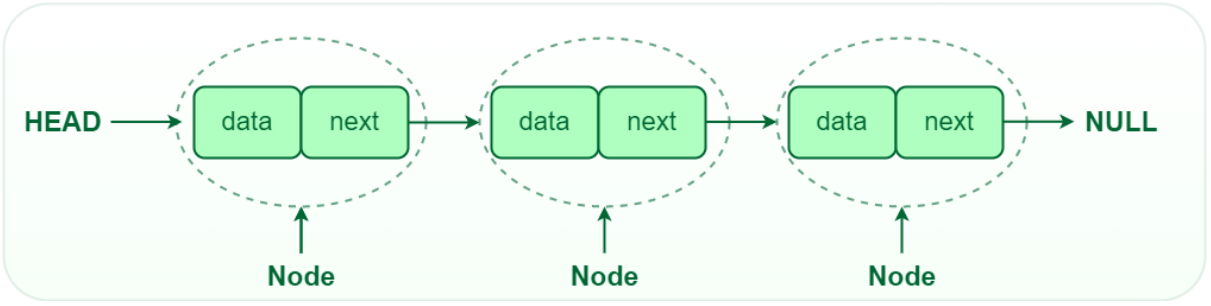
Linked List

> *A linked list is a collection of “nodes” connected together via links. These nodes consist of the data to be stored and a pointer to the address of the next node within the linked list.*



Data: It holds the actual value or data associated with the node.

Next Pointer: It stores the memory address (reference) of the next node in the sequence.

Head and Tail: The linked list is accessed through the head node, which points to the first node in the list. The last node in the list points to NULL or nullptr, indicating the end of the list. This node is known as the tail node.

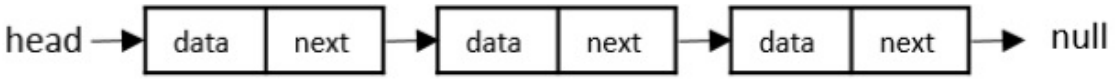
📌 *A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.*

Why Linked List is needed?

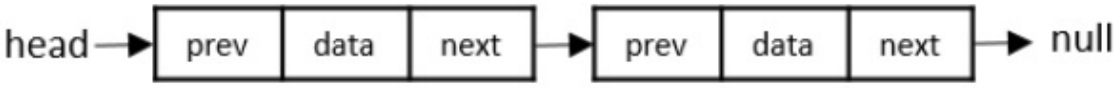
- **Dynamic Data structure:** The size of memory can be allocated or de-allocated at run time based on the operation.
- **Ease of Insertion/Deletion:** The insertion and deletion of elements are simpler. Elements need to be shifted after insertion and deletion, Just the address needed to be updated.
- **Efficient Memory Utilization:** As we know Linked List is a dynamic data structure the size increases or decreases as per the requirement so this avoids the wastage of memory.
- **Implementation:** Various advanced data structures can be implemented using a linked list like a stack, queue, graph, hash maps, etc.

Types of Linked List

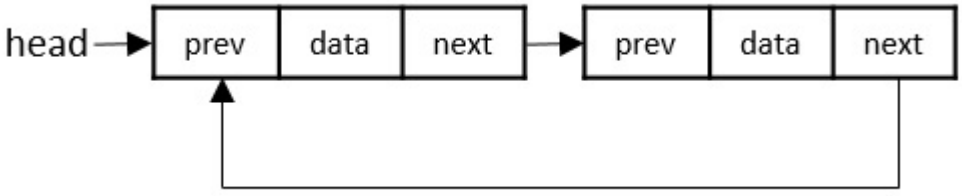
1. **Singly Linked List :** The nodes only point to the address of the next node in the list. It contains two “buckets” in one node; one bucket holds the data and the other bucket holds the address of the next node of the list.Traversals can be done in one direction only as there is only a single link between two nodes of the same list.



2. **Doubly Linked List :** The nodes point to the addresses of both previous and next nodes. It contains three “buckets” in one node; one bucket holds the data and the other buckets hold the addresses of the previous and next nodes in the list. The list is traversed twice as the nodes in the list are connected to each other from both sides.



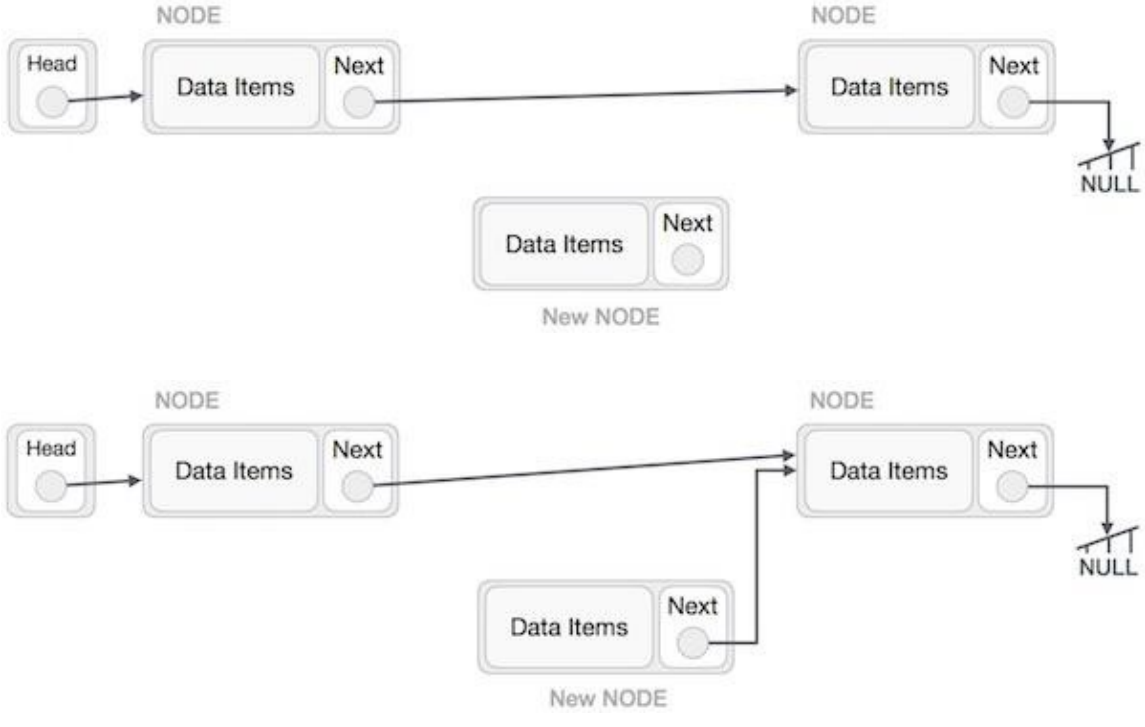
3. **Circular Linked List :** The last node in the list will point to the first node in the list. It can either be singly linked or doubly linked. Since the last node and the first node of the circular linked list are connected, the traversal in this linked list will go on forever until it is broken.

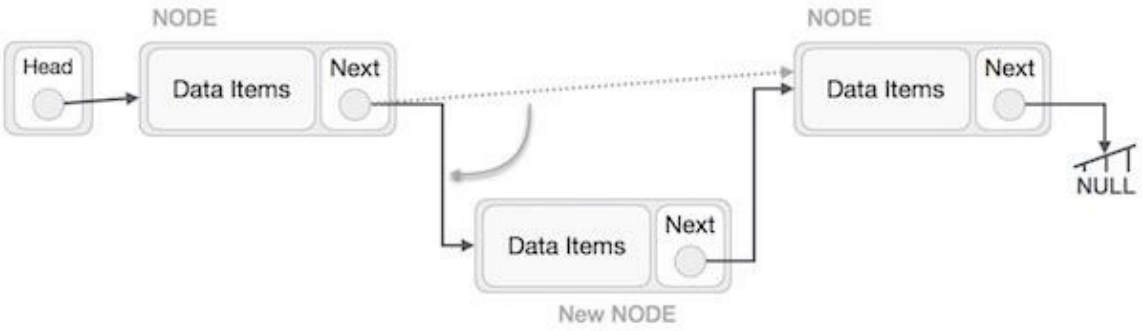


📌 **Traversing** a data structure means: "visiting" or "touching" the elements of the structure, and doing something with the data

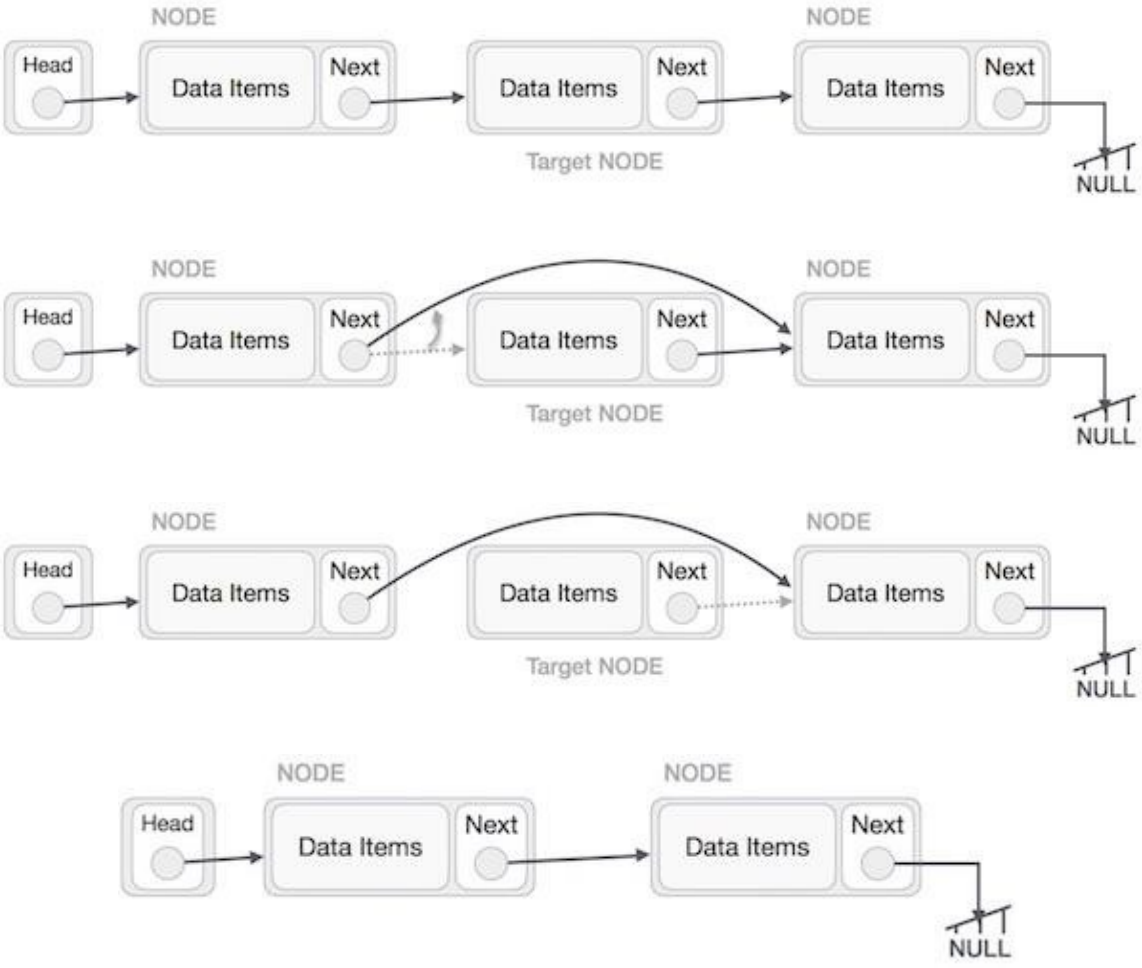
Basic Operations Schemes and Codes for Linked List

1. **INSERTION**

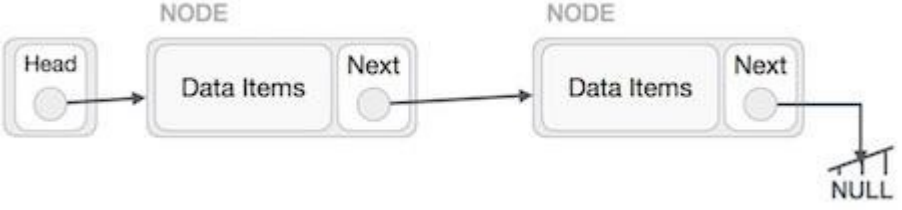


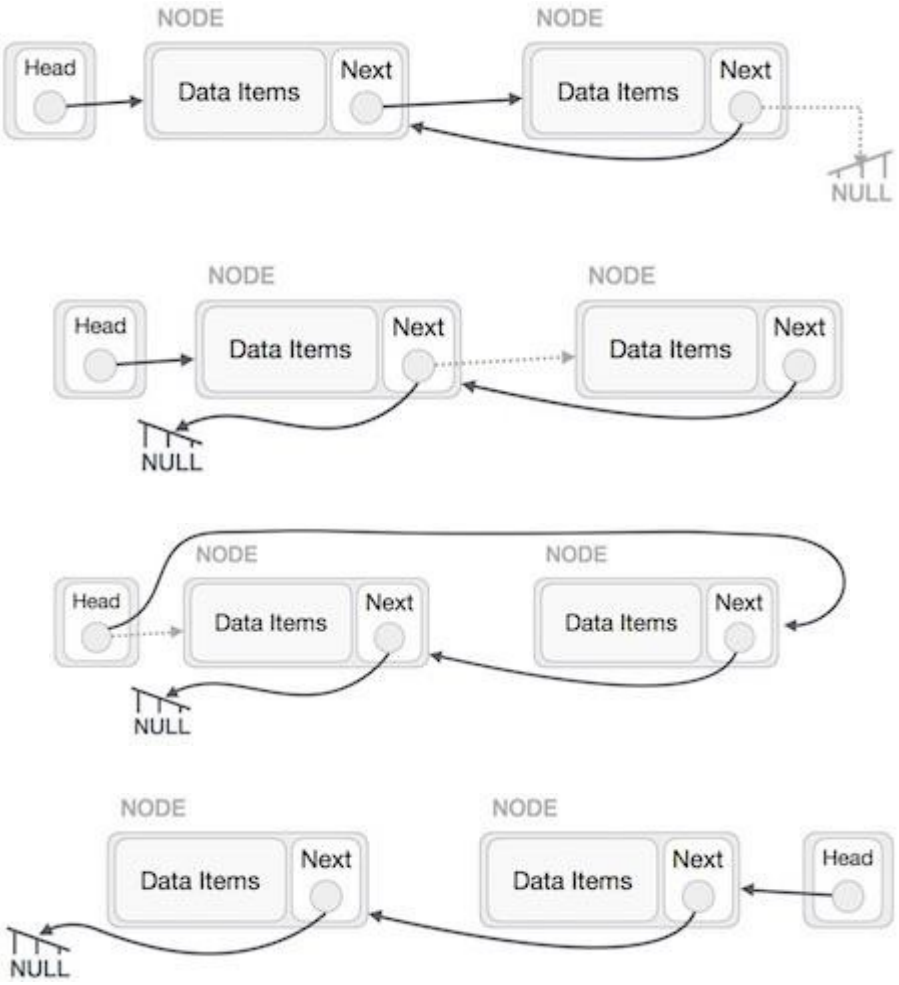


2. **DELETION**



3. **REVERSE**





Advantages Of Linked List

- **Dynamic size:** Linked lists do not have a fixed size, so you can add or remove elements as needed, without having to worry about the size of the list.
- **Efficient Insertion and Deletion:** Inserting or deleting elements in a linked list is fast and efficient, as you only need to modify the reference of the next node, which is an O(1) operation.
- **Memory Efficiency:** Linked lists use only as much memory as they need, so they are more efficient

Disadvantages Of Linked List

- **Slow Access Time:** Accessing elements in a linked list can be slow, as you need to traverse the linked list to find the element you are looking for, which is an O(n) operation.
- **Pointers:** Linked lists use pointers to reference the next node, which can make them more complex to understand and use compared to arrays. This complexity can make linked lists more difficult to debug and maintain.

with memory compared to arrays, which have a fixed size and can waste memory if not all elements are used.

- **Easy to Implement:** Linked lists are relatively simple to implement and understand compared to other data structures like trees and graphs.
- **Flexibility:** Linked lists can be used to implement various abstract data types, such as stacks, queues, and associative arrays.
- **Easy to navigate:** Linked lists can be easily traversed, making it easier to find specific elements or perform operations on the list.
- **Higher overhead:** Linked lists have a higher overhead compared to arrays, as each node in a linked list requires extra memory to store the reference to the next node.
- **Cache Inefficiency:** Linked lists are cache-inefficient because the memory is not contiguous. This means that when you traverse a linked list, you are not likely to get the data you need in the cache, leading to cache misses and slow performance.
- **Extra memory required:** Linked lists require an extra pointer for each node, which takes up extra memory. This can be a problem when you are working with large data sets, as the extra memory required for the pointers can quickly add up.

Applications of Linked Lists

- Linked Lists are used to implement stacks and queues.
- It is used for the various representations of trees and graphs.
- It is used in **dynamic memory allocation**(linked list of free blocks).
- It is used for representing **sparse matrices**.
- It is used for the manipulation of polynomials.
- It is also used for performing arithmetic operations on long integers.
- It is used for finding paths in networks.
- In operating systems, they can be used in Memory management, process scheduling and file system.

- Linked lists can be used to improve the performance of algorithms that need to frequently insert or delete items from large collections of data.
- Implementing algorithms such as the LRU cache, which uses a linked list to keep track of the most recently used items in a cache.

Applications of Linked Lists in real world

- The list of songs in the music player are linked to the previous and next songs.
- In a web browser, previous and next web page URLs are linked through the previous and next buttons.
- In image viewer, the previous and next images are linked with the help of the previous and next buttons.
- Switching between two applications is carried out by using “alt+tab” in windows and “cmd+tab” in mac book. It requires the functionality of circular linked list.
- In mobile phones, we save the contacts of the people. The newly entered contact details will be placed at the correct alphabetical order. This can be achieved by linked list to set contact at correct alphabetical position.
- The modifications that we make in documents are actually created as nodes in doubly linked list. We can simply use the undo option by pressing Ctrl+Z to modify the contents. It is done by the functionality of linked list.

Array VS Linked List

ARRAY	LINKED LISTS
1. Arrays are stored in contiguous location.	1. Linked lists are not stored in contiguous location.
2. Fixed in size.	2. Dynamic in size.
3. Memory is allocated at compile time.	3. Memory is allocated at run time.
4. Uses less memory than linked lists.	4. Uses more memory because it stores both data and the address of next node.
5. Elements can be accessed easily.	5. Element accessing requires the traversal of whole linked list.
6. Insertion and deletion operation takes time.	6. Insertion and deletion operation is faster.

📚 References

GeeksforGeeks | A computer science portal for geeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive

https://www.geeksforgeeks.org/

Online Tutorials, Courses, and eBooks Library | Tutorialspoint

Tutorialspoint is an online learning platform providing free tutorials, paid premium courses, and eBooks. Learn the latest technologies and programming languages C, C++, Java, Python, PHP, Machine Learning, data science, AI, and more.

https://www.tutorialspoint.com

✍🏻 Author → Serhat Kumas

https://www.linkedin.com/in/serhatkumas/

SerhatKumas - Overview

Computer engineering student who loves coding in different fields instead of focusing on a one spesific area. - SerhatKumas

https://github.com/SerhatKumas