Semester: V	
Programme: B.Sc. Computer Science (Hons) (Minor – Artificial Intelligence)	
Course : DATA STRUCTURES	
Paper code: B3CS230512T / B3CS230512P	**Credits: 4**
Hours/week : Theory: 3 / Practical 2	
Category: Core/MDC/SEC/VAC/Minor : Minor	
Theory / Practical / Composite : Composite	
No of Modules : 1	

Course Overview: The course introduces fundamental concepts of algorithm analysis, abstract data types (ADTs), and core linear and non-linear data structures. Students explore time and space complexity, searching and sorting algorithms (linear search, binary search, bubble sort, selection sort, insertion sort, quick sort, merge sort), and implementation of key data structures such as stacks, queues, linked lists, binary trees, and binary search trees (BST). Practical sessions are conducted using Python to reinforce theoretical understanding through hands-on coding. The course aims to build a strong foundation for efficient problem solving and algorithm design—essential for advanced studies in artificial intelligence, machine learning, and software engineering.

Course Outcome:

1. **Recall and explain** Recall and explain fundamental concepts of algorithms, including time and space complexity (Big-O notation), and classify algorithms based on their efficiency in best, average, and worst-case scenarios.
2. **Apply** the concept of Abstract Data Types (ADTs) to model real-world problems by defining interfaces and selecting appropriate data representations for algorithmic solutions.
3. **Analyze** and implement linear and binary search techniques to efficiently locate elements in both ordered and unordered datasets, and justify their suitability based on data characteristics.
4. **Evaluate** and compare various sorting algorithms—Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, and Merge Sort—in terms of their time and space complexities across different input conditions.
5. **Design** and implement linear data structures such as stacks, queues, and singly linked lists using Python, demonstrating dynamic memory allocation and operations like insertion, deletion, and traversal.
6. **Create** and manipulate non-linear hierarchical data structures—specifically Binary Trees and Binary Search Trees (BST)—to support efficient data organization, searching, insertion, and deletion in applications relevant to artificial intelligence and real-world problem-solving.

Prerequisites:

1. Proficiency in a Programming Language
2. Basic Understanding of Algorithms and Problem-Solving Techniques
3. Knowledge of Fundamental Mathematics
4. Logical and Analytical Reasoning Skills

SYLLABUS

UNIT/Module	CONTENT	HOURS	CO Mapping	COGNITIVE LEVEL
I.	Algorithm fundamentals; Introduction to time and space complexity (Big-O notation, best/average/worst cases)	6	CO1	K1, K2 (Remember/Understand)
II.	Abstract Data Type (ADT): Definition, role in modular programming, and interface specification.	5	CO2	K3 (Apply)
III.	Searching Algorithms: Linear search, Binary search	6	CO3, CO4	K4-K5 (Analyze/ Evaluate)

	Sorting Algorithms: Bubble sort, Selection sort, Insertion sort, Quick sort, Merge sort Specifiers. Decision Making Statement: if-else, switch-case, Ternary Operator.			
IV.	Linear Data Structures: Stack (LIFO), Queue (FIFO), Introduction to Singly Linked Lists—advantages over arrays, basic operations (insert, delete, and traverse).	7	CO5	K6 (Create)
V.	Non-linear Data Structures: Binary Tree (terminology, traversal methods), Binary Search Tree (BST)—search, insert, delete operations	7	CO6	K6 (Create)
VI.	Practical Implementation in Python: Hands-on coding of all data structures and algorithms covered in theory.	8	CO3-CO6	K4-K6 (Analyze/ Create)

Text Books		
1. Data Structures – Seymour Lipschutz, McGraw-Hill Education		
2. Data Structures and Algorithms in Python, by Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, WILEY		
3. Fundamentals of Data Structures, Ellis Horowitz		
Suggested readings		
Web Resources		
NPTEL: Data Structures and Algorithms https://nptel.ac.in/courses/106102064		
Evaluation	Theory CIA: 12 Attendance: 3 Semester Exam: 45	Practical CA: 38 Attendance: 2
Paper Structure for Theory Semester Exam Module : Answer 3 out of 5 of 15 marks each		

Course outcomes (COs) and Cognitive Level Mapping

COs	**CO Description**	**Cognitive levels**
CO1	Recall and explain fundamental concepts of algorithms, including time and space complexity, and classify algorithms based on efficiency metrics.	K1-K2 (Remember/Understand)
CO2	Apply the concept of Abstract Data Types (ADTs) to model real-world problems and select appropriate data representations for algorithmic solutions.	K3 (Apply)
CO3	Analyze and implement linear and binary search techniques to efficiently locate elements in ordered and unordered datasets.	K4 (Analyze)

CO4	Evaluate and compare various sorting algorithms (Bubble, Selection, Insertion, Quick, Merge Sort) based on best, worst, and average-case time complexities.	K5 (Evaluate)
CO5	Design and implement linear data structures—stacks, queues, and singly linked lists—in Python to solve computational problems with dynamic memory allocation.	K6 (Create)
CO6	Create and manipulate non-linear hierarchical data structures such as Binary Trees and Binary Search Trees (BST) to support efficient data organization and retrieval in AI-relevant applications.	K6 (Create)