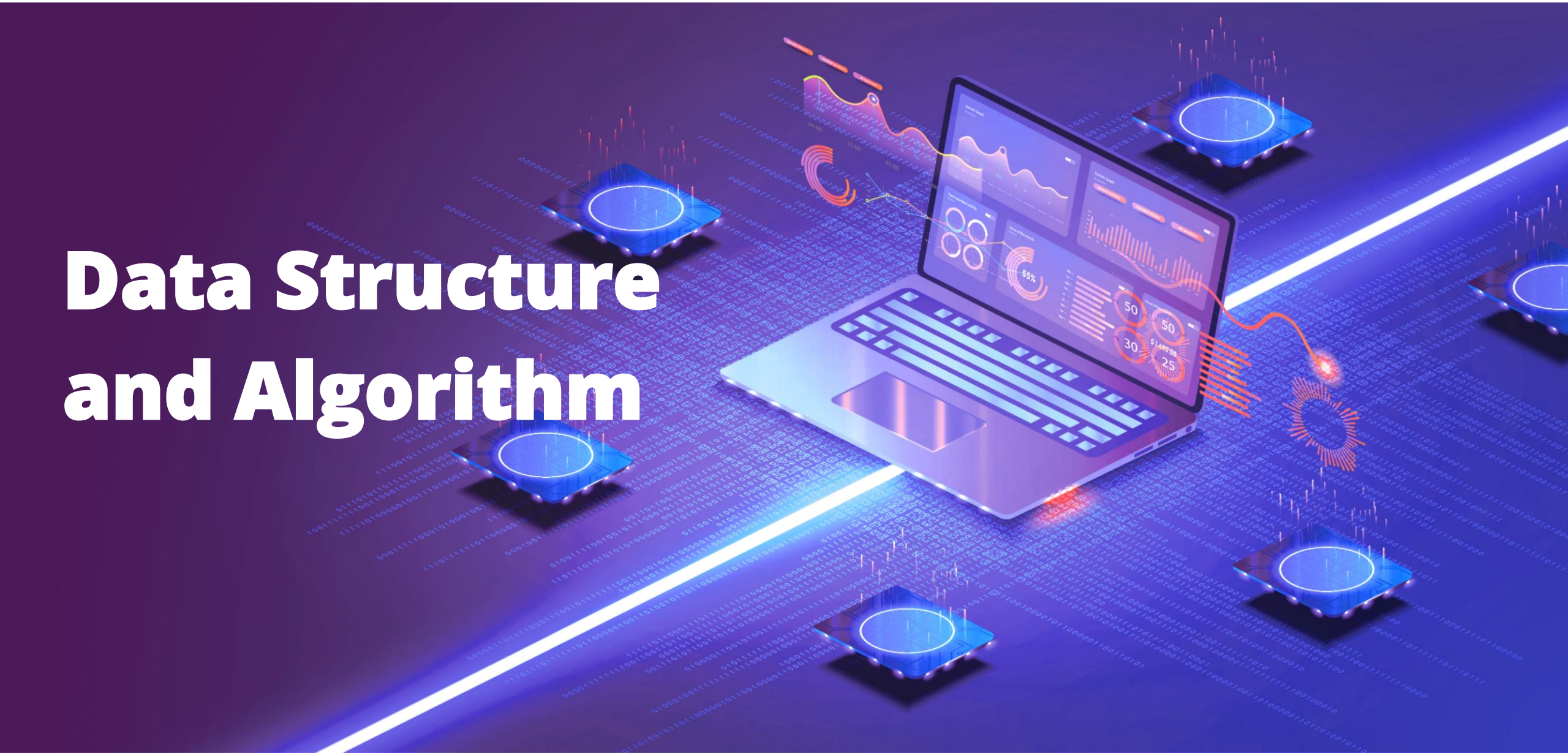
WILEY

ENABLING DISCOVERY | POWERING EDUCATION | SHAPING WORKFORCES

Data Structure and Algorithm



DATA STRUCTURES ALGORITHMS

LINKED LIST

Linked List

A container where data is stored in nodes consisting of a single data item and a reference to the next node.

Linked List Data Structure

Let’s understand about linked list data structure and it's implementation in Python, Java, C, and C++.

A linked list is a linear data structure that includes a series of connected nodes. Here, each node stores the **data** and the **address** of the next node. For example,

You have to start somewhere, so we give the address of the first node a special name called HEAD. Also, the last node in the linked list can be identified because its next portion points to NULL.

Linked lists can be of multiple types: **singly**, **doubly**, and **circular linked list**. In this article, we will focus on the **singly linked list**.

Representation of Linked List

Let's see how each node of the linked list is represented. Each node consists:

- A data item
- An address of another node

Note: You might have played the game Treasure Hunt, where each clue includes the information about the next clue. That is how the linked list operates.

The power of a linked list comes from the ability to break the chain and rejoin it.

E.g. if you wanted to put an element 4 between 1 and 2, the steps would be:

- Create a new struct node and allocate memory to it.
- Add its data value as 4
- Point its next pointer to the struct node containing 2 as the data value
- Change the next pointer of "1" to the node we just created.

Doing something similar in an array would have required shifting the positions of all the subsequent elements.

In python and Java, the linked list can be implemented using classes

Similar to a chain

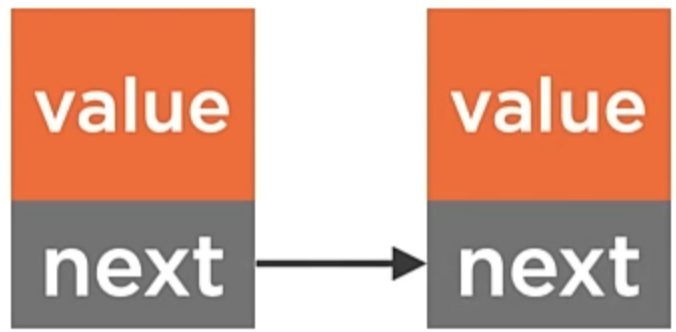
Start at the first link

Follow the chain to the last link

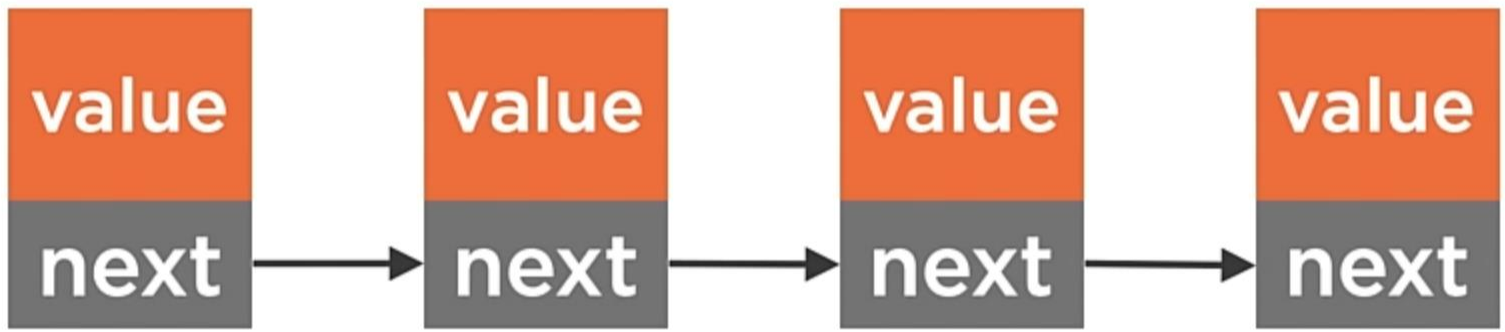
The Node

The Node

The Node



The Node

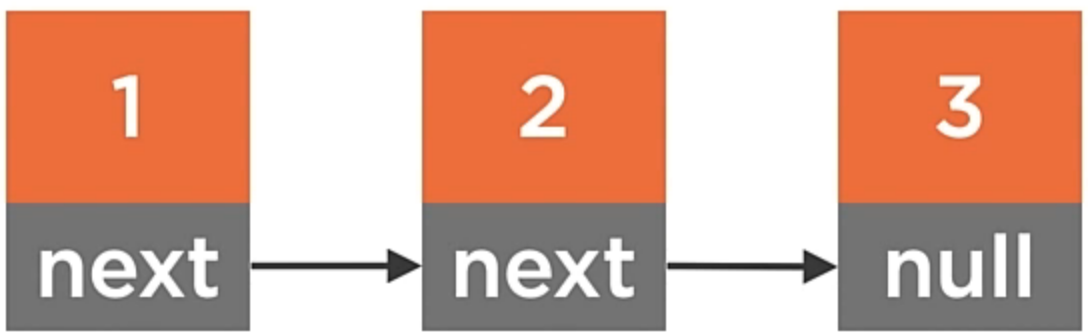


```
Node head = new Node(1);

head.Next = new Node(2);

head.Next.Next = new Node(3);
```

Connecting Nodes into a List



Singly Linked List

A linked list that provides forward iteration from the start to the end of the list.

```
class LinkedListNode<TNode> {

    public LinkedListNode(TNode value, LinkedListNode<TNode> next = null) {
        this.Value = value;
        this.Next = next;
    }

    public LinkedListNode<TNode> Next;
    public TNode Value;
}
```

Singly Linked List Node

A generic class containing the data and reference to the next node

Doubly Linked List

A linked list that provides forward iteration from the start to the end of the list, and reverse iteration, from end to start.

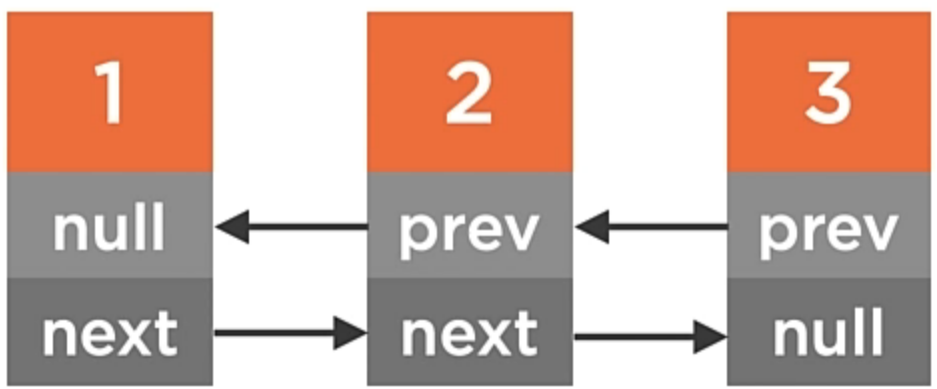
```
class Node
{
    public Node(int value)
    {
        this.Value = value;
        this.Previous = null;
        this.Next = null;
    }

    public Node Previous;
    public Node Next;
    public int Value;
}
```

```
node2.Next = node3;

node3.Previous = node2;
```

Connecting Doubly Linked Nodes Into a List



Doubly Linked List Node

```
class DoublyLinkedListNode<TNode>
{
    public DoublyLinkedListNode(TNode value,
                                Node<TNode> prev = null,
                                Node<TNode> next = null) {
        this.Value = value;
        this.Previous = prev;
        this.Next = next;
    }

    public Node<TNode> Previous;
    public Node<TNode> Next;
    public TNode Value;
}
```

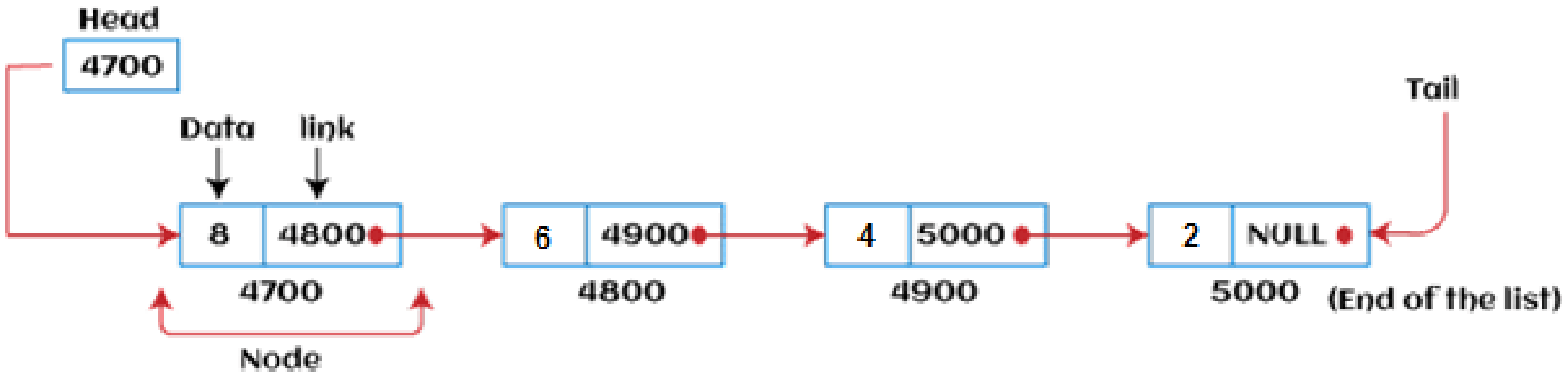
Linked List Implementation

Linked List Complexity

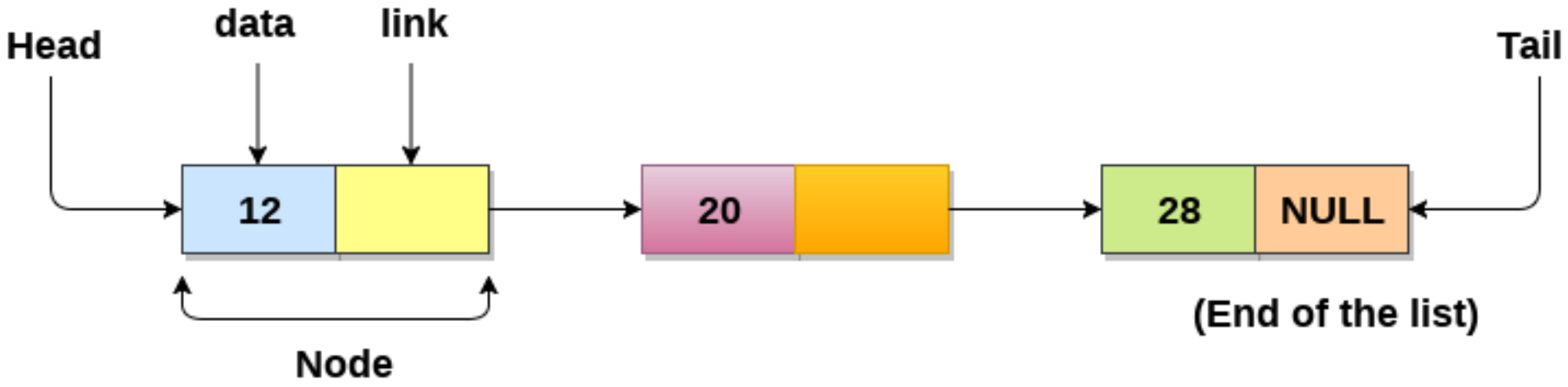
Time Complexity

	Worst case	Average Case
Search	O(n)	O(n)
Insert	O(1)	O(1)
Deletion	O(1)	O(1)

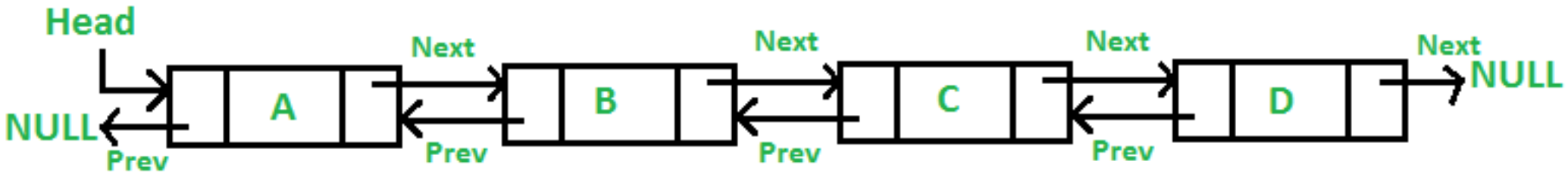
Space Complexity: `O(n)`


Head
4700
Data
link
8
4800
6
4900
4
5000
2
NULL
Tail
4700
4800
4900
5000
(End of the list)
Node

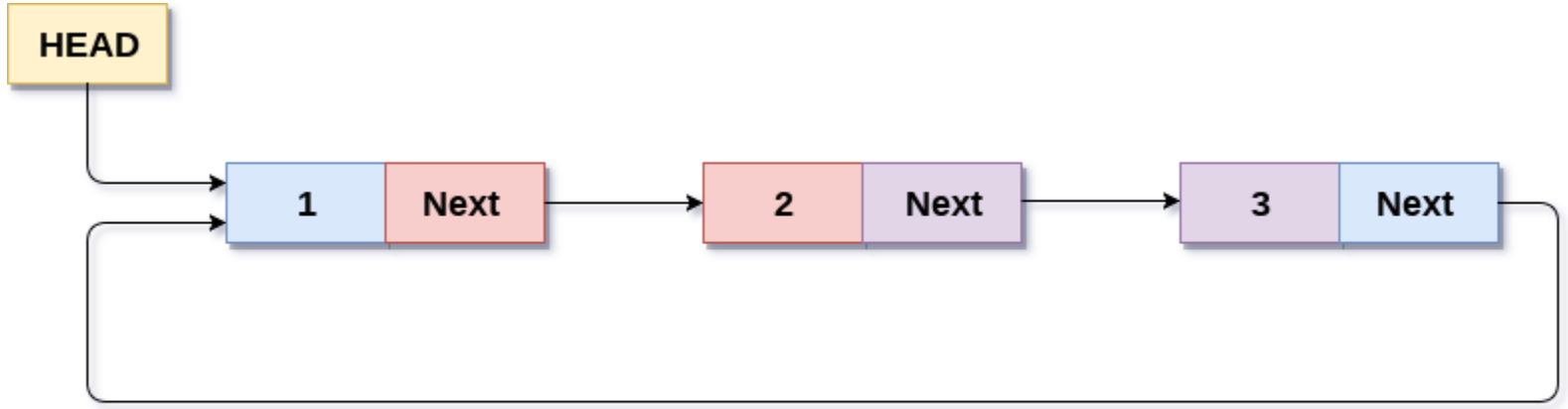
Singly Linked List



Doubly Linked List

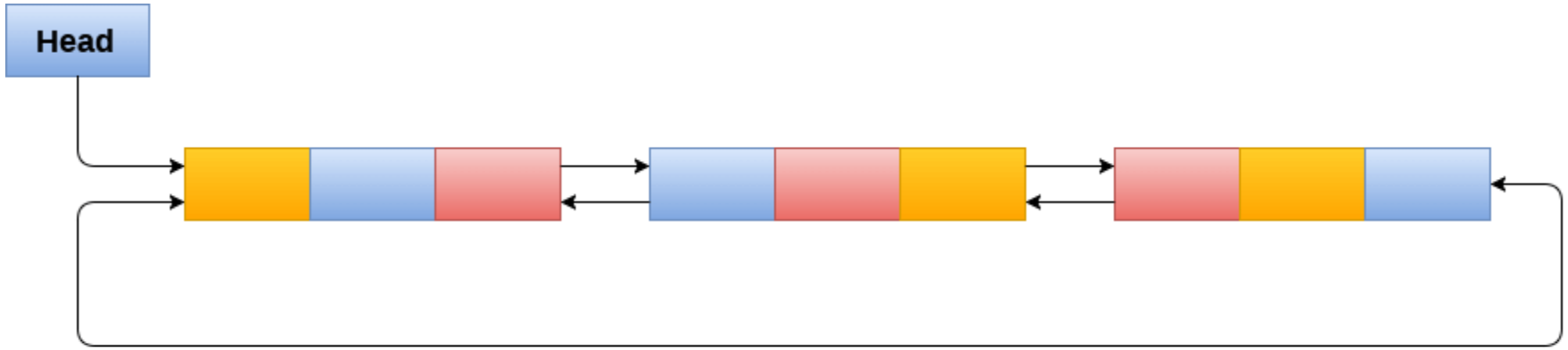


Circular Linked List



Circular Singly Linked List

Circular Doubly Linked List



Circular Doubly Linked List

Linked List Applications

- Dynamic memory allocation
- Implemented in stack and queue
- In **undo** functionality of softwares
- Hash tables, Graphs