DATA STRUCTURE CONCEPTS

PROF. NAVRATI SAXENA

Outline

1. Definition
2. Classification of Data Structures
3. Types of Data Structures
4. Applications of Data Structures

Definition

- **Data:** Collection of **raw facts**.
- **Data structure:** Representation of logical relationship existing between individual elements of data.
- Specialized format for organizing and storing data in memory
- Considering not only the elements stored but also their relationship

Data Structure is a way organizing data in such a way so that data can be easier to use

Why Data Structure ?

- Human requirement with computer are going to complex day by day.
- Data structure is used to solve the complex requirements in efficient way
- Helps in representation and organization of data
- Facilitates access and modification of data
- Provides fastest solution of human requirements
- Ensures efficient solution of complex problem
- Helps in formulating logical or mathematical description of the structure

Introduction

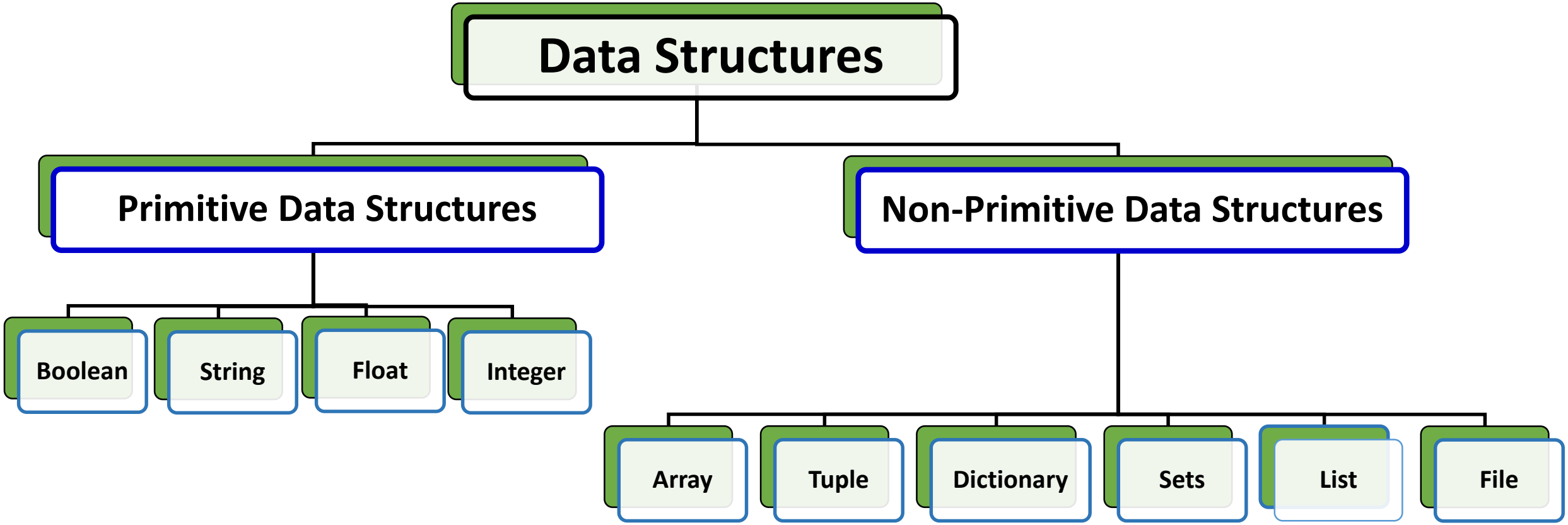
- Data structure affects the design of both structural and functional aspects of a program

  Program = Algorithm + Data Structure

- Algorithm is a step by step procedure to solve a particular function
- Data structure and algorithms are independent of programming language

NOTE: We will use C as a programming language, when required

Classification of Data Structure



Primitive Data Structure

- Basic structures and directly operated upon by machine instructions
- Data Structures that are directly operated upon the machine-level instructions – Primitive Data Structures
- **Common types**: Integer, Float, String, Boolean etc.
- Commonly used operation on data structure

- **Create**
- **Select**
- **Update**
- **Destroy or Delete**

Non-Primitive Data Structure (1/2)

- More sophisticated Data Structures
- Data Structures that are derived from primitive data structures – Non-Primitive Data Structures
- Emphasize on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items

Non-Primitive Data Structure (2/2)

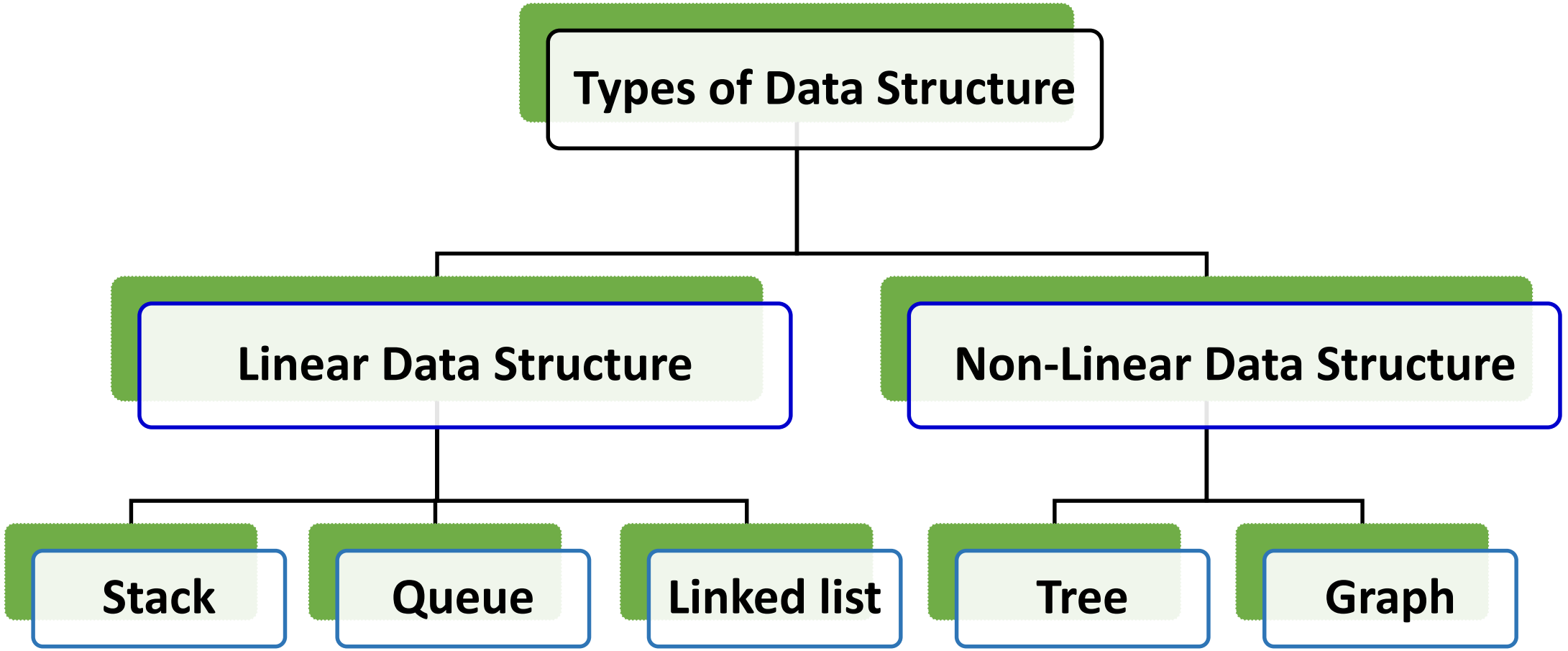
- **Commonly used operation on data structure**

Traversal
Insertion
Selection

Searching
Sorting

Merging
Destroy or Delete

Types of Data Structure



Types of Data Structure

1. Linear Data structures:

- Have homogeneous elements.
- Elements are in a sequence and form a linear series.
- Are easy to implement, since memory of the computer is organized in a linear fashion.
- Some commonly used linear data structures are **Stack, Queue and Linked Lists.**

2. Non-Linear Data structures:

- Data item is connected to several other data items.
- Exhibit either a hierarchical relationship or parent child relationship.
- Data elements are not arranged in a sequential structure.
- Different non-linear data structures are **trees and graphs.**

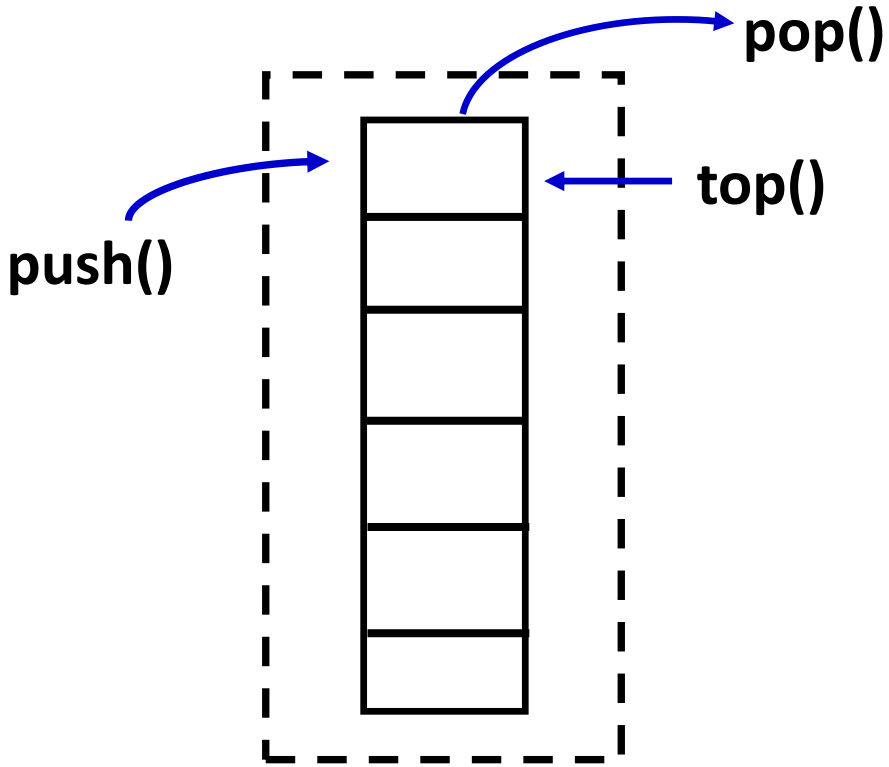
Types of Data Structure – Linear

Linear Data structures

- Homogeneous elements.
- Elements are in a sequence and form a linear series.
- Easy to implement, as computer-memory is organized in a linear fashion.
- Some commonly used linear data structures: **Stack, Queue, Linked Lists.**

Stack

- An abstract data type
- Allows adding and removing elements in a particular order
- Every time an element is added, it goes on the top of the stack
- The only element that can be removed is the element at the top of the stack

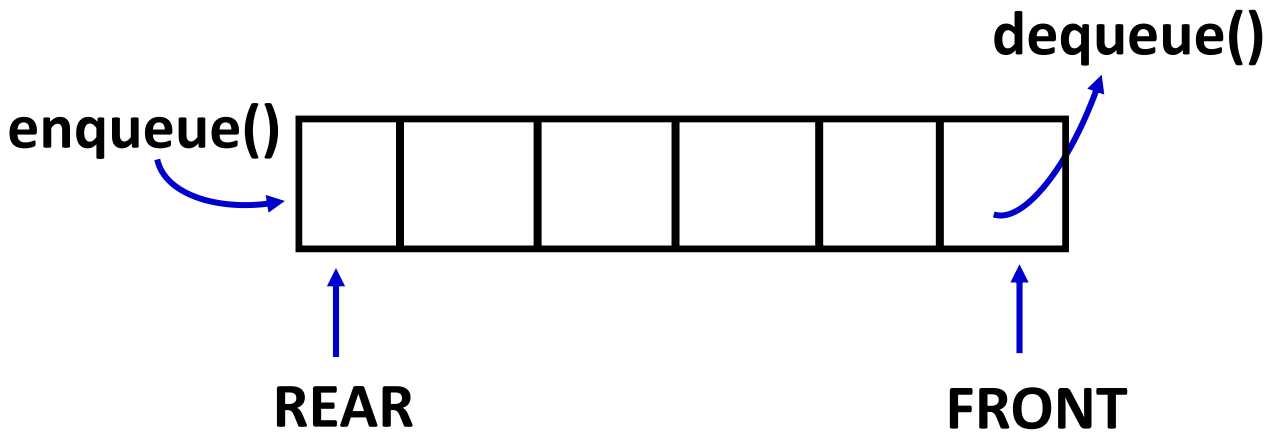


Basic Features of Stack

- Stack is an ordered list of similar data type.
- Stack is a **LIFO** (Last in First out) structure or we can say **FILO** (First in Last out).
- Function used to insert new elements: **push()**
- Function used to delete elements: **pop()**
- Both insertion and removal are allowed at only one end: **Top**.
- **Stack-Overflow**: Stack is completely full and is said to be in
- **Stack-Underflow**: Stack is completely empty.

Queue

- Queue is also an abstract data type or a linear data structure
- First element is inserted from one end called the **REAR**
- Removal of existing element takes place from the other end called as **FRONT**
- This makes queue as **FIFO** (First in First Out) data structure, which means that element inserted first will be removed first.

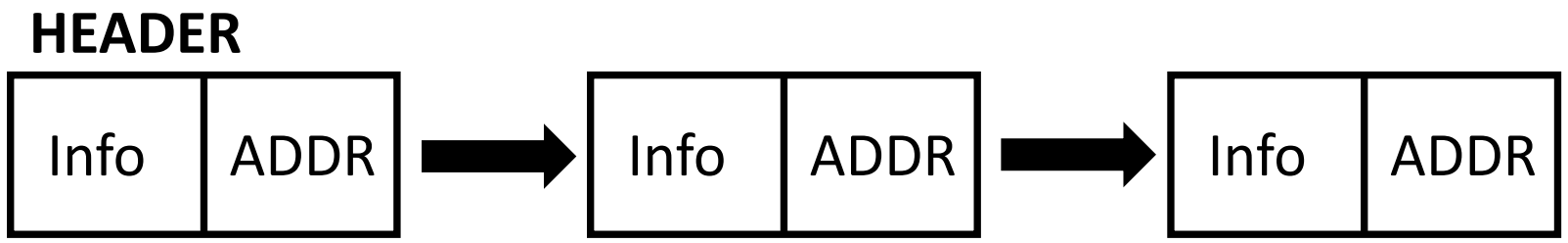


Basic features of Queue

- Queue is an ordered list of elements of similar data types.
- Queue is a First in First out (**FIFO)** data structure.
- Once a new element is inserted into the Queue, all the elements inserted before the new element must be removed, to remove the new element.
- **enque:** inserting element in the **rear** of queue
- **dequeue**: removing element from the **front** of queue
- **peek( )** function is oftenly used to return the value of first element without dequeuing it.

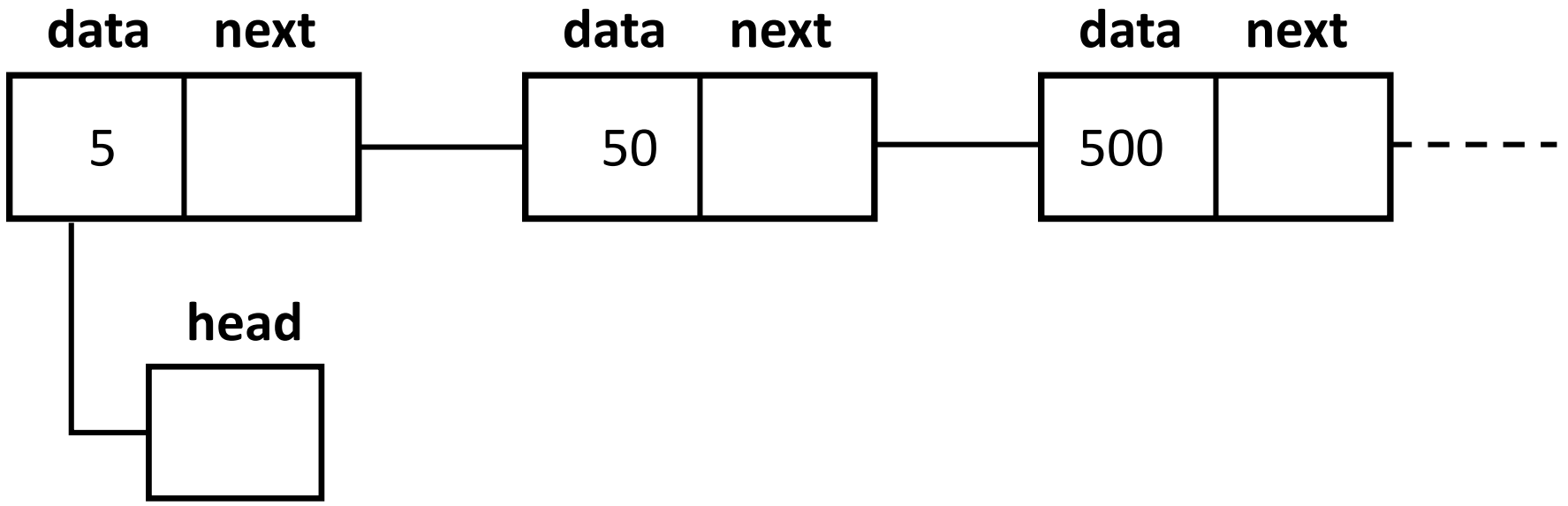
Linked List

- **Linked List**: A commonly used linear data structure, consisting a group of nodes in a sequence.
- Each node holds its own data and the address of the next node, thus forming a chain-like structure.
- 3 different implementations of Linked List:
  - Singly Linked List
  - Doubly Linked List
  - Circular Linked List



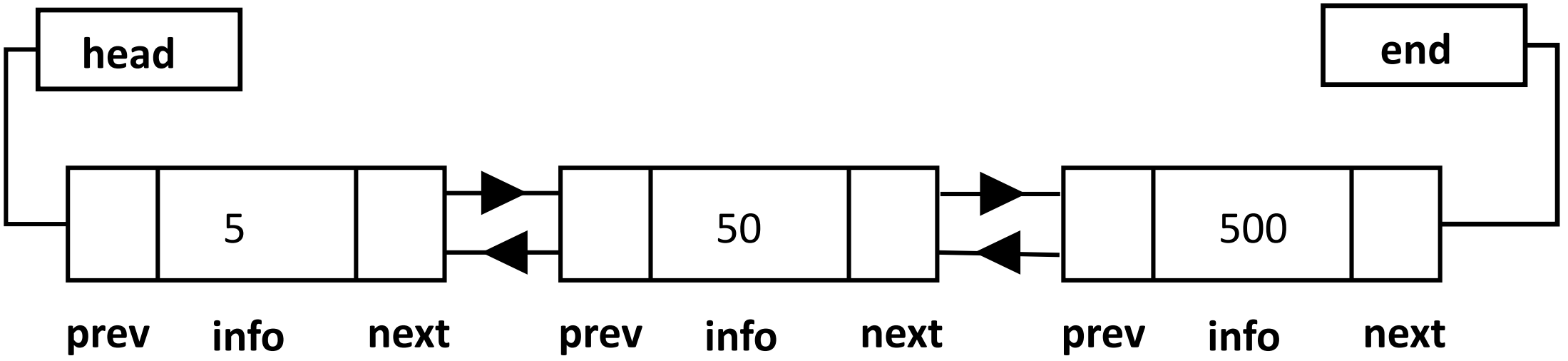
Singly Linked List

- Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in the sequence of nodes.
- Operations performed: insertion, deletion and traversal.



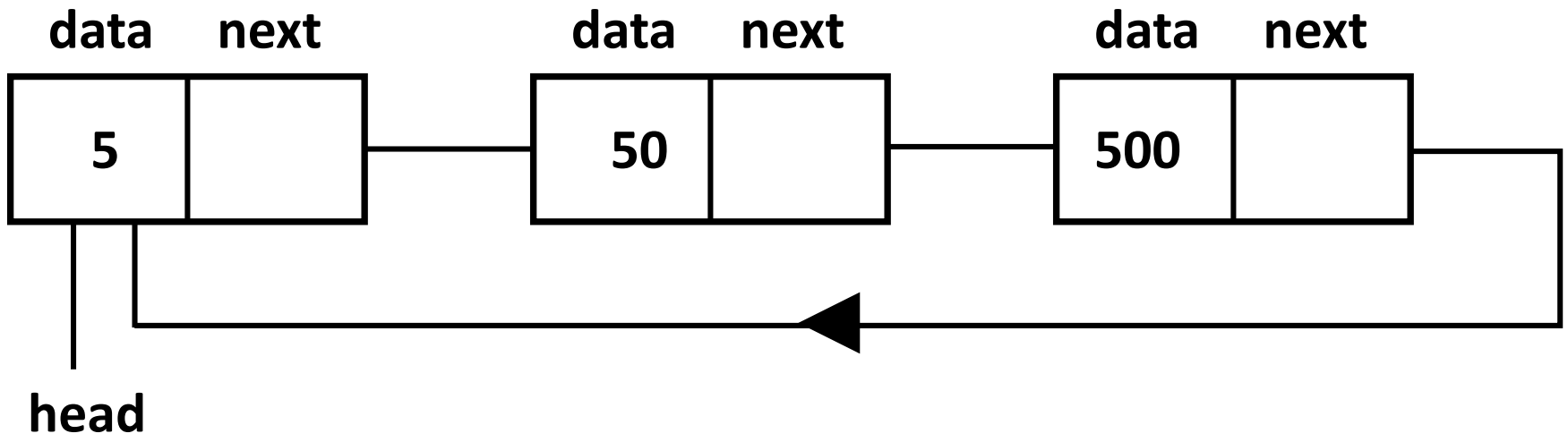
Doubly Linked List

In a doubly linked list, each node contains a **data** part and two addresses, one for the **previous** node and one for the **next** node.



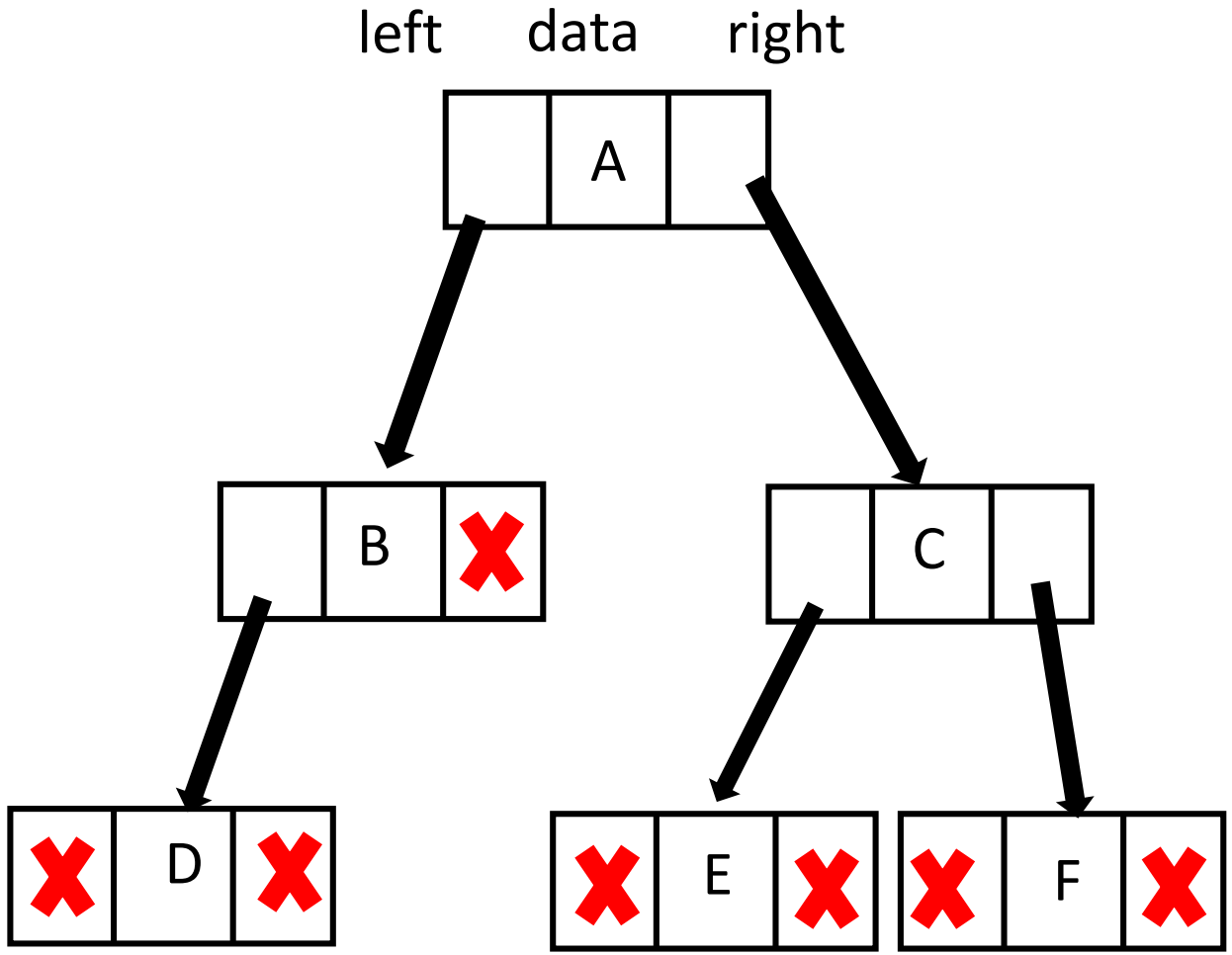
Circular Linked List

In circular linked list the last node of the list holds the address of the first node hence forming a circular chain.

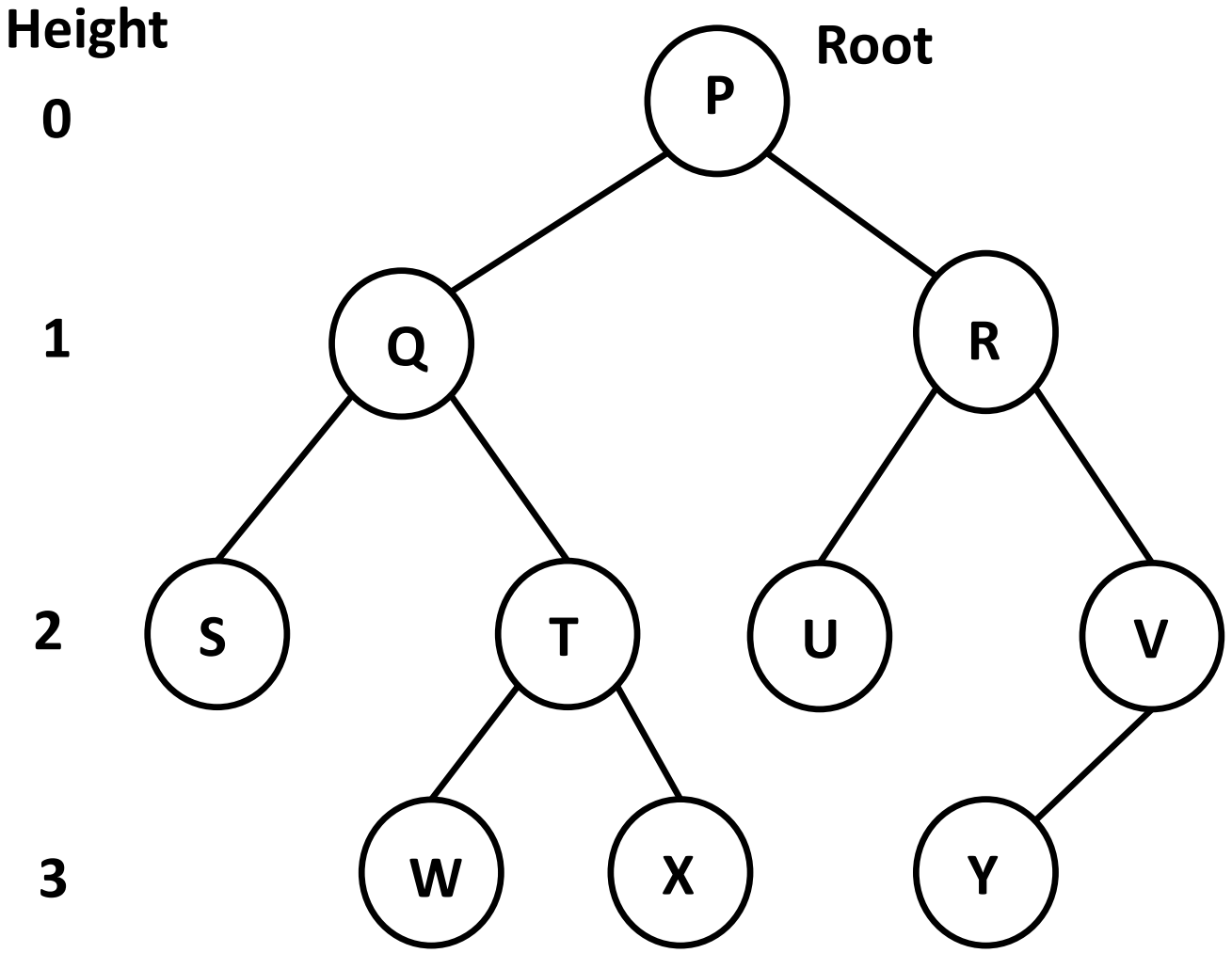


Binary Tree

- A hierarchical data structure in which each node has at most two children
- Referred as **left child** and **right child**.
- Each node contains three components:
  - Pointer to left subtree
  - Pointer to right subtree
  - Information element
- Topmost node in the tree: Root



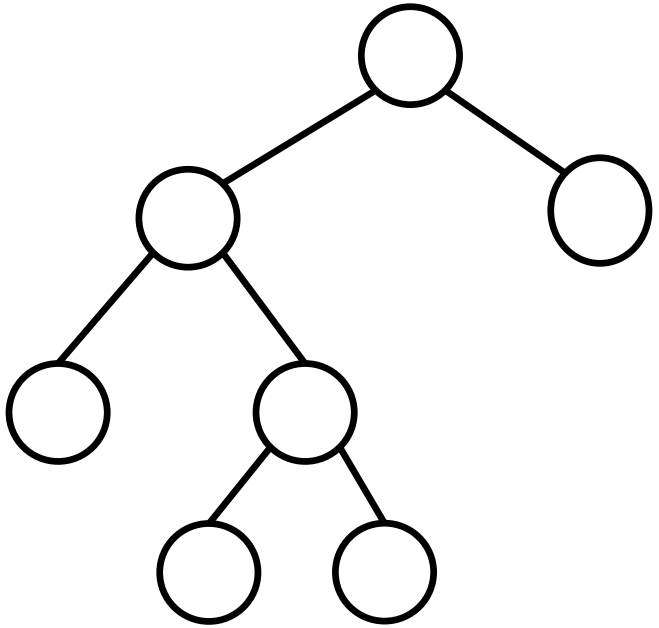
Binary Tree: Common Terminologies



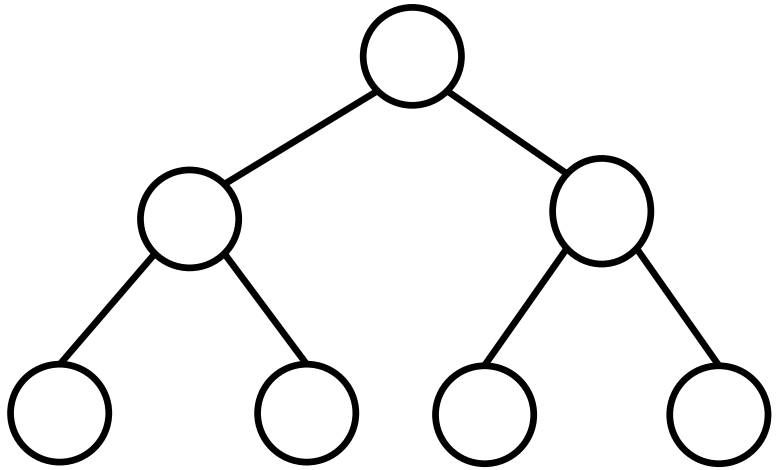
- Root: P
- Nodes: **10**
- Height of Tree: **3**
- **P** is the parent of **Q** and **R**
- **W** and **X** are children of **T**

Different Types of Binary Trees

1. **Rooted Binary Tree**: It has a root node and every node has atmost two children

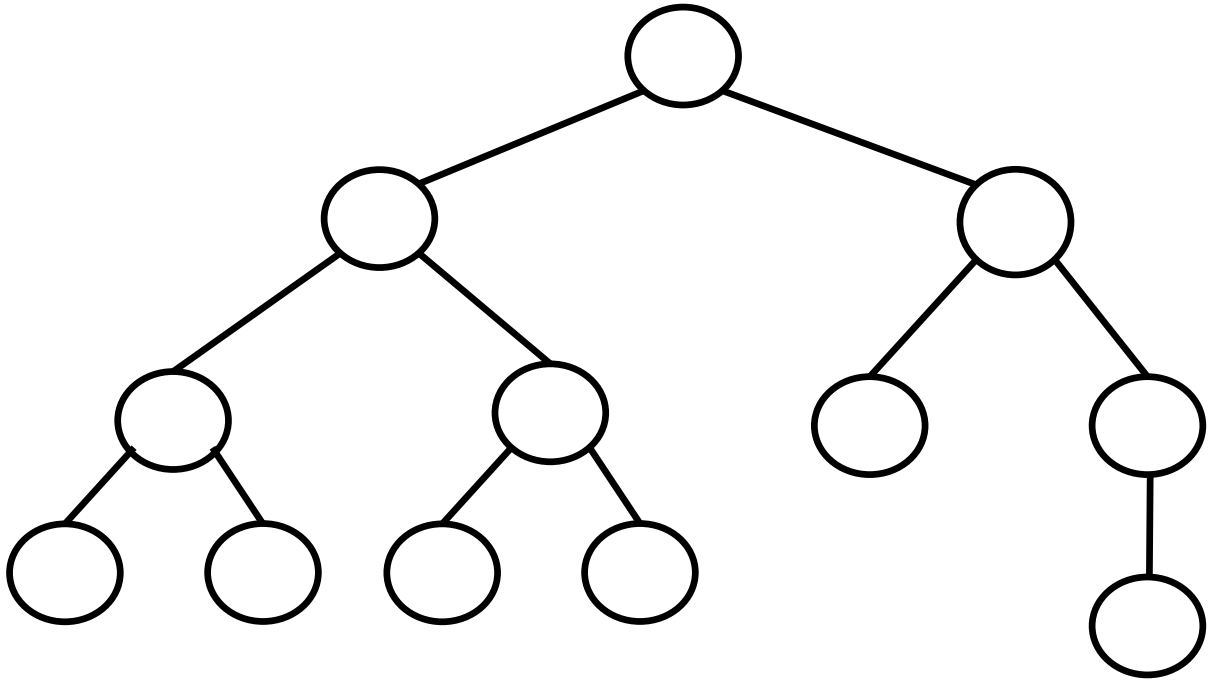


2. Full Binary Tree

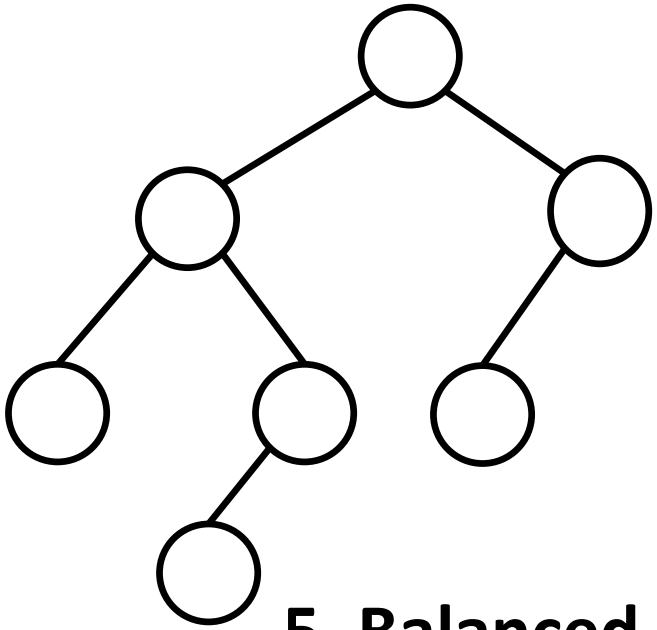


3. Perfect Binary Tree

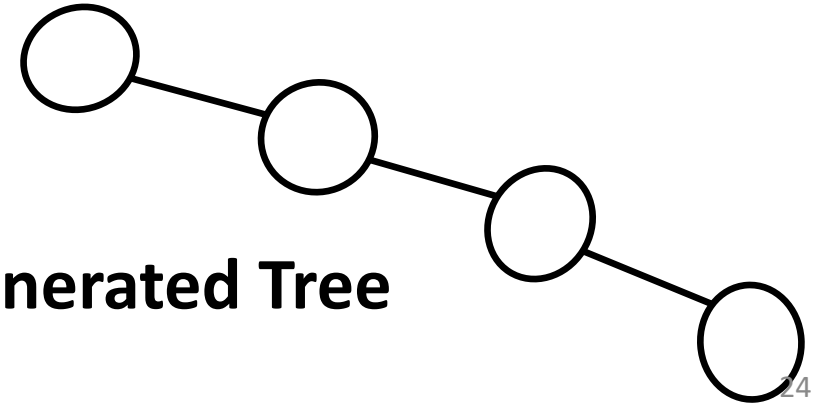
Types of Binary Trees



4. Complete Binary Tree



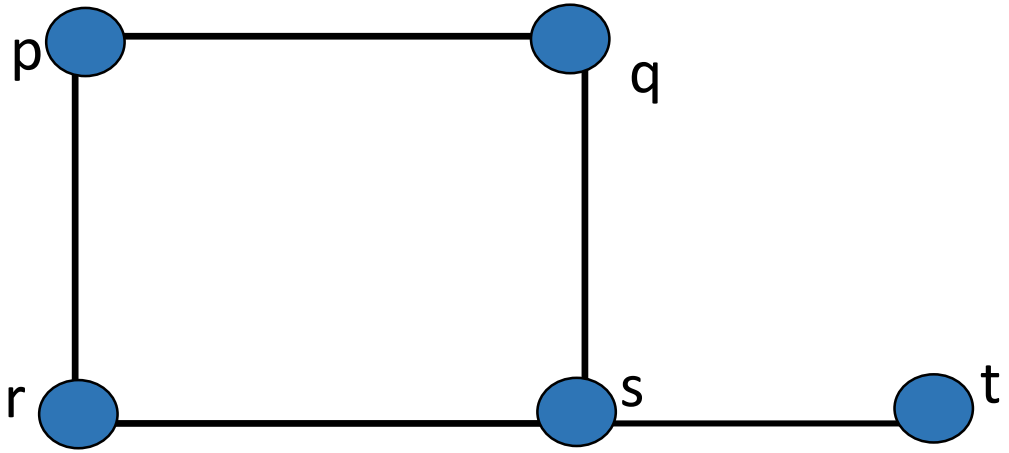
5. Balanced Binary Tree



6. Degenerated Tree

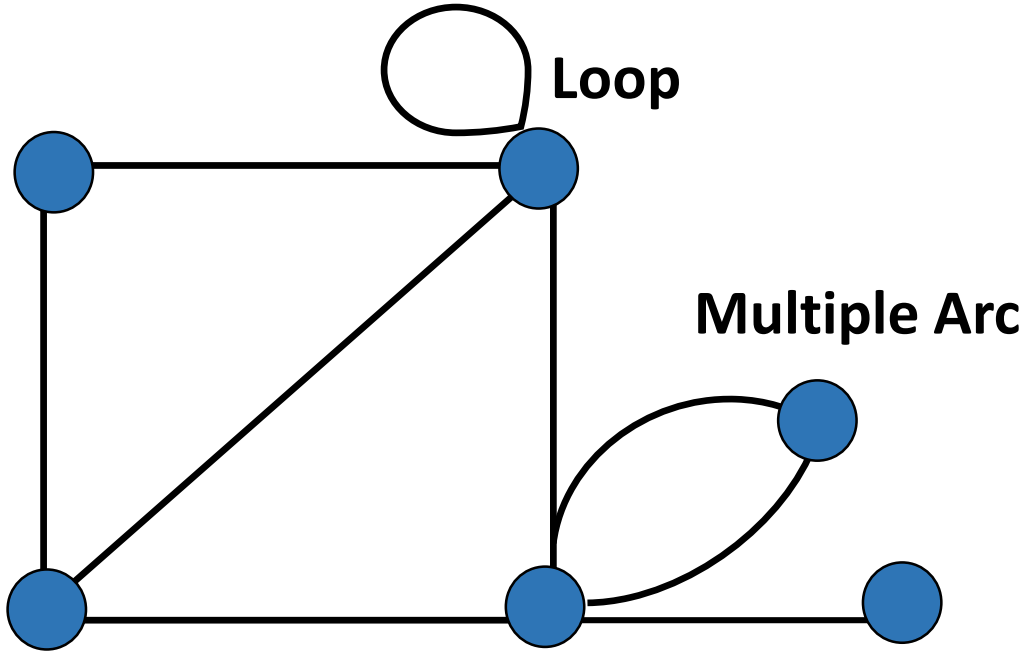
Graph

- A pictorial representation of a set of objects, where some pairs of objects are connected by links.
- The interconnected objects are represented by points termed as ***vertices (V)***, and the links that connect the vertices are called ***edges (E)***.
- **Graph**: A pair of sets ***(V, E)***, where ***V*** is set of vertices and ***E*** is set of edges, connecting the pairs of vertices.

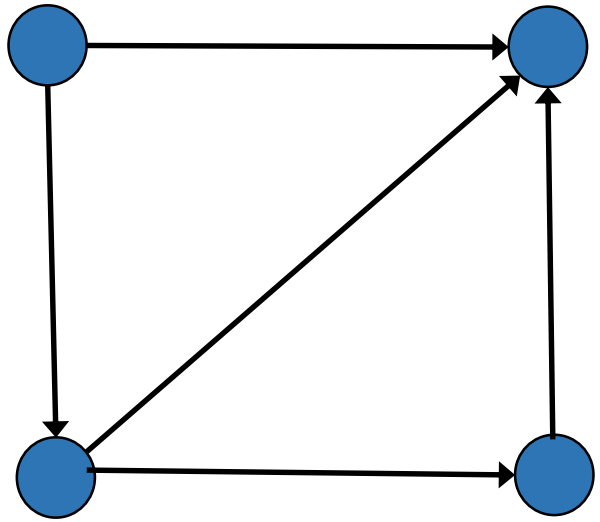


V = {p, q, r, s, t}
E = {pq, pr, rs, qs, st}

Types of Graph (1/2)

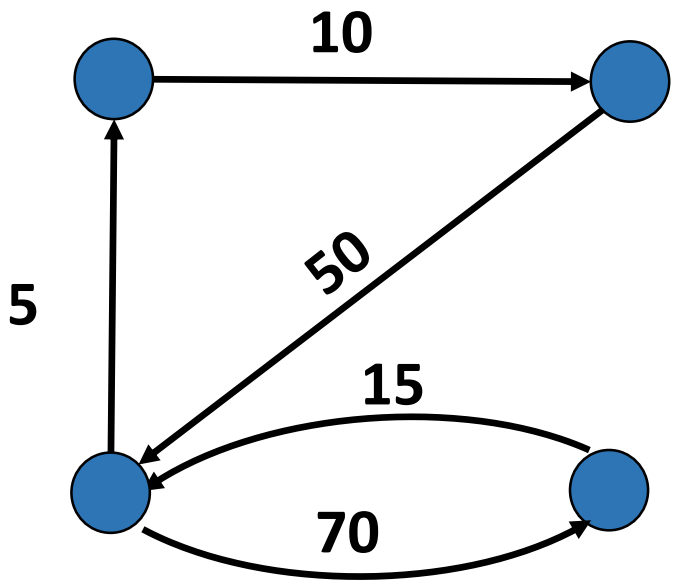


Simple Graph: undirected graphs

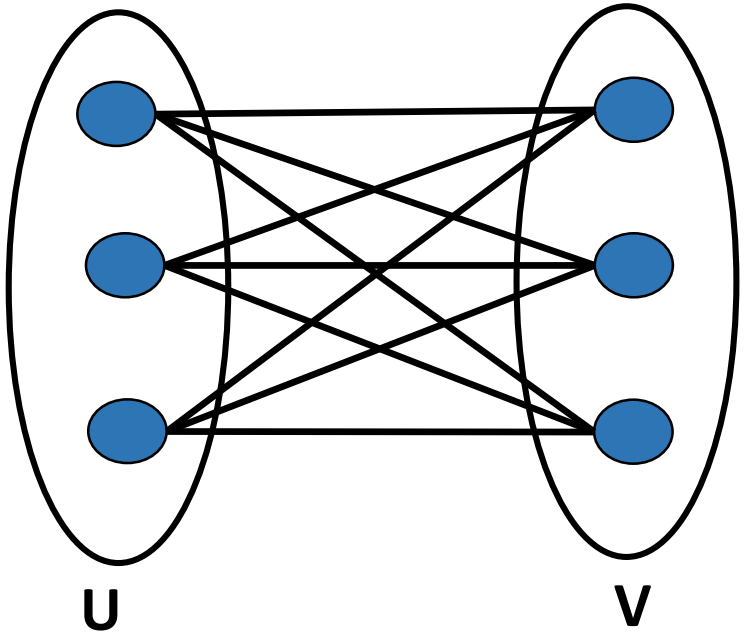


Directed Graph: graphs with edges having direction

Types of Graph (2/2)



Weighted Graph: edges have weights



Bipartite graph: vertices can be divided into two disjoint, independent sets **U** and **V,** such that every edge connects a vertex in **U** to one in **V**.

Applications of Data Structures

- **Queues**
  - Hardware queues
  - Network routers
  - Operating systems - CPU Scheduling
  - Airport take-off
- **Stacks**
  - Undo mechanisms
  - Recursion/function calling
  - Expression conversion

- **Linked Lists**
  - Making any list
  - Basic of dynamic memory allocation
  - Efficient memory management
- **Trees**
  - Searching
  - Time, geography, ancestry
- **Graphs**
  - Almost any complex problem
  - Network nodes

Thank You!