### **DATA STRUCTURES**

III Semester: CSE / IT / ECE / ME / CE | IV Semester AE / EEE

Course Code	Category	Hours / Week			Credits	Maximum Marks		
ACSB03	Core	L	T	P	C	CIA	SEE	Total
		3	0	0	3	30	70	100
Contact Classes: 45	<b>Tutorial Classes: Nil</b>	Pı	ractical	l Classe	s: Nil	Total Classes: 60		

### **OBJECTIVES:**

### The students will try to learn:

- 1. To provide students with skills needed to understand and analyze performance trade-offs of different algorithms / implementations and asymptotic analysis of their running time and memory usage.
- 2. To provide knowledge of basic abstract data types (ADT) and associated algorithms: stacks, queues, lists, tree, graphs, hashing and sorting, selection and searching.
- 3. The fundamentals of how to store, retrieve, and process data efficiently.
- 4. To provide practice by specifying and implementing these data structures and algorithms in Python.
- 5. Understand essential for future programming and software engineering courses.

### **COURSE OUTCOMES:**

### After successful completion of the course, students will be able to:

- 1. **Carryout** the analysis of a range of algorithms in terms of algorithm analysis and express algorithm complex it using the O notation (**Understand**).
- 2. Make use of recursive algorithm design technique in appropriate contexts (Apply).
- 3. **Represent** standard ADTs by means of appropriate data structures (**Understand**).
- 4. **Select** appropriate sorting technique for given problem (**Understand**).
- 5. **Select** appropriate searching technique for given problem (**Understand**).
- 6. **Implement** standard searching and sorting algorithms; including binary search; merge sort and quick sort; and their complexities (**Apply**).
- 7. Design and **implement** linked lists, stacks and queues in Python (**Apply**).
- 8. **Explain** the use of basic data structures such as arrays, stacks, queues and linked lists in program design(**Understand**).
- 9. **Extend** their knowledge of data structures to more sophisticated data structures to solve problems involving balanced binary search trees, AVL Trees, B-trees and B+ trees, hashing, and basic graphs.
- 10.**Design** and implement tree structures in Python(**Apply**).
- 11. **Compare** and contrast the benefits of dynamic and static data structures implementations and choose appropriate data structure for specified problem domain(**Understand**).
- 12. Quickly **determine and explain** how efficient an algorithm or data structure will be, apply appropriate data structures for solving computing problems with respect to performance (**Analyze**).

# MODULE-I INTRODUCTION TO DATA STRUCTURES, SEARCHING AND SORTING

Basic concepts: Introduction to data structures, classification of data structures, operations on data structures; Algorithm Specification, Recursive algorithms, Data Abstraction, Performance analysis- time complexity and space complexity, Asymptotic Notation-Big O, Omega, and Theta notations. Introduction

to Linear and Non Linear data structures, Searching techniques: Linear and Binary search; Sorting techniques: Bubble, Selection, Insertion, Quick and Heap Sort and comparison of sorting algorithms.

## MODULE-II LINEAR DATA STRUCTURES

Stacks: Stack ADT, definition and operations, Implementations of stacks using array, applications of stacks, Arithmetic expression conversion and evaluation; Queues: Primitive operations; Implementation of queues using Arrays, applications of linear queue, circular queue and double ended queue (deque).

## MODULE-III LINKED LISTS

Linked lists: Introduction, singly linked list, representation of a linked list in memory, operations on a single linked list; Applications of linked lists: Polynomial representation and sparse matrix manipulation.

Types of linked lists: Circular linked lists, doubly linked lists; Linked list representation and operations of Stack, linked list representation and operations of queue.

# MODULE-IV NON LINEAR DATA STRUCTURES

Trees: Basic concept, binary tree, binary tree representation, array and linked representations, binary tree traversal, binary tree variants, threaded binary trees, application of trees, Graphs: Basic concept, graph terminology, Graph Representations - Adjacency matrix, Adjacency lists, graph implementation, Graph traversals – BFS, DFS, Application of graphs, Minimum spanning trees – Prims and Kruskal algorithms.

# MODULE-V BINARY TREES AND HASHING

Binary search trees: Binary search trees, properties and operations; Balanced search trees: AVL trees; Introduction to M-Way search trees, B trees; Hashing and collision: Introduction, hash tables, hash functions, collisions, applications of hashing.

### **Text Books:**

- 1. Rance D. Necaise, "Data Structures and Algorithms using Python", Wiley StudentEdition.
- 2. Benjamin Baka, David Julian, "Python Data Structures and Algorithms", Packt Publishers, 2017.

### **Reference Books:**

- 1. S. Lipschutz, "Data Structures", Tata McGraw Hill Education, 1st Edition, 2008.
- 2. D. Samanta, "Classic Data Structures", PHI Learning, 2<sup>nd</sup> Edition, 2004.

### **Web References:**

- 1. https://www.tutorialspoint.com/data\_structures\_algorithms/algorithms\_basics.htm
- 2. https://www.codechef.com/certification/data-structures-and-algorithms/prepare
- 3. https://www.cs.auckland.ac.nz/software/AlgAnim/dsToC.html
- 4. https://online-learning.harvard.edu/course/data-structures-and-algorithms