
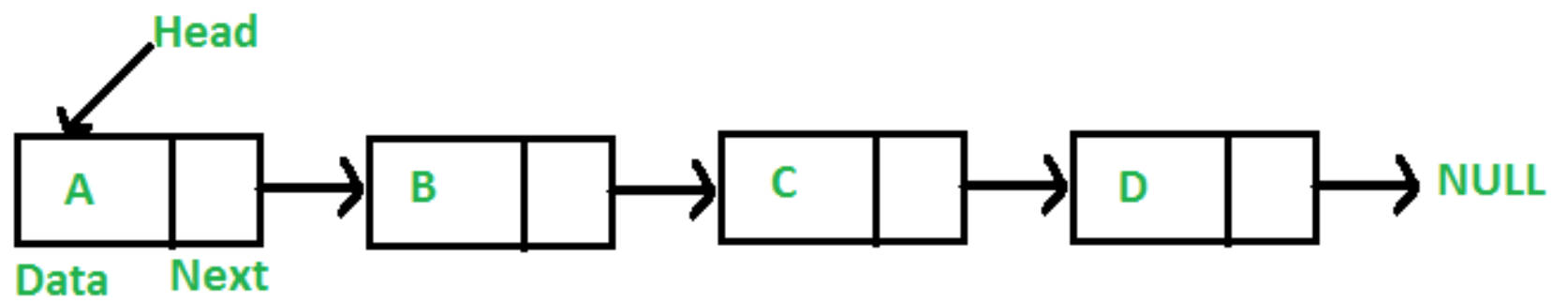


# LINKED LIST



**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 as shown in the below image:**





**In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.**

# TYPES

**Singly Linked List**

**Circular Linked List**

**Doubly Linked List**

# INTRODUCTION

**Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers.**

# Why Linked List?

**Arrays can be used to store linear data of similar types, but arrays have the following limitations.**

- 1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.**
- 2) Inserting a new element in an array of elements is expensive because the room has to be created for the new elements and to create room existing elements have to be shifted.**

**For example, in a system, if we maintain a sorted list of IDs in an array `id[]`.**

**`id[] = [1000, 1010, 1050, 2000, 2040].`**

# TYPES

**And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).**

**Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.**



# TYPES

## **Advantages over arrays**

- 1) Dynamic size**
- 2) Ease of insertion/deletion**

# TYPES

## **Drawbacks:**

- 1) Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation. [Read about it here.](#)**
- 2) Extra memory space for a pointer is required with each element of the list.**
- 3) Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.**

# TYPES

## **Representation:**

**A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head is NULL.**

**Each node in a list consists of at least two parts:**

**1) data**

**2) Pointer (Or Reference) to the next node**

**In C, we can represent a node using structures. Below is an example of a linked list node with integer data.**

**In Java or C#, LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.**

# TYPES

```
class Node {  
public:  
    int data;  
    Node* next;  
};
```

# Applications of Linked List

**Implementation of stacks and queues**

**Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to store adjacent vertices.**

**Dynamic memory allocation : We use linked list of free blocks.**

**Maintaining directory of names**

**Performing arithmetic operations on long integers**

**Manipulation of polynomials by storing constants in the node of linked list**

**representing sparse matrices**