Syllabus

Unit	Contents	Hrs
I	Introduction –Common operations on data structures, Types of data structures, Data structures & Programming, Program Design, Complexities, Time Complexity, order of Growth, Asymptotic Notation. Sorting and Searching Introduction, Sorting, Insertion Sort, Selection Sort, Merging, Merge-Sort, Shell Sort, Radix Sort, Searching and Data Modification, Hashing	9
II	Arrays: Introduction, Linear Arrays, Arrays as ADT, Representation of Linear array in Memory, Traversing Linear Arrays, Inserting and deleting, Sorting; Bubble Sort, Searching; Linear Search, Binary Search, : Linked List Introduction: Linked Lists, Representation of Linked Lists in Memory, Traversing a Linked List, Searching a Linked List, Memory Allocation; Garbage Collection, Insertion into a Linked List, Deletion from a Linked List, Header Linked List, Circularly Linked Lists, Two-Way Lists (or Doubly Linked Lists).	9
III	Stacks, Queue and Recursion- Introduction, Stacks ,Array Representation of Stacks ,Linked Representation of Stacks, Stack as ADT, Arithmetic Expression; Polish Notation, Application of Stacks, Recursion, Towers of Hanoi, Implementation of Recursive Procedures by Stacks, Queue, Linked Representation of Queues, Queues as ADT, Circular Queues, Deques, Priority Queues, Applications of Queues	9

Syllabus

Unit	Contents	Hours
IV	Trees and Binary Trees -Binary Trees • Representation, Operations: Insert, Delete, Traversal: Preorder, Inorder, Postorder,Traversal Algorithms Using Stacks,Header Nodes; Threads,Threaded Binary Trees,Binary Search Trees ,Searching and Inserting in Binary Search Trees,Deleting in a Binary Search Tree,Balanced Binary Trees,AVL Search Trees,Insertion in an AVL Search Tree, Deletion in an AVL Search Tree,m-way Search Trees ,Searching, Insertion and Deletion in an m-way Search tree,B-Trees ,Searching, Insertion and Deletion in a B-tree,B+-Trees Graph Algorithms	10
V	Graphs and their Applications -) Introduction, Graph Theory Terminology, Sequential Representation of Graphs, Adjacency Matrix; Path Matrix, Linked Representation of a Graph, Operations on Graphs, Traversing a Graph,Posets; Topological Sorting,Spanning Trees	8

Data Structure

- Way of storing data in computer's memory so that it can be used easily and efficiently.
- There are different data-structures used for the storage of data.
- Also define as a mathematical or logical model of a particular organization of data items.
- The representation of particular data structure in the main memory of a computer is called as storage structure.
- **For Examples:** Array, Stack, Queue, Tree, Graph, etc.

Common operations on data structure

- **Insertion:** Insertion means to add an element in the given data structure.
- **Searching:** Searching means to find a particular element in the given data-structure
- **Deletion:** Deletion means to delete an element in the given data structure
- **Traversing:** Traversing a Data Structure means to visit the element stored in it.
- **Sorting:** To sort in particular order

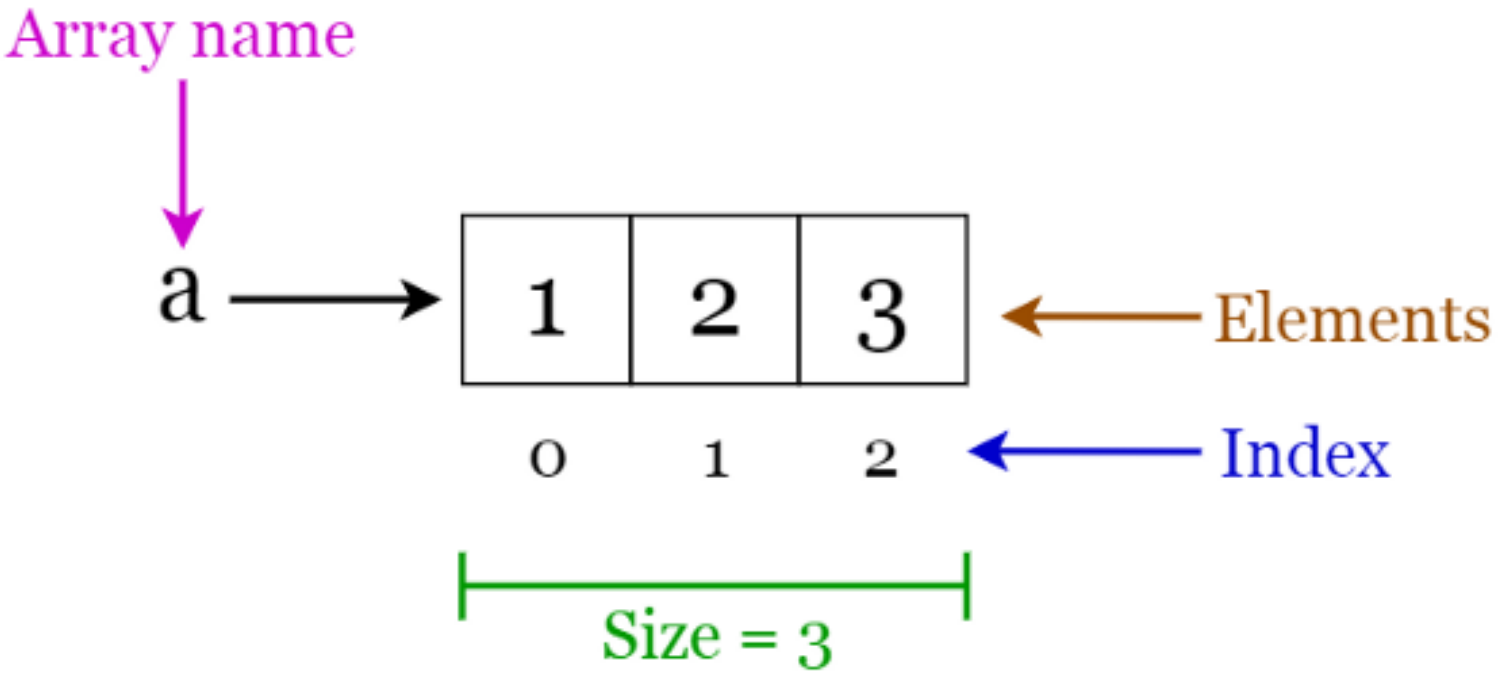
Types of data structures

- Arrays
- Linked Lists
- Stacks
- Queues
- Hash Tables
- Trees
- Graphs

Arrays

- An **array** is a structure of fixed-size, which can hold items of the same data type.
- It can be an array of integers, floating-point numbers, strings or even an array of arrays (such as *2-dimensional arrays*).
- Arrays are indexed
- Random access

Arrays



int a[3]; a=100+2

Array operations

- **Traverse**: Go through the elements and print them.
- **Search**: Search for an element in the array. You can search the element by its value or its index
- **Update**: Update the value of an existing element at a given index

Applications of arrays

- Used as the building blocks to build other data structures such as array lists, heaps, hash tables, vectors and matrices.
- Used for different sorting algorithms such as insertion sort, quick sort, bubble sort and merge sort.

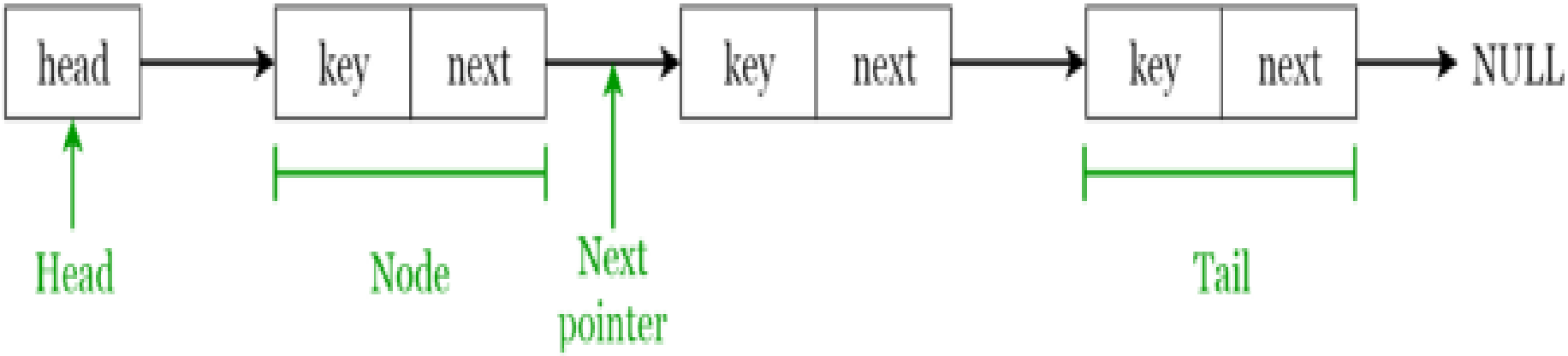
Linked List

- A **linked list** is a sequential structure that consists of a sequence of items in linear order which are linked to each other.
- Hence, you have to access data sequentially and random access is not possible.
- Linked lists provide a simple and flexible representation of dynamic sets.

Linked List

- Elements in a linked list are known as **nodes**.
- Each node contains a **key** and a pointer to its successor node, known as **next**.
- The attribute named **head** points to the first element of the linked list.
- The last element of the linked list is known as the **tail**.

Linked List


head
key
next
key
next
key
next
NULL
Head
Node
Next
pointer
Tail

Linked List

- **Singly linked list** — Traversal of items can be done in the forward direction only.
- **Doubly linked list** — Traversal of items can be done in both forward and backward directions. Nodes consist of an additional pointer known as **prev**, pointing to the previous node.
- **Circular linked lists** — Linked lists where the prev pointer of the head points to the tail and the next pointer of the tail points to the head.

Linked List Operations

- **Search**: Find the first element with the key **k** in the given linked list by a simple linear search and returns a pointer to this element
- **Insert**: Insert a key to the linked list. insert at the beginning of the list, insert at the end of the list and insert in the middle of the list.
- **Delete**: Removes an element **x** from a given linked list. Delete from the beginning of the list, delete from the end of the list and delete from the middle of the list.

Applications of linked lists

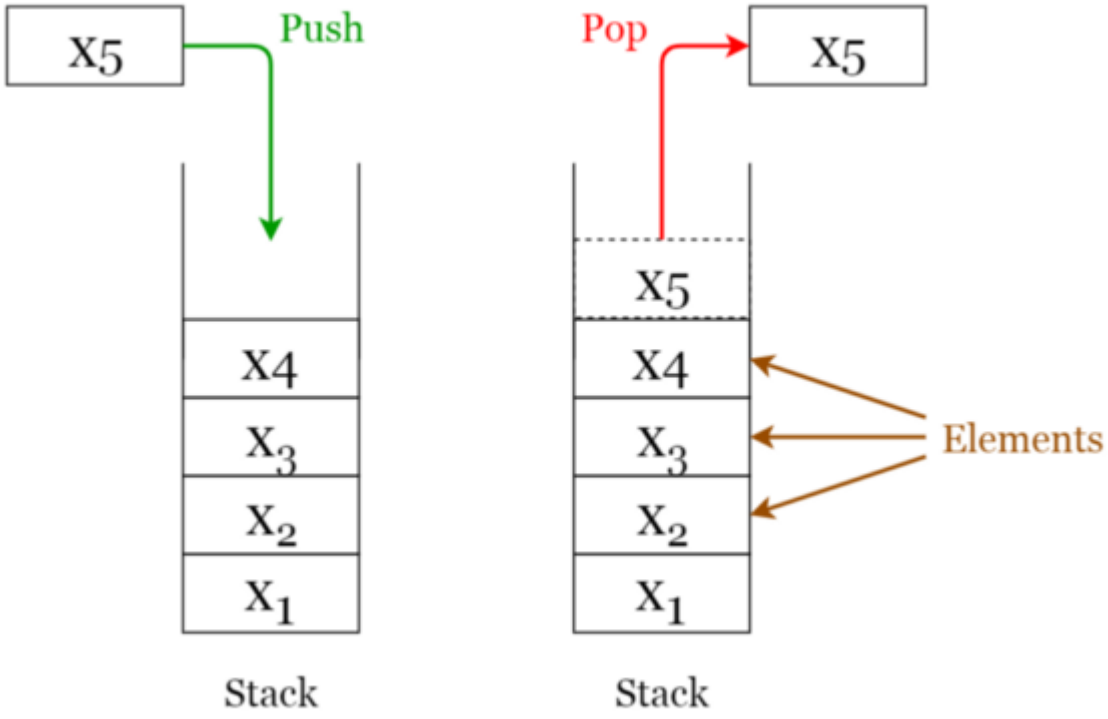
- Used for *symbol table management* in compiler design.
- Used in switching between programs using Alt + Tab (implemented using Circular Linked List).

Stacks

- A **stack** is a **LIFO** (Last In First Out — the element placed at last can be accessed at first) structure

Stack operations

- **Push**: Insert an element on to the top of the stack.
- **Pop**: Delete the topmost element and return it.



Applications of stacks

- Used for expression evaluation (e.g.: *shunting-yard algorithm* for parsing and evaluating mathematical expressions).
- Used to implement function calls in recursion programming.

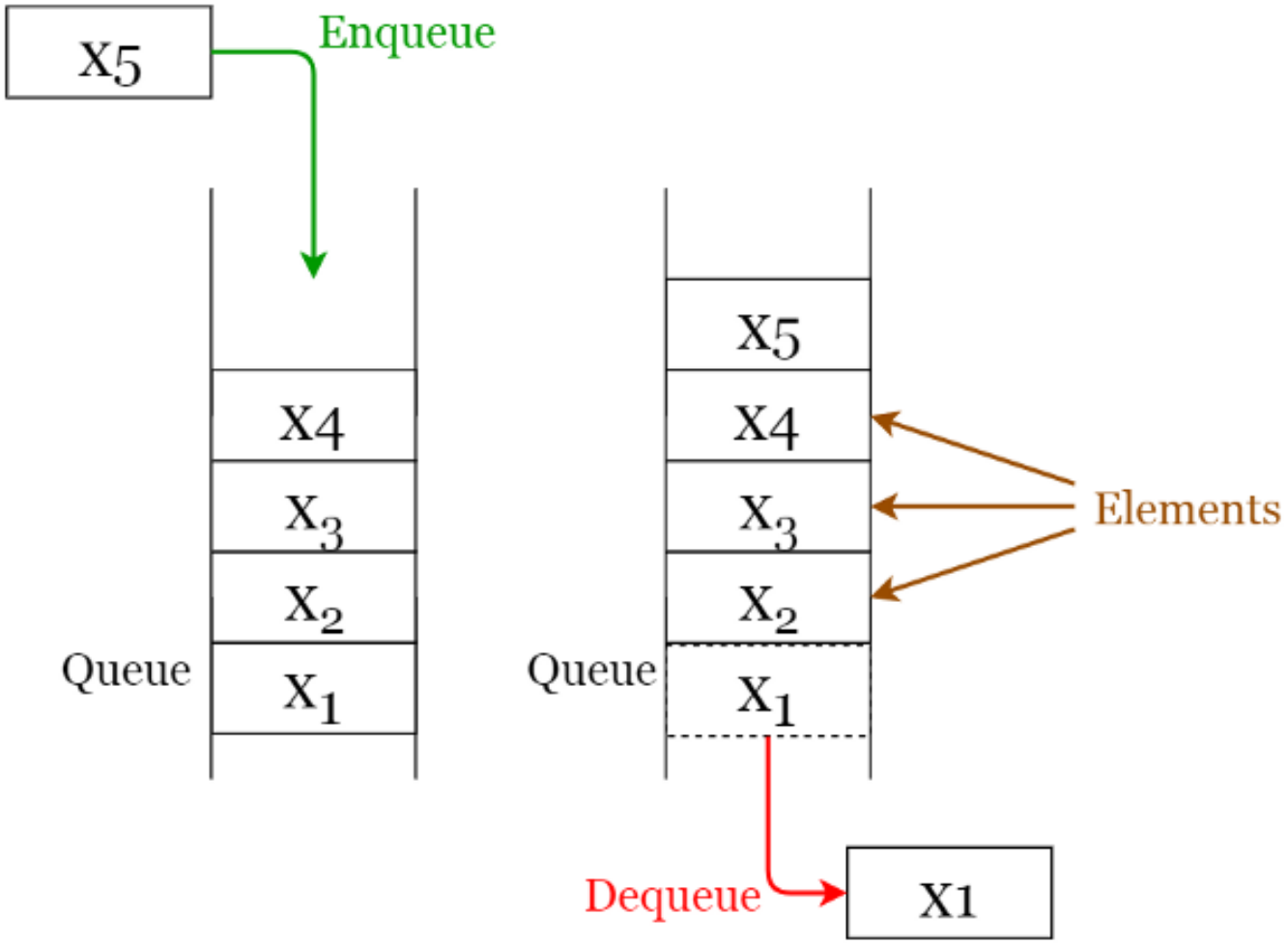
Queues

- A **queue** is a **FIFO** (First In First Out — the element placed at first can be accessed at first) structure

Queue operations

- **Enqueue**: Insert an element to the end of the queue.—rear end
- **Dequeue**: Delete the element from the beginning of the queue—front end

Queue operations



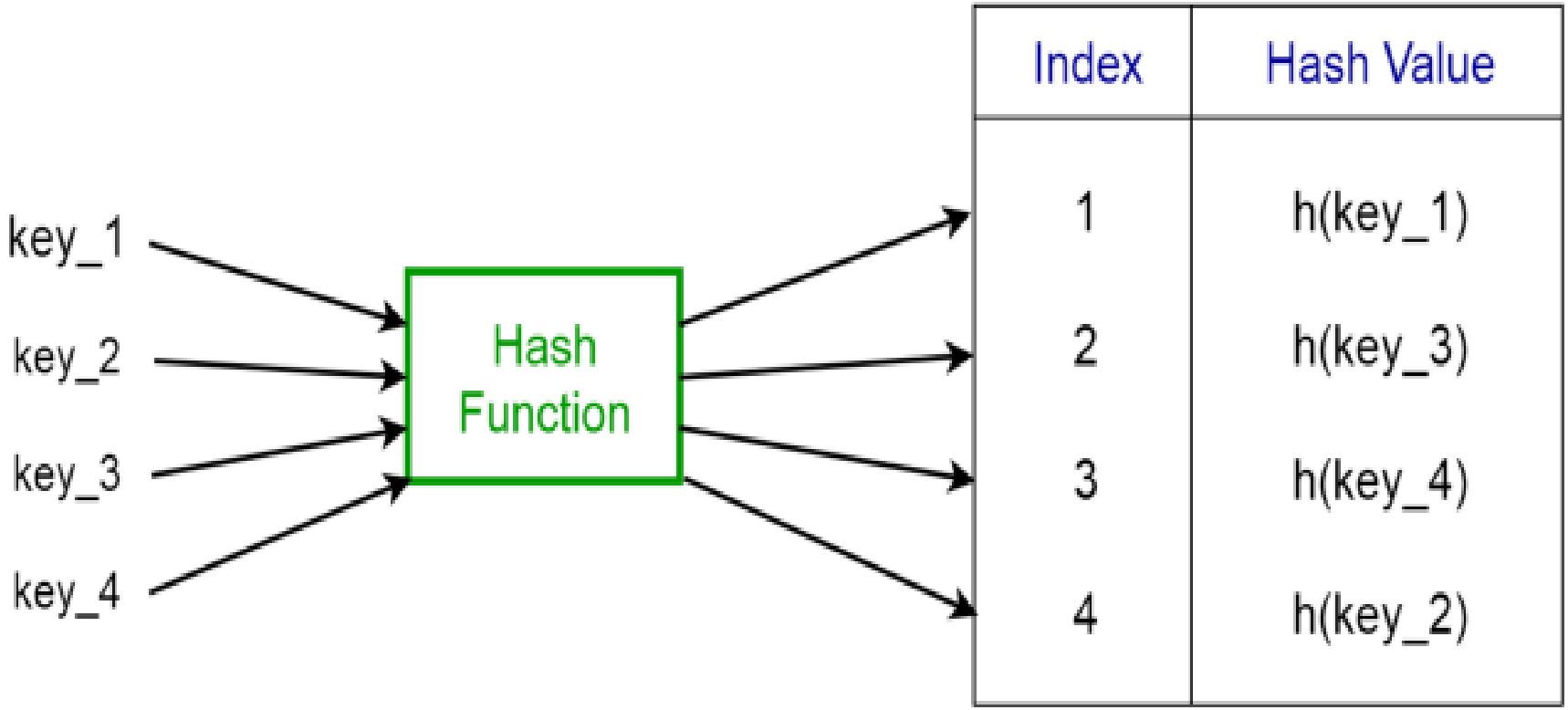
Applications of queues

- Used to manage threads in multithreading.
- Used to implement queuing systems (e.g.: priority queues).

Hash Tables

- A **Hash Table** is a data structure that stores values which have keys associated with each of them.
- It supports lookup efficiently if we know the key associated with the value.
- Hence it is very efficient in inserting and searching, irrespective of the size of the data.

Hash Tables

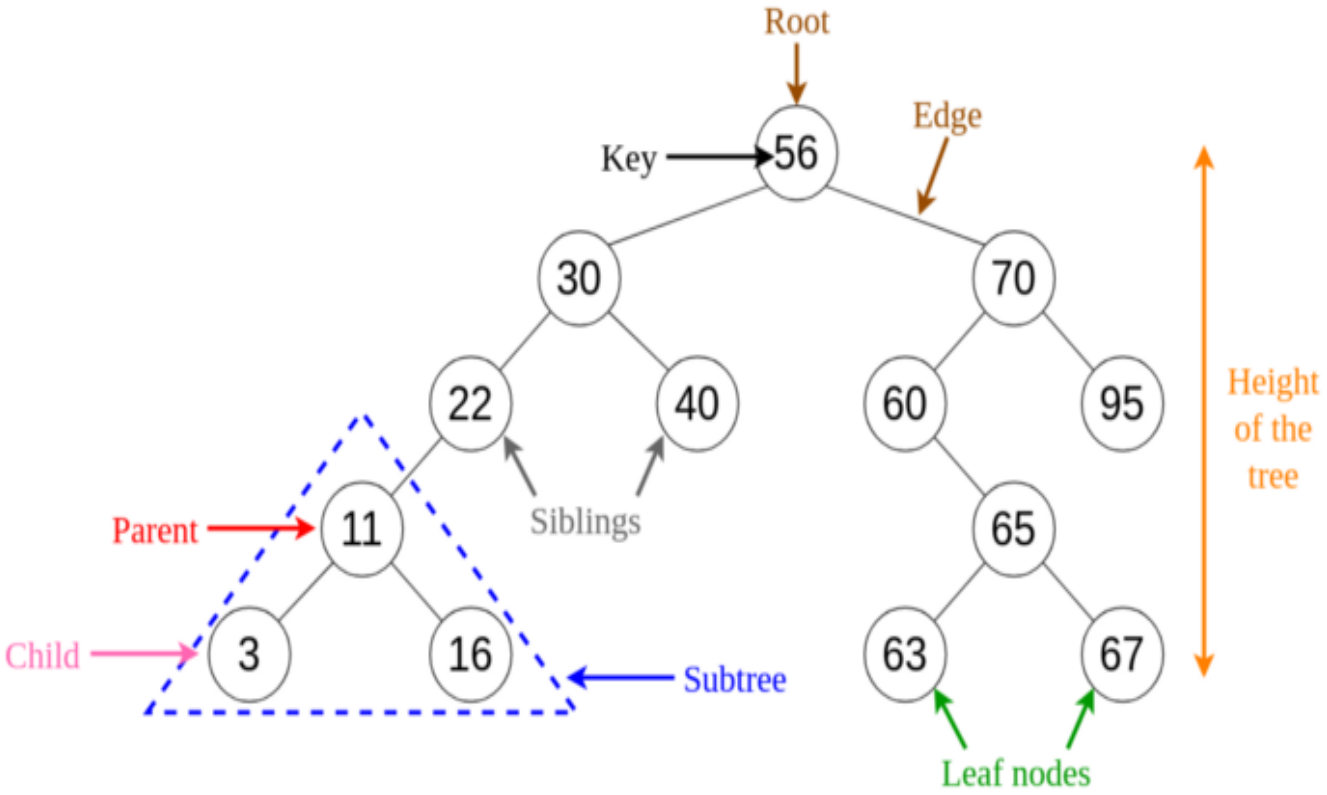

key_1
key_2
key_3
key_4
Hash Function
Index
Hash Value
1
h(key_1)
2
h(key_3)
3
h(key_4)
4
h(key_2)

Applications of hash tables

- Used to implement database indexes.
- Used to implement associative arrays.
- Used to implement the “set” data structure.

Trees

- A **tree** is a hierarchical structure where data is organized hierarchically and are linked together.

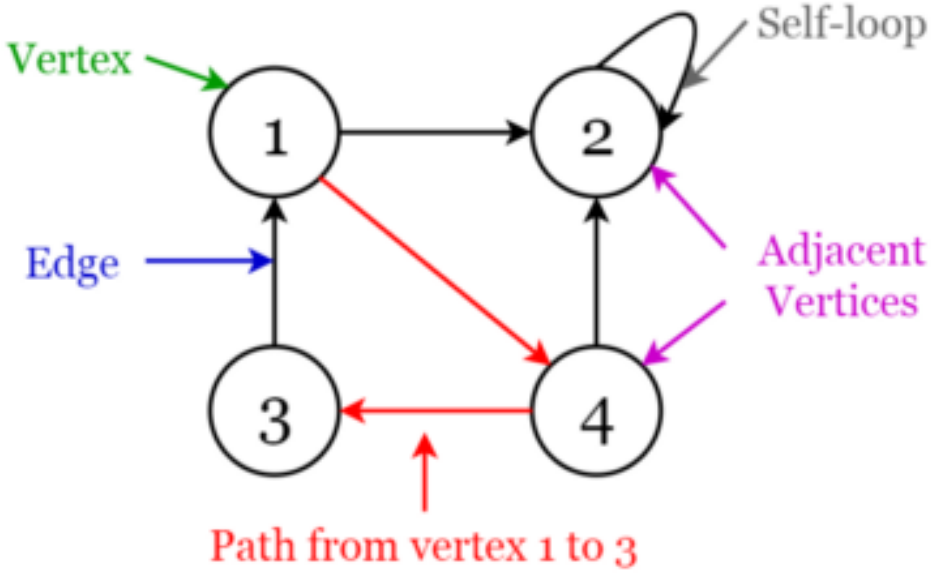


Applications of trees

- **Binary Trees**: Used to implement expression parsers and expression solvers.
- **Binary Search Tree**: used in many search applications where data are constantly entering and leaving.
- **Heaps**: used by JVM (Java Virtual Machine) to store Java objects.
- **Treaps**: used in wireless networking.

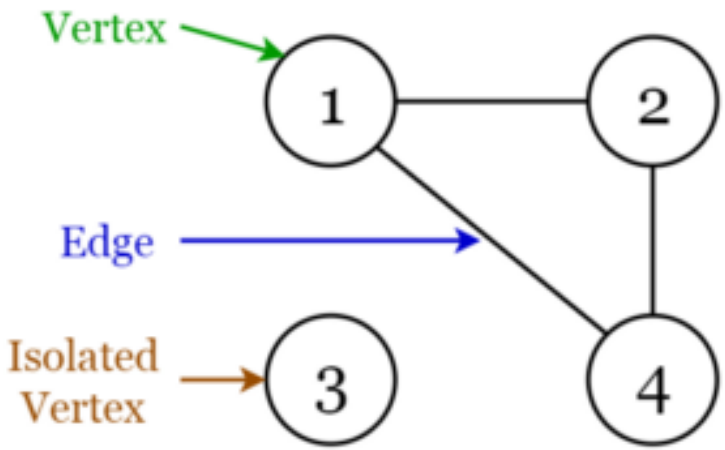
Graphs

- A **graph** consists of a finite set of **vertices** or nodes and a set of **edges** connecting these vertices.
- The **order** of a graph is the number of vertices in the graph.
- The **size** of a graph is the number of edges in the graph.
- Two nodes are said to be **adjacent** if they are connected to each other by the same edge.



Directed Graph

G = {1, 2, 3, 4}
E = {(1, 2), (1, 4), (2, 2), (3, 1), (4, 3), (4, 2)}



Undirected Graph

G = {1, 2, 3, 4}
E = {(1, 2), (1, 4), (2, 4)}

Applications of graphs

- Used to represent social media networks. Each user is a vertex, and when users connect they create an edge.
- Used to represent web pages and links by search engines. Web pages on the internet are linked to each other by hyperlinks. Each page is a vertex and the hyperlink between two pages is an edge. Used for Page Ranking in Google.
- Used to represent locations and routes in GPS. Locations are vertices and the routes connecting locations are edges. Used to calculate the shortest route between two locations.

Data structures & Programming,

- Data structures and algorithms play a major role in implementing software
- The data structure and algorithm provide a set of techniques to the programmer for handling the data efficiently.
- The programmer should understand the core concepts of data handling.

Program Design

- Solving Real-World Problems
- A data structure is a specialized format for organizing, processing, retrieving and storing data.
- There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose.
- Data structures make it easy for users to access and work with the data they need in appropriate ways.
- Most importantly, data structures frame the organization of information so that machines and humans can better understand it.