Course code	Course Name	Hours/week			Credit	Max. Marks
		L	T	P	C	100
21BTCS23C02	Data Structures and Algorithms	3	0	1	4	
Pre-requisite	Mathematical Preliminaries					

Evaluation Scheme	Theory	Hours	Marks
	External (End Semester Exam)	2	50
	Internal (1) Midterm + Assignment/Seminar/Activity/ Quiz-40 Marks / (2) Attendance -10 Marks)	1.5	50

UNIT-I	Foundations of Algorithms	4

Basics of Algorithm, Characteristics of good algorithms
Introduction to Algorithm Analysis: Efficiency of Algorithms, Best, Worst, and Average Case, Amortized Analysis, Master Method

UNIT-II	Sorting Algorithms and Basic Analysis Techniques	9

Analyzing Control Statements, Loop Invariants and Correctness, Basic Sorting Algorithms and Their Analysis: Bubble Sort, Selection Sort, Insertion Sort, Heap Sort
Linear Time Sorting: Bucket Sort, Radix Sort, Counting Sort

UNIT-III	Divide and Conquer Techniques	10

Introduction to Recurrence Relations, Methods for Solving Recurrence,
Divide and Conquer Paradigm: Binary Search, Max-Min Problem, Merge Sort, Quick Sort, Matrix Multiplication, Large Integer Multiplication

UNIT-IV	Dynamic Programming and Greedy Algorithm	12

Principle of Optimality, Dynamic Programming Applications: Binomial Coefficient, Making Change, Assembly Line Scheduling, Matrix Chain Multiplication, Longest Common Subsequence, Knapsack Problem, All-Pairs Shortest Path
Greedy Algorithm Concepts, Greedy Applications: Activity Selection, Greedy Knapsack, Job Scheduling, Huffman Coding.

UNIT-V	Advanced Problem Solving and Complexity Theory	10

Backtracking: N-Queens Problem, Knapsack Problem, Travelling Salesman Problem (TSP), Minimax Principle
String Matching Algorithms: Naive Algorithm, Rabin-Karp, Finite Automata-based Matching
Introduction to Complexity Theory: P, NP, NP-Complete, NP-Hard, Complexity of TSP and Hamiltonian Path

	Total hours	45 periods

Course Outcomes: At the end of the course, the students will be able to:		
COs	Statements	Bloom's Level
CO1	Understand foundational algorithmic concepts, mathematical preliminaries, and perform basic algorithm analysis.	L2
CO2	Implement and analyze basic and linear-time sorting algorithms using control flow and loop invariants..	L4
CO3	Apply divide and conquer strategies to design and analyze efficient algorithms for classical problems..	L3
CO4	Design solutions using dynamic programming and greedy strategies for optimization problems	L6
CO5	Apply backtracking and string matching algorithms to solve complex problems and understand the basics of computational complexity theory.	L3

	TEXT BOOK:	
1.	**Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein** *Introduction to Algorithms*, 3rd Edition, MIT Press	
2.	**S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani** *Algorithms*, McGraw-Hill Education.	
	REFERENCES:	
3.	**Jon Kleinberg and Éva Tardos** *Algorithm Design*, Pearson Education.	
4.	**S. K. Basu** *Design Methods and Analysis of Algorithms*, PHI Learning.	
	ONLINE REFERNCES	

1. MIT OpenCourseWare – Introduction to Algorithms (6.006)
   🔗 https://ocw.mit.edu/courses/6-006-introduction-to-algorithms-fall-2011/
2. GeeksforGeeks – DAA Tutorials
   🔗 https://www.geeksforgeeks.org/fundamentals-of-algorithms

CO	List of Experiments	Hrs.
CO1	**Practical 1:** Implement an algorithm to compute the GCD of two numbers using Euclid's method and analyze its time complexity (Best/Worst/Average case).	2
	Practical 2: Compare the empirical runtime of iterative vs. recursive factorial algorithms and justify using asymptotic notation (O, Ω, Θ).	2
	Practical 3: Implement and analyze the Master Theorem for a divide-and-conquer recurrence (e.g., $T(n) = 2T(n/2) + n$).	2
CO2	**Practical 4:** Implement Bubble Sort, Selection Sort, and Insertion Sort. Compare their performance on random, sorted, and reverse-sorted datasets.	2
	Practical 5: Implement Heap Sort and analyze its time complexity using loop invariants.	2
	Practical 6: Implement Counting Sort and Radix Sort for integer arrays. Compare their efficiency with $O(n \log n)$ sorts.	2
CO3	**Practical 7:** Solve the Max-Min problem using Divide and Conquer and compare its efficiency with the brute-force approach.	2
	Practical 8: Implement Merge Sort and Quick Sort. Analyze their performance on large datasets and discuss pivot selection impact	2
	Practical 9: Implement Strassen's Matrix Multiplication and compare its runtime with the standard $O(n^3)$ method.	2
CO4	**Practical 10:** Solve the 0/1 Knapsack problem using DP and its fractional variant using Greedy. Compare results.	2
	Practical 11: Implement the Longest Common Subsequence (LCS) problem using DP and validate with sample strings.	2
	Practical 12: Implement Huffman Coding for text compression and analyze its optimality.	
CO5	**Practical 13:** Solve the N-Queens problem using Backtracking and count valid configurations for N=4, 8.	2
	Practical 14: Implement the Rabin-Karp algorithm for pattern matching and compare its efficiency with the Naive method.	2
	Practical 15: Classify problems (e.g., TSP, Knapsack) into P, NP, NP-Complete, or NP-Hard categories with justifications.	2
	TOTAL HOURS	**30**

Course Outcomes with Program Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12
CO 1	3	2	2	2	2	-	-	-	-	-	2	1
CO 2	3	3	2	1	2	1	-	-	-	1	2	-
CO 3	3	3	3	2	3	1	-	-	-	1	2	1
CO 4	3	3	2	2	2	1	-	1	-	-	-	-
CO 5	3	3	2	2	2	1	1	-	-	-	-	2