Program	Bachelor of Technology (BTech)	**Semester - 4**
Type of Course	Professional Core	
Prerequisite	Data Structure	
Course Objective	This course introduces various methods to design and analyze algorithms. Students will learn different algorithms for given computational tasks and evaluate their relative merits based on the performance measures.	

Teaching Scheme (Contact Hours)				Examination Scheme				
				Theory Marks		Practical Marks		
Lecture	Tutorial	Practical	Credit	SEE	CIA	SEE	CIA	Total Marks
3	0	2	4	70	30	25	25	150

***SEE** - Semester End Examination, **CIA** - Continuous Internal Assessment (It consists of Assignments/Seminars/Presentations/MCQ Tests, etc.)*

Course Content

T - Teaching Hours | **W** - Weightage

Sr.	Topics	T	W
1	**Introduction to Algorithms and Mathematics and Analysis of Algorithm:** Definition of Algorithm, Characteristics of Algorithm, Algorithm design techniques, Mathematics for algorithmic sets, Functions, and Relations, Linear inequalities, and Linear equations **Analysis of Algorithm:** Algorithm Analysis, Average, best and worst case analysis, Asymptotic Notations, Analysing control statement, Sorting Algorithms and Performance analysis: Bubble sort, Selection sort, Insertion sort, Shell sort, Heap sort, Bucket sort, Radix sort, and Counting sort	**10**	**20**
2	**Divide and Conquer Algorithm** Solving Recurrences, Substitution Method, Recurrence Tree Method, Master's Method, Linear, and binary searching, Merge sort, Quick sort, Multiplying Large Integers, Matrix Multiplication.	**9**	**20**
3	**Greedy Algorithm** General Characteristics of greedy algorithms, Make a change problem, Activity selection problem, Minimum Spanning Tree, Krushkal's Algorithm, Prim's Algorithm, Single source shortest path dijkstra's Algorithm, Knapsack Problem, Huffman Code, Job Scheduling Problem	**9**	**20**
4	**Dynamic Programming** The Principle of Optimality, Generalized solution using Dynamic Programming, Make a change Problem, 0/1 Knapsack Problem, All pair shorted path – Floyd's Algorithm, Chain Matrix Multiplication, Longest common subsequence, Assembly Line Scheduling, Calculating the Binomial Coefficient	**9**	**20**
5	**Graph Algorithm, Branch and Bound, String Matching and NP-Completeness**	**8**	**20**

Course Content

T - Teaching Hours | **W** - Weightage

Sr.	Topics	T	W
	An introduction using graphs, Directed and Undirected graphs, Graph Traversal: DFS, BFS, Articulation Point, and Topological Sorting. **Branch and bound:** Introduction, The Eight queens problem, Knapsack problem using branch and bound Min-Max Principle. **String Matching:** Introduction, The naive string matching algorithm, The Rabin-Karp algorithm, String Matching with finite automata, The Knuth-Morris-Pratt algorithm. **NP-Completeness:** Computational Classes: – P, NP, NP-Complete, and NP-Hard		
		Total	**45**	**100**

Suggested Distribution Of Theory Marks Using Bloom's Taxonomy

Level	Remembrance	Understanding	Application	Analyze	Evaluate	Create
Weightage	15	25	25	25	10	0

NOTE : This specification table shall be treated as a general guideline for the students and the teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcomes

At the end of this course, students will be able to:	
CO1	**discuss** the basics of algorithmic techniques.
CO2	**apply** the time complexity and their notations in problem-solving.
CO3	**analyse** the general strategies of algorithms.
CO4	**implementation** of algorithmic problems.
CO5	**describe** the classes P, NP, and NP Complete.

Reference Books

1.	**Fundamental of Algorithms** By Gills Brassard, Paul Bratley | PHI
2.	**Introduction to Algorithms** By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein | PHI
3.	**Fundamentals of Algorithms** By E. Horowitz et al.

List of Practical

1.	**Implement various problems using iterative and recursive approach** 1. Write a recursive program for calculation of factorial of an integer. 2. Write a program to print the first 50 natural numbers using recursion. 3. Write a program to calculate the sum of numbers from 1 to n using recursion.
2.	**Implement various array operations using recursion** 1. Write a program to print the array elements using recursion. 2. Write a program to count the digits of a given number using recursion. 3. Write a program to calculate the power of any number using recursion.
3.	**Implement matrix transpose, addition and multiplication operations** 1. Write a program to print the transpose of a matrix. 2. Write a program to find the sum of all diagonal elements of a matrix.

	3. Write a program to print the lower triangle of a matrix. 4. Write a program to print the maximum element from a matrix.
4.	**Implementing various data structures like stack, queue and linked list** 1. Write a program to implement stack operations (PUSH, POP, PEEP, CHANGE & DISPLAY) 2. Write a program to implement queue operations (INSERT, DELETE, DISPLAY) 3. Write a program to implement singly linked list operations (INSERT, DELETE, DISPLAY)
5.	**Implement and Analyse time complexity of Bubble and Insertion sort** 1. Write a program to sort array elements using bubble sort. 2. Write a program to sort array elements using insertion sort.
6.	**Implement and Analyse time complexity of Selection sort and Heap sort** 1. Write a program to sort array elements using selection sort. 2. Write a program to sort array elements using heap sort.
7.	**Implementation and Time analysis of Linear and Binary search algorithm** 1. Write a program to implement linear search algorithm. 2. Write a program to implement binary search algorithm.
8.	**Implement and Analyse time complexity of Quick sort and Merge sort** 1. Write a program to implement quick sort algorithm. 2. Write a program to implement merge sort algorithm.
9.	**Implementation of Kruskal's and Prim's algorithms using Greedy algorithm** 1. Write a program to study and implement minimum spanning tree using Kruskal's algorithm. 2. Write a program to study and implement minimum spanning tree using Prim's algorithm.
10.	**Implementation of Dijkstra's and Huffman code algorithms using Greedy algorithm** 1. Write a program to study and implement Dijkstra's algorithm. 2. Write a program to study and implement Huffman code algorithm.
11.	**Implementation of Making a change problem and Largest Common Sub-sequence using Dynamic programming** 1. Write a program to implement making a change problem using dynamic programming. 2. Write a program to implement Largest Common Sub-sequence.
12.	**Implementation of a Knapsack problem using Greedy approach and Dynamic programming** 1. Implement knapsack problem using greedy approach 2. Implement 0/1 knapsack problem using dynamic programming.
13.	**Implementation of DFS and BFS algorithms** 1. Write a program to implement the DFS algorithm. 2. Write a program to implement the BFS algorithm.
14.	**Implementation of Rabin-Karp string matching algorithm** 1. Write a program to implement Rabin-Karp method for pattern searching
15.	**Illustrating Various Algorithms using HTML and JavaScript** 1. Create a visual application using HTML and JavaScript that demonstrates one of the various algorithms.

Miscellaneous

Useful Links
GCC, JAVA W1: https://www2.cs.duke.edu/courses/fall10/cps130/lectures.htm W2: https://www.isical.ac.in/~arijit/courses/spring2017/daa-mtech.html W3: http://www.cs.umd.edu/class/fall2015/cmsc451/