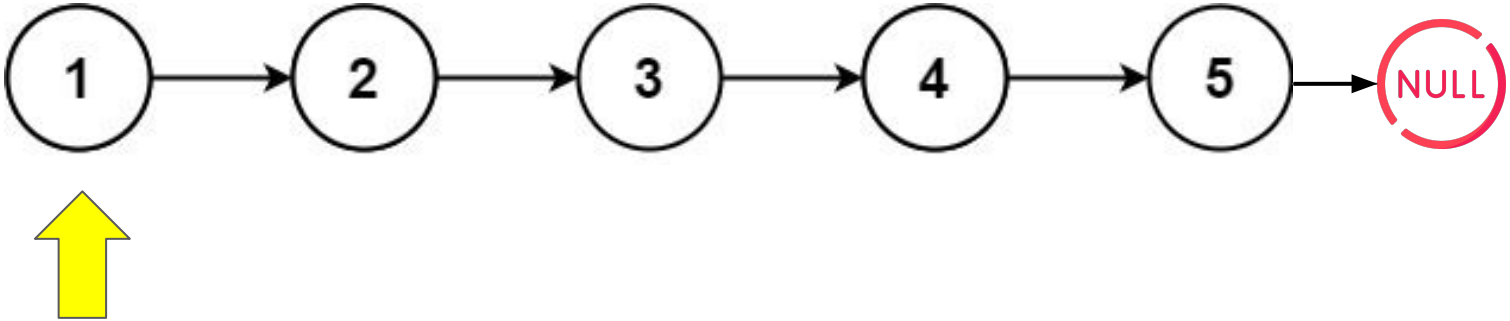
Insert Node at the end:



1. Traverse until the last node is reached.
2. The next pointer of the last node points to NULL.

Newton School of Technology

Insert Node at the end:

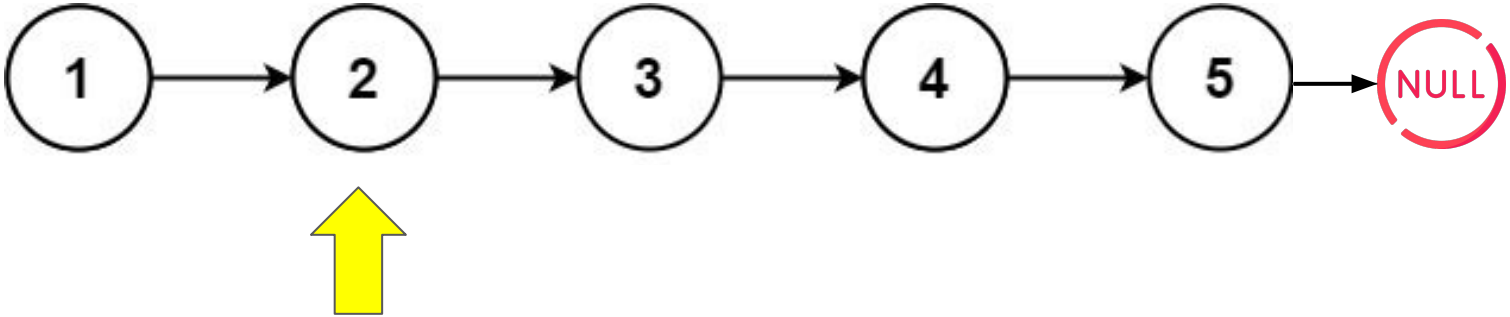


Node next is not Null

Jump to next node

Newton School of Technology

Insert Node at the end:

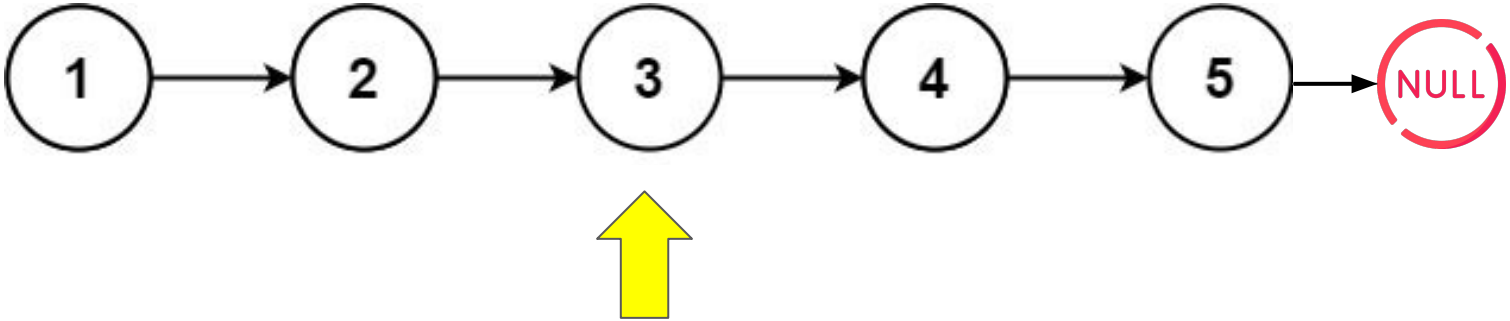


Node next is not Null

Jump to next node

Newton School of Technology

Insert Node at the end:

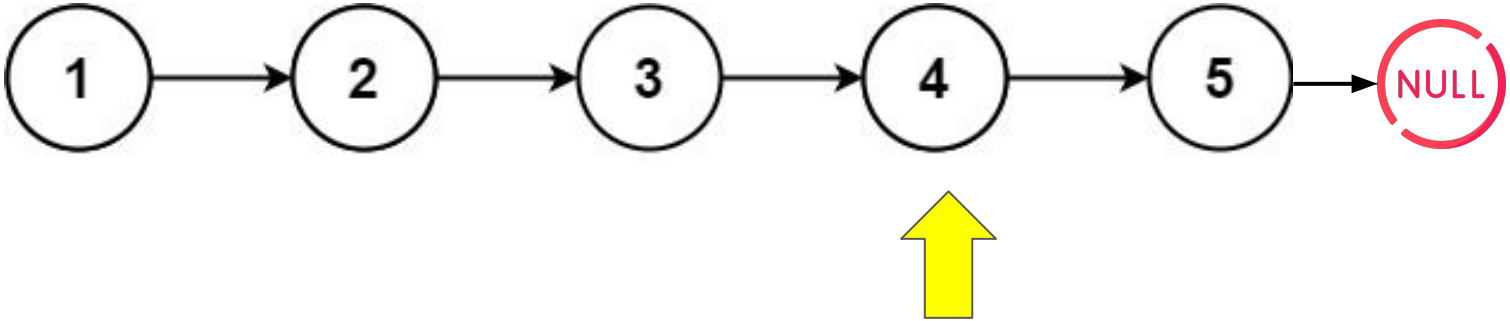


Node next is not Null

Jump to next node

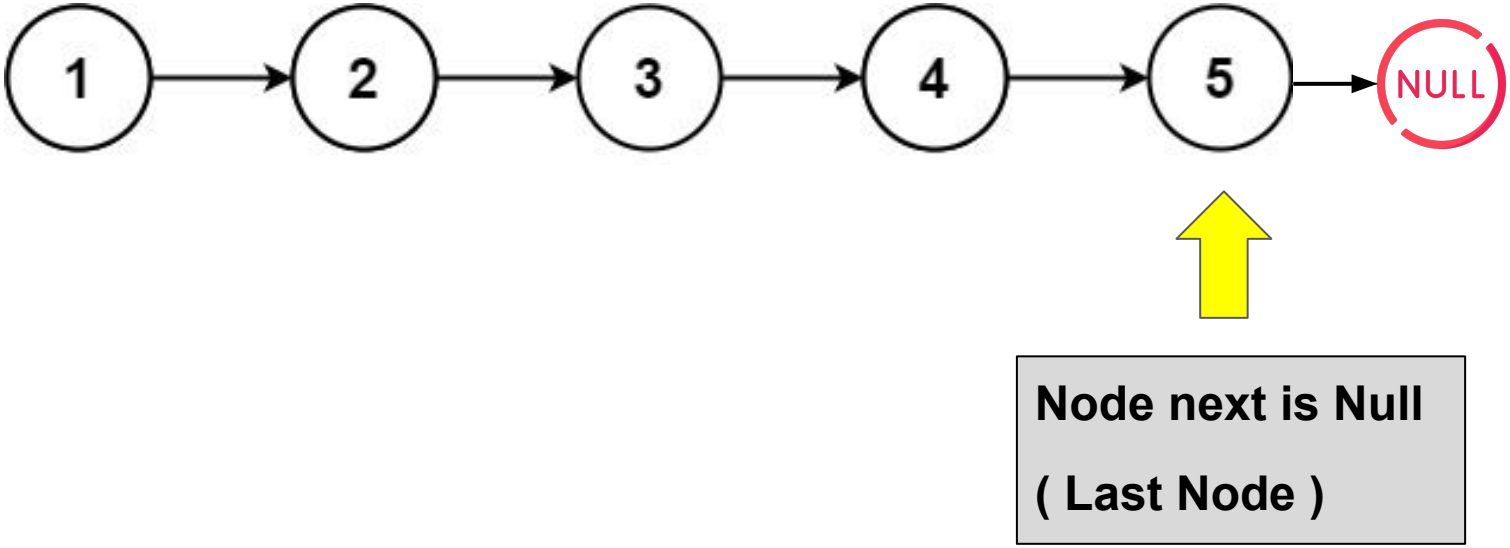
Newton School of Technology

Insert Node at the end:



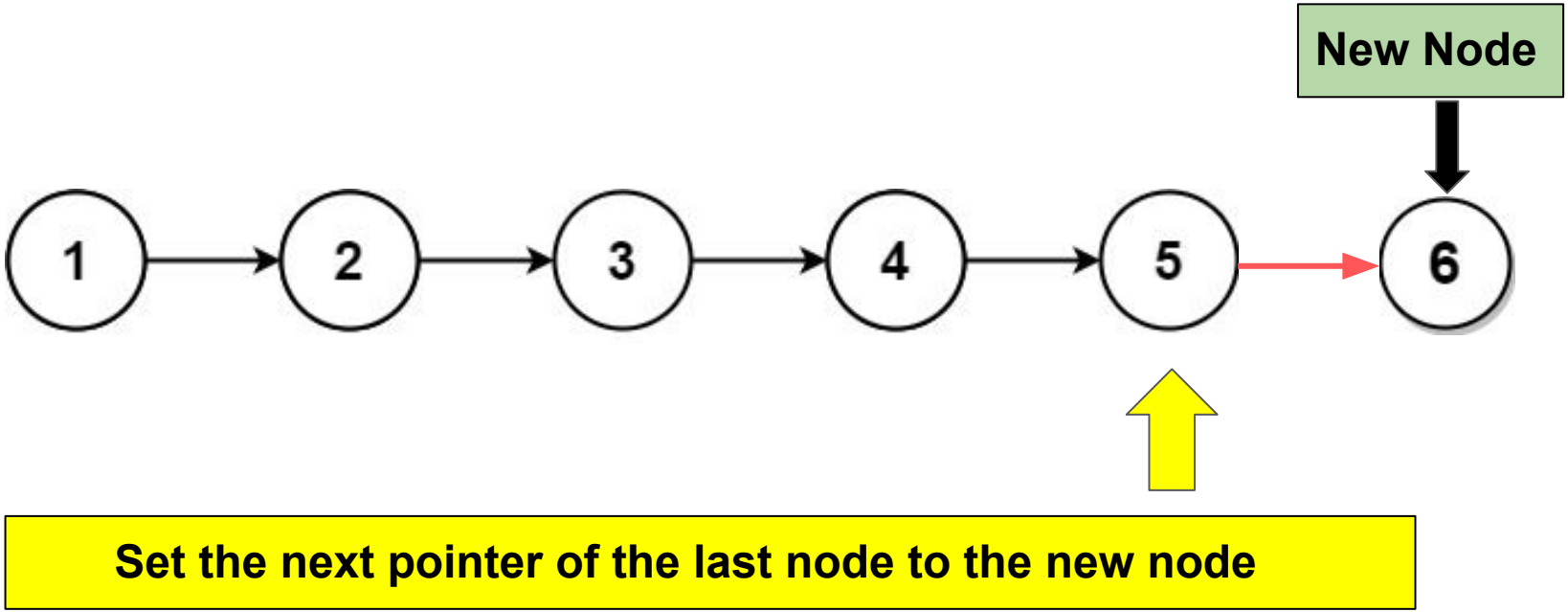
Newton School of Technology

Insert Node at the end:



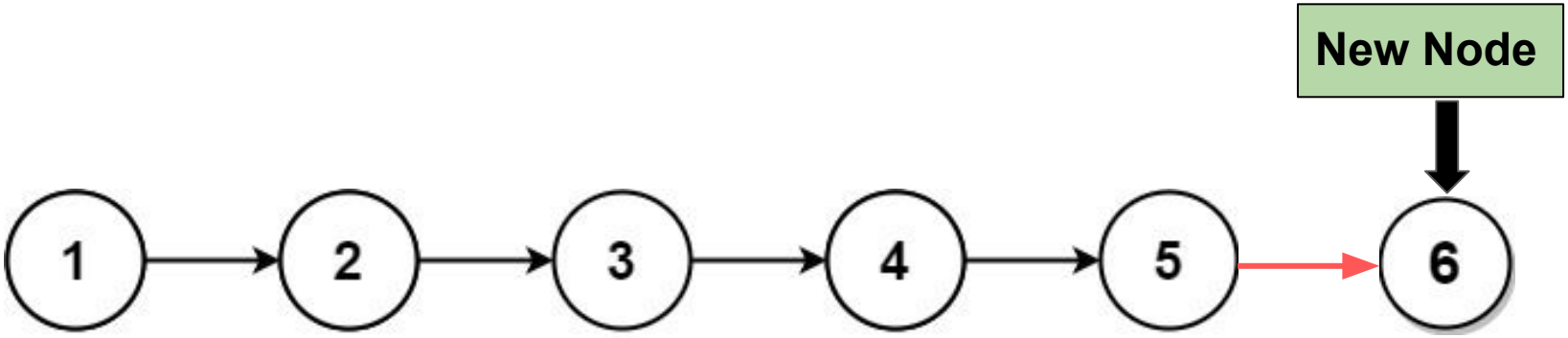
Newton School of Technology

Insert Node at the end:



Newton School of Technology

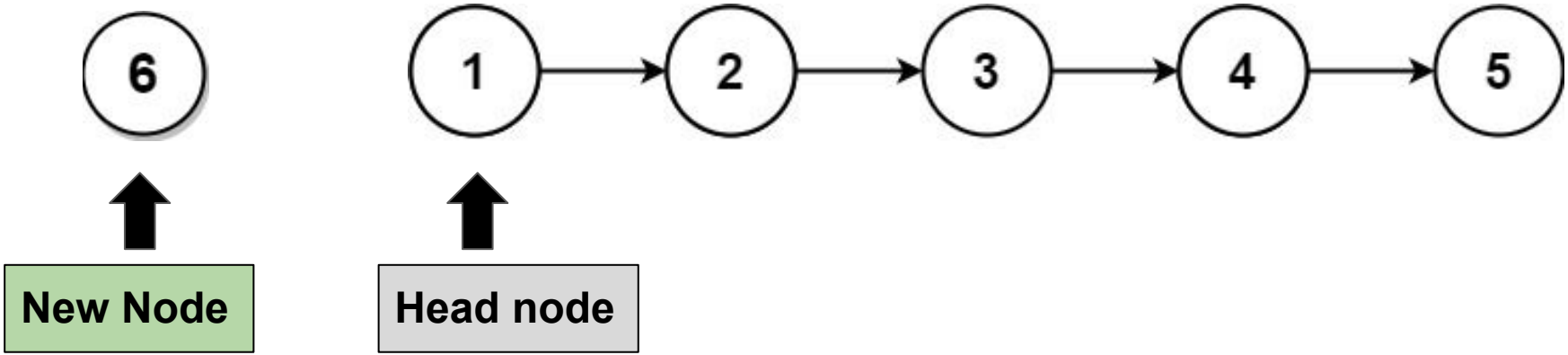
Insert Node at the end:



Time Complexity : O(n)

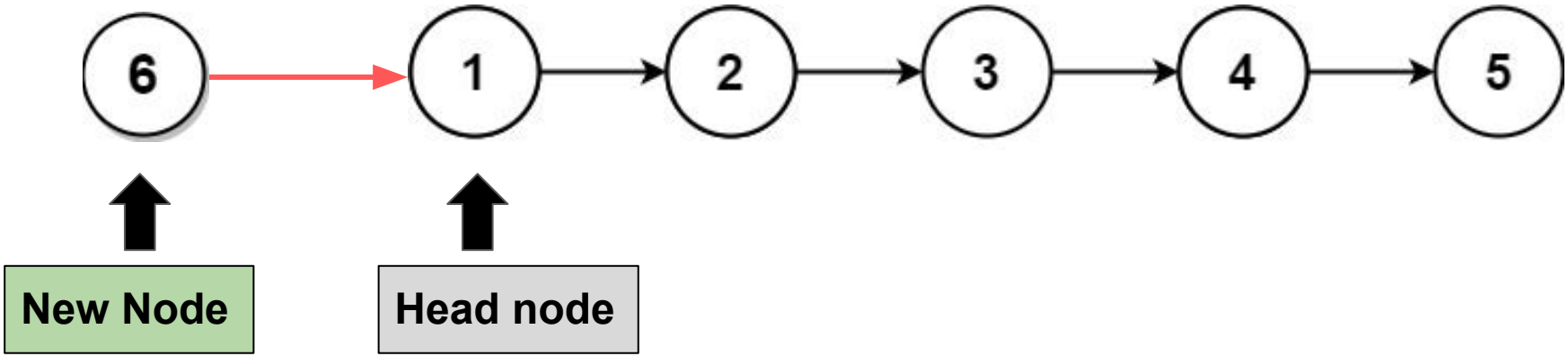
Newton School of Technology

Insert Node at the beginning:



Newton School of Technology

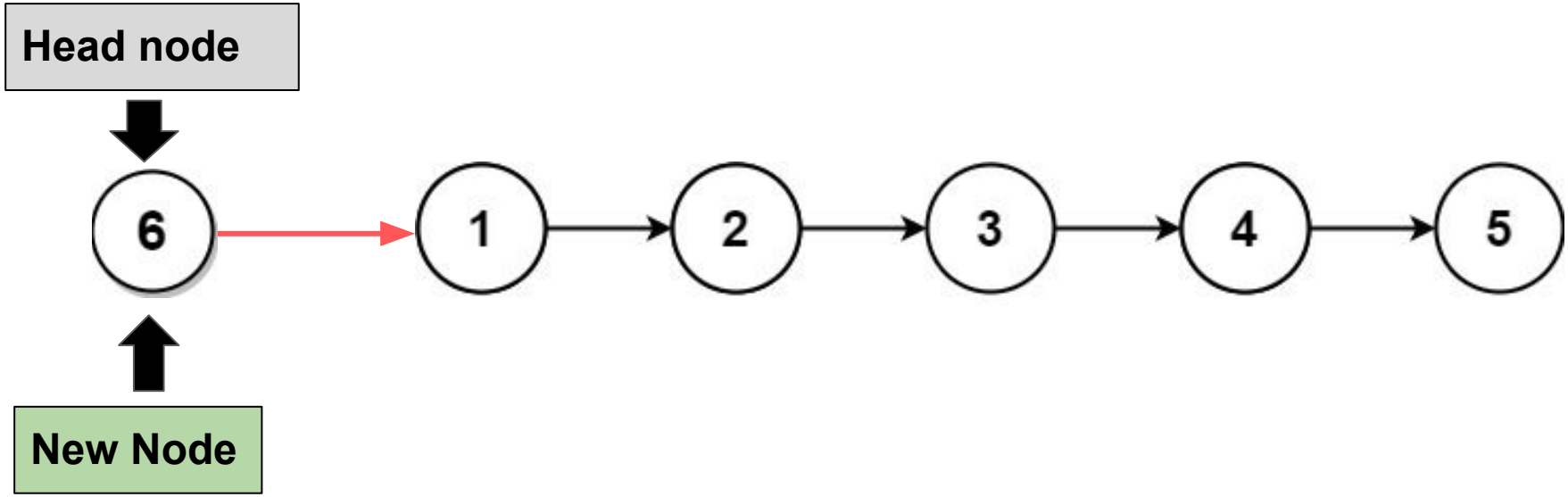
Insert Node at the beginning:



Set the next of the new node to the head node

Newton School of Technology

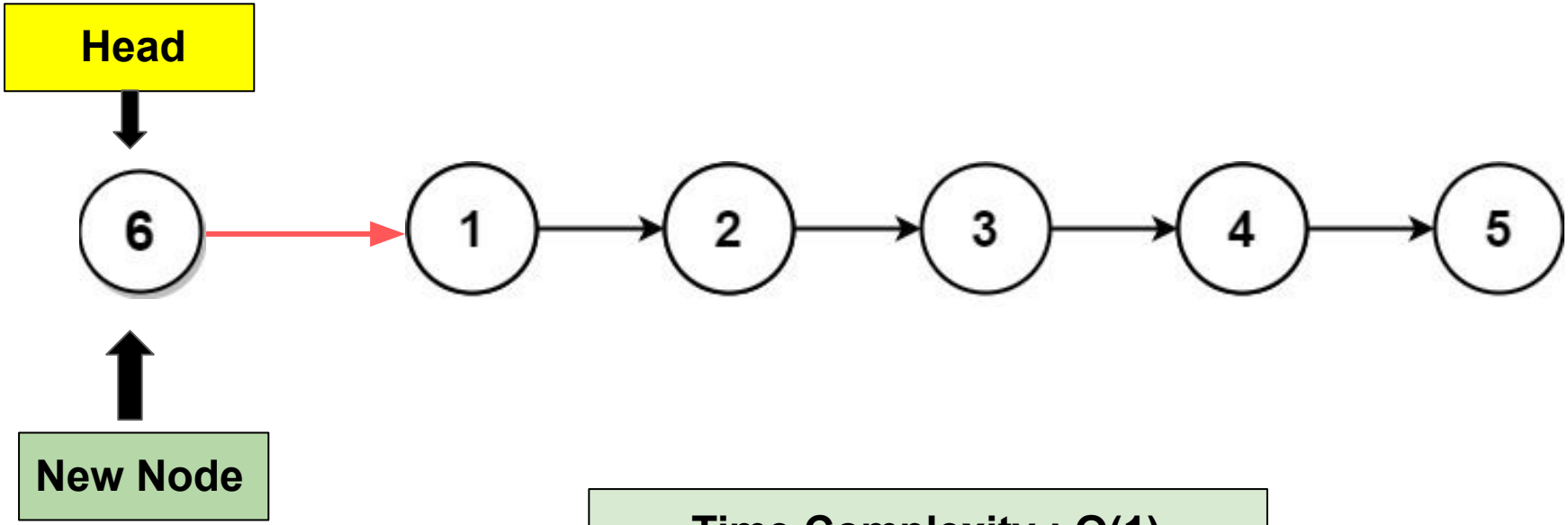
Insert Node at the beginning:



Update the head node to point to the new node

Newton School of Technology

Insert Node at the beginning:



Time Complexity : O(1)

Newton School of Technology

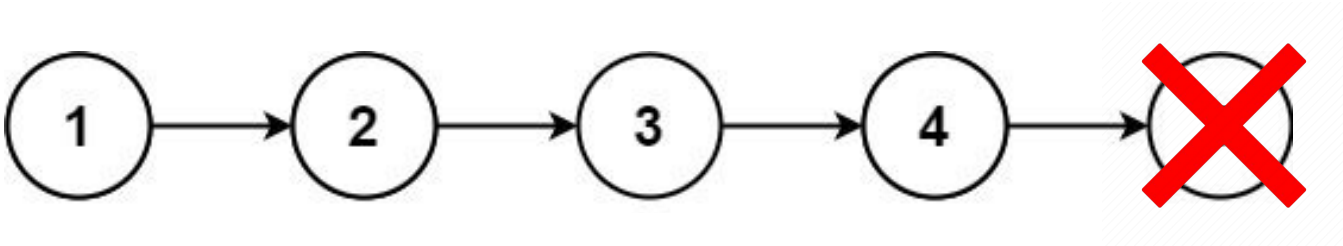
Deletion

Newton School of Technology

Delete in Linked List :

1. Delete Node at the beginning
2. Delete Node at the end

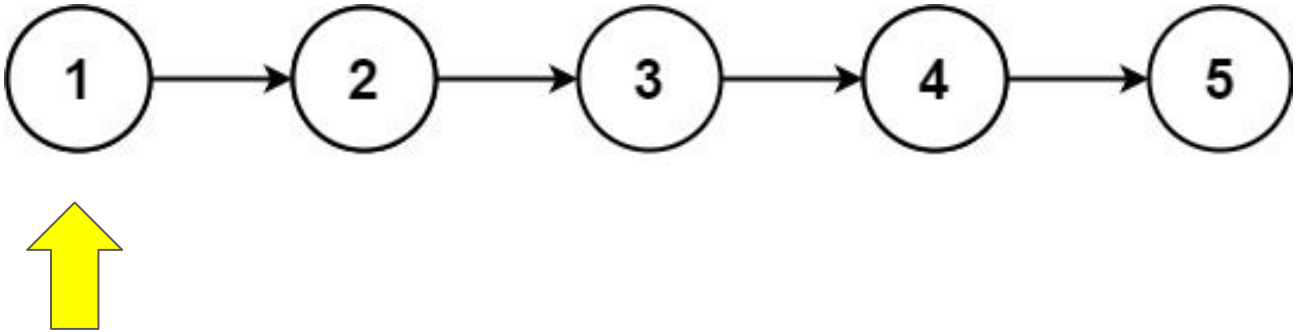
Delete last Node



1. Iterate until the second last node.
2. The second last node is where next.next is null.

Newton School of Technology

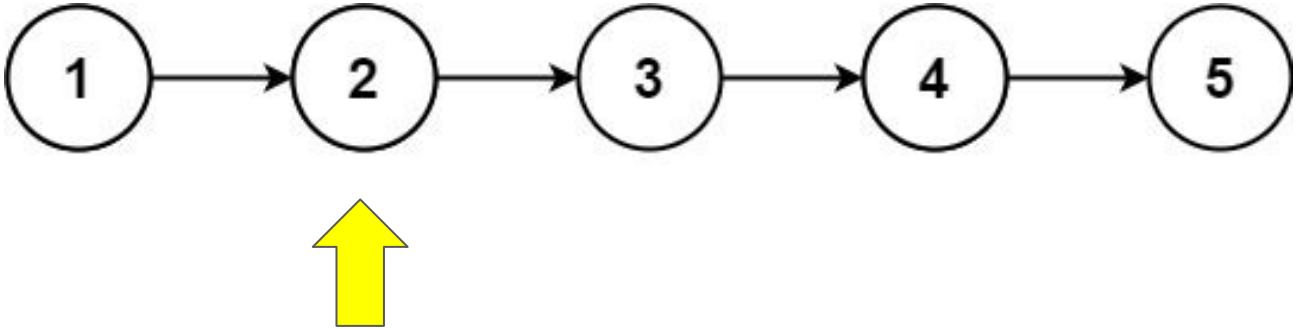
Delete last Node



Node next next is not Null

Jump to next node

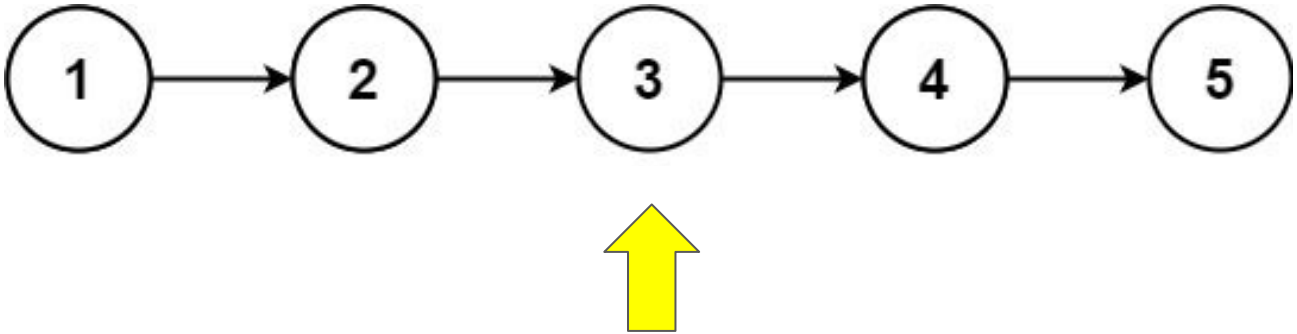
Delete last Node



Node next next is not Null

Jump to next node

Delete last Node

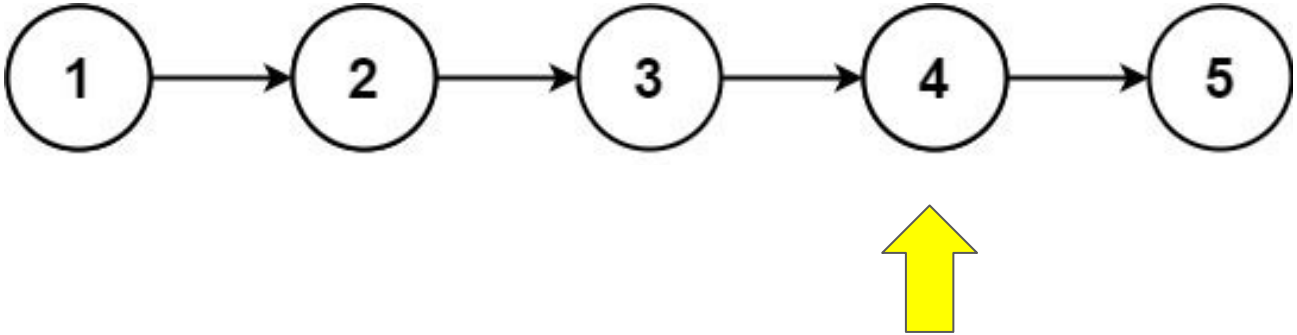


Node next next is not Null

Jump to next node

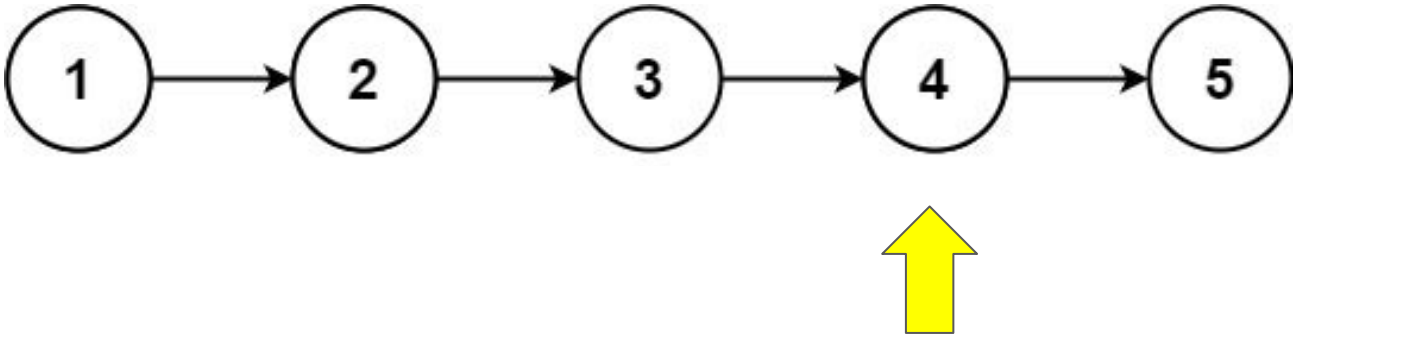
Newton School of Technology

Delete last Node



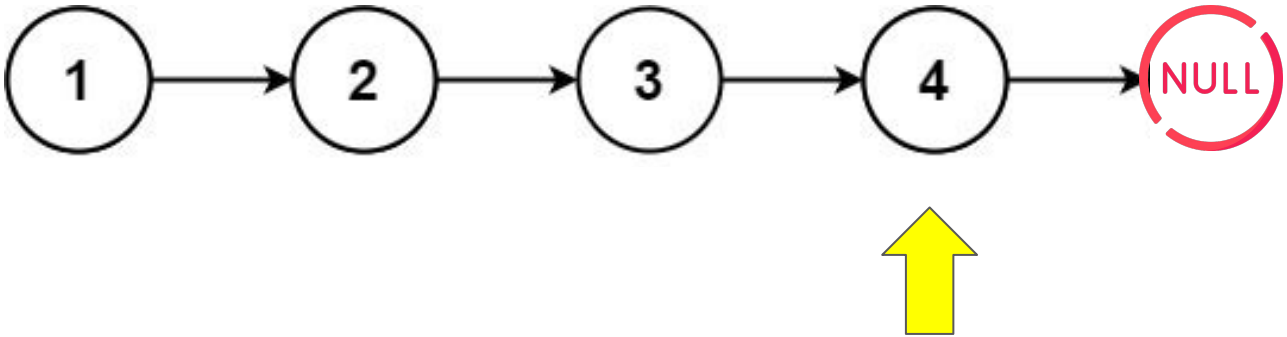
Node next next is Null

Delete last Node



Set next pointer to NULL

Delete last Node



Set next pointer to NULL

Delete last Node

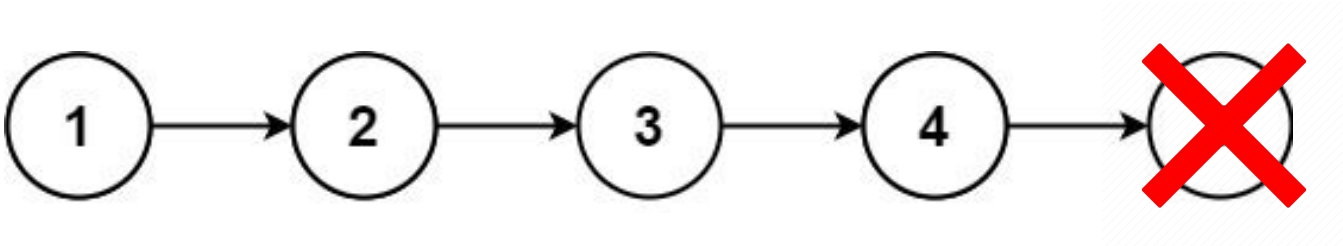
Special Case

Linked List has only one node

For this case we will remove current node and set head as NULL

Newton School of Technology

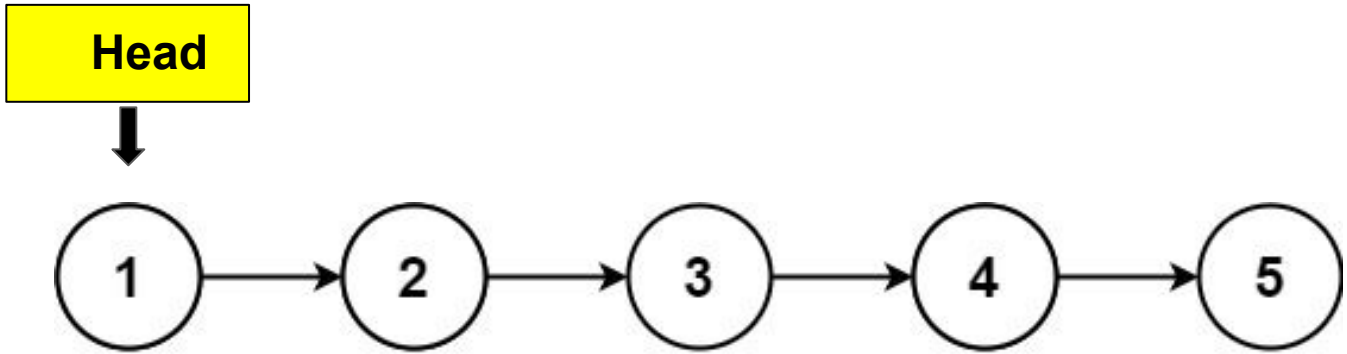
Delete last Node



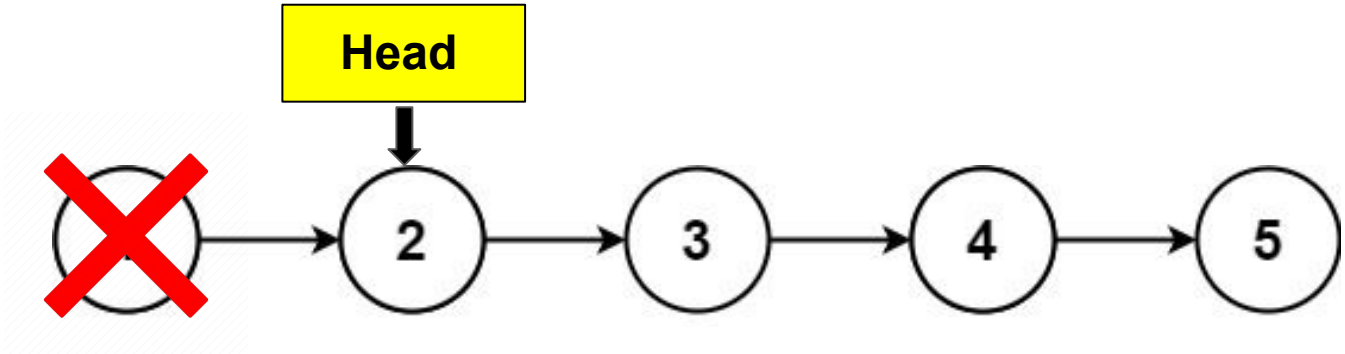
Time Complexity : O(n)

Newton School of Technology

Delete first Node



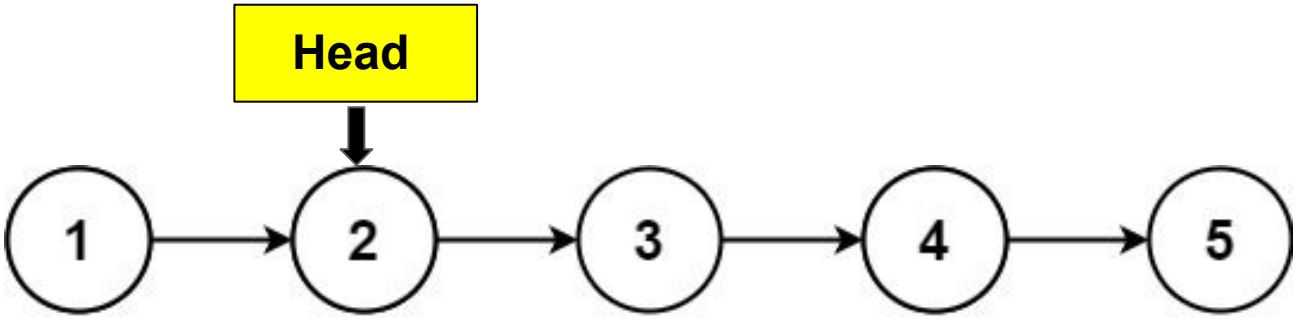
Delete first Node



To remove head node
Set Head pointer to it’s next node

Newton School of Technology

Delete first Node

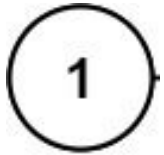


Head is pointing to it's next node
We never iterate backwards

Delete first Node

Special Case

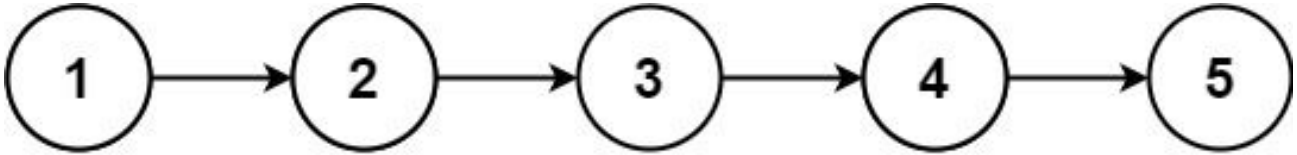
Linked List has only one node



For this case we will remove current node and set head as NULL

Newton School of Technology

Delete first Node



Time Complexity : O(1)

Newton School of Technology

END