Semester: 3				
Programme: Data Science				
Course: Data Structure and Analysis of Algorithms				
Paper code: C2DS250312T / C2DS250312P			**Credits: 4**	
Hours/week: 4 (3Th + 2Pr)				
Category: Core/MDC/SEC/VAC: Core				
Theory / Practical / Composite: Composite				
No of Modules: 1				
Course Outcome:				
1. **Remember** the fundamental definitions and properties of data structures, including abstract data types, arrays, linked lists, stacks, queues, and trees.				
2. **Understand** the principles of algorithm analysis, the significance of asymptotic notations, and the operational logic of iterative and recursive algorithms.				
3. **Apply** various searching and sorting techniques, such as binary search, merge sort, and quick sort, along with hashing and heap concepts to manage data.				
4. **Analyze** the efficiency and behavior of different algorithm design paradigms, including divide and conquer, greedy, dynamic programming, and backtracking.				
5. **Evaluate** the computational complexity of algorithms and the distinctions between complexity classes such as P, NP, NP-hard, and NP-complete to make informed technical decisions.				
6. **Create** efficient, integrated programs by selecting and implementing the most suitable data structures and algorithms to solve complex computational problems.				
SYLLABUS				
UNIT	**CONTENT**	**HOURS or NUMBER OF CLASSES**	**CO Mapping**	**COGNITIVE LEVEL**
1.	**Introduction to Data structures:** Abstract data types, Arrays, Linked Lists, Stack, Queues, Circular Queues	**8**	**CO1**	**K1**
2.	**Introduction to Algorithms:** Algorithms design principles, Analysing Algorithms – Time and space complexity, Iterative and recursive algorithms, Asymptotic notations and their significance	**4**	**CO2**	**K2**
3.	**Trees:** Binary trees, Traversal techniques, Binary search trees, Hashing, Concept of Heap.	**8**	**CO1, CO3**	**K1, K3**
4.	**Searching and Sorting:** Linear search, Binary search, Bubble sort, Insertion sort, selection sort, Merge sort, Quick sort, Worst and average computing complexity, Median and order statistics.	**10**	**CO3**	**K3**
5.	**Algorithm Design Paradigms:** Divide and conquer, Greedy, Dynamic programming and Backtracking with suitable examples.	**5**	**CO4**	**K4**

6.	**Computational Complexity classes:** Introduction to NP-completeness, P class, NP-hard class, NP complete class.	**4**	**CO5**	**K5**
Practical	Based on theory topics using Python		**CO6**	**K6**

Text Books

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction to Algorithms", MIT Press, 3rd Edition, 2009.
2. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, "Computer Algorithms", Silicon Press Publications, 2 nd Edition, 2008
3. Ellis Horowitz, Sartaj Sahni, Dinesh Mehta, "Fundamentals of Data Structures using C+ +", 2 nd Edition, Universities Press, 2008.
4. Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, "Data Structures and Algorithms", Pearson, 1 st Edition, 2006
5. S.Sridhar, "Design and Analysis of Algorithms", Oxford University Press, 2015

Evaluation	Theory CIA: 15 Sem Exam: 45	Practical CA: 40 Sem Exam: NA
Paper Structure for Theory Semester Exam Module:	Short questions (5 marks each)	Long questions (15 marks each)
	3 out of 5	2 out of 3

Course outcomes (COs) and Cognitive Level Mapping

COs	CO Description	Cognitive levels
CO1	**Remember** the fundamental definitions and properties of data structures, including abstract data types, arrays, linked lists, stacks, queues, and trees.	K1
CO2	**Understand** the principles of algorithm analysis, the significance of asymptotic notations, and the operational logic of iterative and recursive algorithms.	K2
CO3	**Apply** various searching and sorting techniques, such as binary search, merge sort, and quick sort, along with hashing and heap concepts to manage data.	K3
CO4	**Analyze** the efficiency and behavior of different algorithm design paradigms, including divide and conquer, greedy, dynamic programming, and backtracking.	K4
CO5	**Evaluate** the computational complexity of algorithms and the distinctions between complexity classes such as P, NP, NP-hard, and NP-complete to make informed technical decisions.	K5
CO6	**Create** efficient, integrated programs by selecting and implementing the most suitable data structures and algorithms to solve complex computational problems.	K6