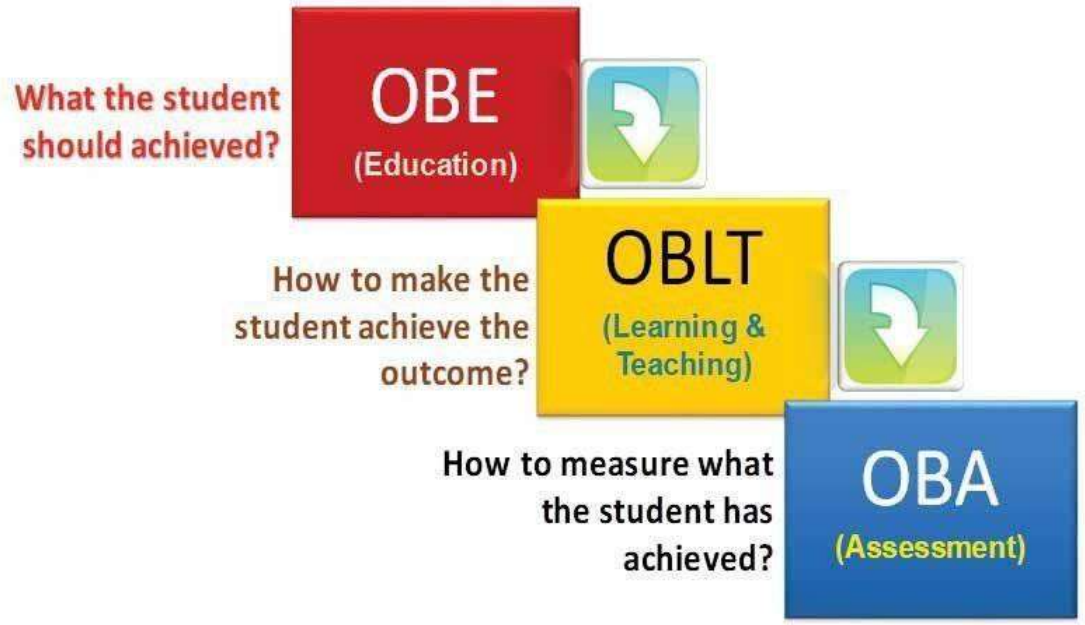
GALGOTIAS UNIVERSITY

GUSOP-01
Course-Pack Framework
A comprehensive instructional delivery document



Prepared by:

VCO

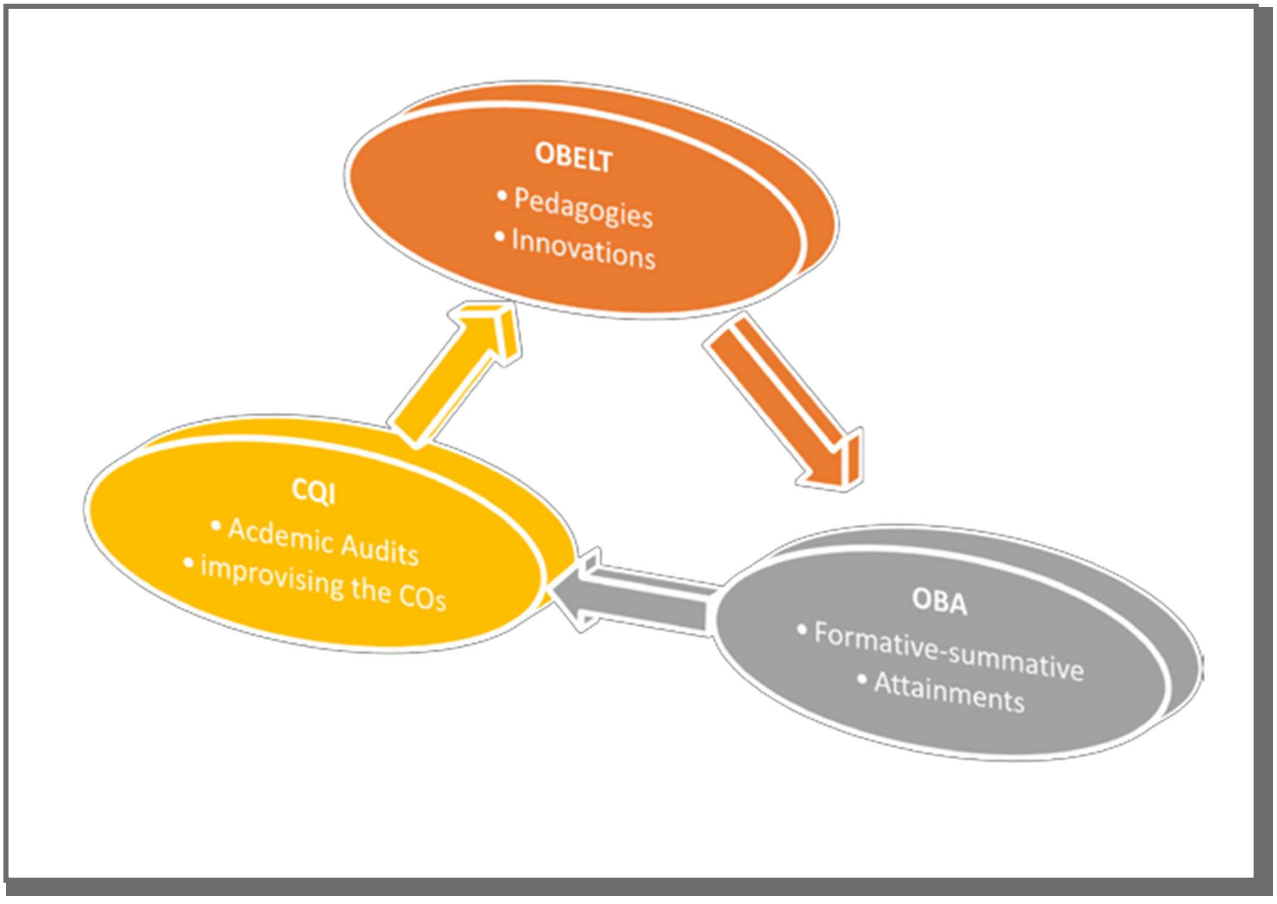
(Revised on Aug-2023)

COURSEPACK
FRAMEWORK

The Course Pack is a comprehensive and complete pedagogical guideline document that describes the components of instruction delivery by a faculty member. It consists of the scheme of the course, Course Overview, Course Objectives, Prerequisite course, Program-specific Outcomes (PSOs), Course outcomes (COs), Bloom's taxonomy (Knowledge Levels), Types of Courses, Course articulation matrix, Course assessment patterns, Course content, Lesson Plan, Bibliography, Problem-based learning/case-studies/clinical, and Student-Centered learning (self- learning towards life-long-learning). It not only provides a uniform design of Course delivery across the University but also ensures freedom and flexibility to introduce innovations in learning and teaching and create vivid kinds of assessment tools (alternate assessment tools) by a faculty member.

The course pack is developed by the faculty member teaching a course. If more than one faculty teaches the same course, all the faculty members teaching the course shall be formed as a cluster, and a senior faculty member (Course-lead) lead the Course delivery design in a team effort. The Course Pack provides ample scope and opportunity to bring innovations in teaching pedagogies in a school/department.

Hence, the Course pack is a comprehensive learning-teaching strategy framework to be followed by all the faculty members in schools/departments in the university. It is not only a tool for measuring the learning of a class but also analyses the achievement levels (learning outcomes of the course) of all the students in a class in a continuous manner.



COURSEPACK

SCHEME

The scheme is an overview of work-integrated learning opportunities and gets students out into the real world. This will give what a course entails.

<table>
<tr><td>Course Title</td><td colspan="3">Design and Analysis of Algorithm</td><td colspan="2">Course Type</td><td colspan="4">Integrated</td></tr>
<tr><td>Course Code</td><td colspan="3">R1UC407B</td><td colspan="2">Class</td><td colspan="4">B.Tech CSE and Specialization IV Sem</td></tr>
<tr><td rowspan="6">Instruction delivery</td><td>Activity</td><td>Credits</td><td>Credit Hours</td><td colspan="4" rowspan="2">Total Number of Classes per Semester</td><td colspan="2" rowspan="2">Assessment in Weightage</td></tr>
<tr><td>Lecture</td><td>3</td><td>4</td></tr>
<tr><td>Tutorial</td><td>0</td><td>0</td><td rowspan="3">Theory</td><td rowspan="3">Tutorial</td><td rowspan="3">Practical</td><td rowspan="3">Self-study</td><td rowspan="3">CIE</td><td rowspan="3">SEE</td></tr>
<tr><td>Practical</td><td>1</td><td>4</td></tr>
<tr><td>Self-study</td><td>1</td><td></td></tr>
<tr><td>Total</td><td>5</td><td>8</td><td>45</td><td>0</td><td>30</td><td>105</td><td>50%</td><td>50%</td></tr>
<tr><td>Course Lead</td><td colspan="2">Dr. Arvind Dagur</td><td>Course Coordinator</td><td colspan="6">Mr. Pradeep Chauhan</td></tr>
<tr><td>Names Course Instructors</td><td colspan="3">Theory</td><td colspan="6">Practical</td></tr>
<tr><td colspan="3">Faculty Name
Akhilesh Kumar Tripathi
Akshay Kumar Malyan
Anurag Maurya
Ankit Sharma
Arvind Dagur
Ashwin Perti
Avaneesh Kumar Yadav
Jyoti Yaduwanshi
Mithilesh Kumar Yadav
Pooja Singh
Pradeep Chauhan
Rakesh Bharati
Sachin Kumar Tyagi
Shachi Mall
Sunil Kumar
Tushar Mehrotra</td><td colspan="6">Faculty Name
Adarsh Singh Chauhan
Akhilesh Kumar Tripathi
Akshay Kumar Malyan
Anurag Maurya
Ankit Sharma
Arvind Dagur
Ashwin Perti
Avaneesh Kumar Yadav
Geeta Gaytri Behera
Ashish Kumar
Jyoti Yaduwanshi
Mandeep
Mithilesh Kumar Yadav
Pooja Singh
Pradeep Chauhan
Punit Kumar Chaubey
Rakesh Bharati</td></tr>
</table>

		Roobal Yadav Sachin Kumar Tyagi Shachi Mall Sunil Kumar Tushar Mehrotra

COURSE OVERVIEW

An Algorithm is a sequence of steps to solve a problem. Design and Analysis of Algorithm is very important for designing algorithm to solve different types of problems in the branch of computer science and information technology. This tutorial introduces the fundamental concepts of Designing Strategies, Complexity analysis of Algorithms, followed by problems on Graph Theory and Sorting methods. This tutorial also includes the basic concepts on Complexity theory.

PREREQUISITE COURSE

PREREQUISITE COURSE REQUIRED	Yes	
If, yes please fill in the Details	**Prerequisite course code**	**Prerequisite course name**
	R1UC308B	**Data Structures**

COURSE OBJECTIVE

This course will introduce students to searching and sorting as well as data structures like stack, queue, and binary tree. Students will write many programs around these topics during this course. The objective of the course:

- Analyze and design different searching and sorting algorithm algorithms based upon different designing approaches.
- Analyze and design different tree algorithms based upon different designing approaches.
- Design efficient algorithmic techniques for solving industry oriented real time problems using Greedy Algorithm.
- Design efficient algorithmic techniques for solving industry oriented real time problems using Dynamic Programming.
- Apply and synthesize efficient algorithms.

COURSE OUTCOMES (COs)

After the completion of the course, the student will be able to:

CO No.	Course Outcomes
R1UC407B.1	Able to analyze the algorithms and find out their time and space complexity.
R1UC407B.2	Able to apply the concepts of brute force and divide and conquer techniques in problem solving.
R1UC407B.3	Apply and understand the mathematical criterion for deciding whether an algorithm is efficient, and know many practically important problems that do not admit any efficient algorithms.
R1UC407B.4	Analyze and apply classical sorting, searching, optimization and graph algorithms.
R1UC407B.5	Apply and understand basic techniques for designing algorithms, including the techniques of recursion, dynamic programming, and greedy.

PROGRAM OUTCOMES (POs):

PO1:	**Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2:	**Problem Analysis:** Identify, formulate, review research literature and analyse complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4).
PO3:	**Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5).
PO4:	**Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5:	**Modern Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6).
PO6:	**The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO7:	**Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9).
PO8:	**Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
PO9:	**Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.
PO10:	**Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments..
PO11:	**Life-Long Learning:** Recognize the need for, and have the preparation and ability for: i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8).
PSO1:	Have the ability to work with emerging technologies in Computer Science and Engineering requisite to Industry 4.0.
PSO2:	Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

BLOOM'S LEVEL OF THE COURSE OUTCOMES

INTEGRATED

CO No.	Remember **BTL1**	Understand **BTL2**	Apply **BTL3**	Analyse **BTL4**	Evaluate **BTL2**	Create **BTL6**
R1UC407B.1				√		
R1UC407B.2						√
R1UC407B.3					√	
R1UC407B.4				√		
R1UC407B.5			√			

PROGRAM OUTCOMES (POs): AS DEFINED BY CONCERNED THE APEX BODIES

COURSE ARTICULATIONMATRIX

COs#/POs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**
R1UC407B.1	3	3	3											
R1UC407B.2	3	2	2		2									
R1UC407B.3	3	2	2											
R1UC407B.4	3	2	2											
R1UC407B.5	3	3	3	3	3									

Note: 1-Low, 2-Medium, 3-High

COURSE CONTENT

Content
Introduction Notion of an Algorithm, Fundamentals of Algorithmic Problem Solving, Problem Types, Analysis of Algorithm Efficiency, Asymptotic Notations and their properties, Mathematical analysis for Recursive and Non-recursive algorithms, Solving Recurrence relations. Substitution method, recursive tree and master method. **Divide and conquer:** Analysis of Binary search, Merge sort, Quick sort, Heap sort, Strassen's Matrix Multiplication. **Tree**: AVL Tree, B Tree, RB Tree **Brute Force:** Traveling Salesman Problem, Knapsack Problem **Greedy Technique:** Minimum spanning tree, Prim's and Kruskal's Algorithm, Huffman Trees and Codes, fractional Knapsack Problem, Single source shortest path: Dijkstra's Algorithm. **Dynamic Programming:** Matrix Chain Multiplication, Longest Common Subsequence Problem, All pair shortest paths: Floyd–Warshall's algorithm, Knapsack Problem. **Backtracking and Branch & Bound** **Backtracking:** n-Queens problem, Subset sum problem, Graph Coloring **Branch & Bound:** Knapsack Problem, Traveling Salesman Problem **String Matching:** Brute force, Rabin Karp, KMP. Introduction to P, NP NP- Complete and NP Hard Problems, Introduction to Approximation algorithms, Approximation Algorithms for NP-Hard Problems.
List of Practical
1. Write a program to sort given set of numbers in ascending/descending order using Bubble sort and also search a number using binary search.
2. Write a program to sort given set of numbers in ascending/descending order using Insertion sort and also search a number using linear search.
3. Write a program to sort given set of numbers in ascending/descending order using Quick sort and any other sorting algorithm. Also record the time taken by these two programs and compare them.
4. Write a program to sort given set of numbers using Heap sort.
5. Write a program to sort given set of numbers Counting Sort.
6. Write a program to sort given set of numbers in ascending/descending order using merge sort
7. Implement a Naive string-matching approach
8. Implement Rabin Karp string-matching approach
9. Write a program to implement Matrix Chain Multiplication.

10. Implement KMP string-matching approach
11. Write a program to sort a given set of numbers without comparing/swapping
12. Write a program to sort a given set of numbers using Maxhipify
13. WAP to implement fractional knapsack problem
14. WAP to implement Strassen's Matrix Multiplication
15. Write a program to implement minimum spanning trees using Prims algorithm
16. Write a program to implement Knapsack using Greedy technique.
17. Write a program to implement the Bellman-Ford Algorithm.
18. Compute the transitive closure of a given directed graph using Warshall's algorithm. Write a program to implement the shortest path algorithm
19. Write a program to solve the LCS problem.
20. Write a program to implement Knapsack using the dynamic technique.
21. Write a program to implement the n-Queen Problem using backtracking.
22. Write a program to implement coin -exchange Problem
23. Write a program to implement sum of subset problem
24. Write a program to implement graph coloring technique
25. Obtain the Topological ordering of vertices in a given digraph.
26. Implement TCS using branch and bound method
27. Write a program to implement the Greedy algorithm using Activity Selection Problem.
28. WAP to implement Huffman Trees and Codes
29. Write a program to implement Dijkstra's Algorithm.
30. Write a program to implement Greedy algorithm using Task Scheduling Problem

COURSE ASSESSMENT

The course assessment patterns are the assessment tools used both in formative and summative examinations.

Assessment Tools	CIE			Total Marks		Final Marks
	LAB @ (Work + Record)	MTE	LAB EXAM*	CIE	SEE	CIE * 0.5 + SEE * 0.5
Integrated	**25**	**50**	**25**	**100**	**100**	**100**

@ Lab Work – 15 marks + Lab record – 10 marks

* Passing criteria – 30% of marks to be secured in the lab exam conducted by two examiners (one internal + one external).

LESSON PLAN FOR COMPREHENSIVE COURSES

FOR THEORY 15 weeks * 3 Hours = 45 Classes) (1credit = 1Lecture Hour)
FOR PRACTICAL 15 weeks * 2Hours = 30 Hours lab sessions (1 credit = 2 lab hours)

LNo	T/L	Topics	Skills	Competency
1	T	Introduction to Algorithms – Definition, characteristics, real-world examples	Designing Algorithms and checking the complexity	Designing algorithms, analyze their space and time complexities.
2	T	Algorithmic problem-solving strategies, problem types		
3	T	Algorithm efficiency – Time & space complexity, order of growth		
4	L	Write a program to sort given set of numbers in ascending/descending order using Bubble sort and also search a number using binary search.		
5	L	Write a program to sort given set of numbers in ascending/descending order using Insertion sort and also search a number using linear search.		
6	T	Asymptotic notations – Big-O, Omega, Theta, properties	Analyze the order of the growth of any algorithm	
7	T	Analysis of non-recursive algorithms (examples: max, sum, matrix operations)	Analyze the complexity of any algorithm with recurrence relation	
8	T	Analysis of recursive algorithms – Recurrence relations introduction		
9	L	Write a program to sort given set of numbers in ascending/descending order using Quick sort and any other sorting algorithm. Also record the time taken by these two programs and compare them.		
10	L	Write a program to sort given set of numbers using Heap sort.		
11	T	Solving recurrences – Substitution method		
12	T	Solving recurrences – Recursion tree method		
13	T	Solving recurrences – Master theorem		
14	L	Write a program to sort given set of numbers Counting Sort.		
15	L	Write a program to sort given set of numbers in ascending/descending order using merge sort	Searching and sorting Algorithms	Analyze and apply searching and sorting algorithms
16	T	Divide and conquer paradigm, Binary search – Analysis, recurrence, iterative vs recursive	Analysis of Pattern Matching Approaches	Finding the optimal solution using different approaches and analyze the problem
17	T	Merge sort – Algorithm, analysis, stability		
18	T	Quick sort – Algorithm, best/worst/average case, partitioning		
19	L	Implement a Naive string-matching approach		

20	L	Implement Rabin Karp string-matching approach		
21	T	Heap sort – Heapify, build heap, analysis	Designing and using Brute Force methods	
22	T	Strassen's matrix multiplication – Idea, recurrence, comparison with naïve method		
23	T	Balanced BSTs – AVL tree rotations, insertion, deletion		
24	T	B-Trees – Structure, operations, applications in databases		
25	L	Write a program to implement Matrix Chain Multiplication.		
26	L	Implement KMP string-matching approach		
27	T	Red-Black Trees – Properties	Designing and analysis of divide and conquer methods	
28	T	Red-Black Trees: Insertion		
29	T	Brute force approach – TSP		
30	T	Brute Force: Knapsack (0/1)		
31	L	Write a program to sort a given set of numbers without comparing/swapping		
32	L	Write a program to sort a given set of numbers using Maxhipify		
33	T	Greedy technique – Characteristics,Minimum spanning tree – Prim's algorithm		
34	T	Minimum spanning tree – Kruskal's algorithm	Designing and using greedy methods	
35	T	Huffman coding – Tree construction, prefix codes		
36	L	WAP to implement fractional knapsack problem		
37	L	WAP to implement Strassen's Matrix Multiplication		
38	T	Fractional knapsack problem		
39	T	Single-source shortest path – Dijkstra's algorithm		
40	T	Dynamic programming – Principle of optimality, vs greedy		
41	L	Write a program to implement minimum spanning trees using Prims algorithm		
42	L	Write a program to implement Knapsack using Greedy technique.		
43	T	Matrix chain multiplication		
44	T	Longest common subsequence (LCS)	Finding the Optimal solution using dynamic approach	
45	T	All-pairs shortest path – Floyd–Warshall algorithm		
46	L	Write a program to implement the Bellman-Ford Algorithm.		
47	L	Compute the transitive closure of a given directed graph using Warshall's algorithm. Write a program to implement the shortest path		

		algorithm		
48	T	DP for 0/1 knapsack problem		
49	T	Backtracking – Concept, state space tree		
50	T	n-Queens problem		
51	L	Write a program to solve the LCS problem.		
52	L	Write a program to implement Knapsack using the dynamic technique.		
53	T	Subset sum problem	Analysis and applications of backtracking method.	
54	L	Write a program to implement the n-Queen Problem using backtracking.		
55	L	Write a program to implement coin -exchange Problem		
56	T	Graph coloring problem		
57	T	Branch & bound – Concept, FIFO, LIFO, LC search		
58	L	Write a program to implement sum of subset problem		
59	L	Write a program to implement graph coloring technique		
60	T	0/1 knapsack using branch & bound	Analysis and applications of Branch and Bound method.	
61	T	Branch & Bound: Traveling Salesman Problem		
62	L	Obtain the Topological ordering of vertices in a given digraph.		
63	L	String matching – Brute force, Rabin–Karp algorithm		
64	T	Knuth–Morris–Pratt (KMP) algorithm		
65	T	Complexity classes – P, NP, NP-Complete, NP-Hard; Introduction to approximation algorithms	Finding the solution of problems in polynomial time , nondeterministic polynomial-time	Understand to solve problems in polynomial time , nondeterministic polynomial-time.
66	T	Complete and NP-Hard Problems.		
67	T	Revision and doubt-clearing session		
68	T	Revision and doubt-clearing session		
69	L	Write a program to implement the Greedy algorithm using Activity Selection Problem.		

70	L	WAP to implement Huffman Trees and Codes		
71	T	Revision and doubt-clearing session		
72	T	Revision and doubt-clearing session		
73	T	Revision and doubt-clearing session		
74	L	Write a program to implement Dijkstra's Algorithm.		
75	L	Write a program to implement Greedy algorithm using Task Scheduling Problem		

BIBLIOGRAPHY

Text Book

1. Thomas H. Coreman, Charles E. Leiserson and Ronald L. Rivest, “Introduction to Algorithms”, The MIT Press, 3rd edition, 2009.

Reference Books

1. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran. Fundamentals of Computer Algorithms, MIT Press, Second Edition (Indian reprint: Prentice-Hall), 2008.
2. Aho, Hopcraft, Ullman, “The Design and Analysis of Computer Algorithms” Pearson Education.

Journals/Magazines/Govt.Reports/Gazatte/IndustryTrends

1. https://leetcode.com

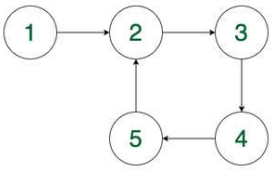
SWAYAM/NPTEL/MOOCs Certification

1. Code Tantra
2. Coding Ninjas
3. Code Chef
4. https://onlinecourses.nptel.ac.in/noc19_cs47/preview (Swayam)
5. GeeksforGeeks

PROBLEM-BASED LEARNING

Exercises in Problem-based Learning (Value Added Experiments)

SNo	Problem	KL
1	Write a program to implement Minimum Cost spanning tree.	KL5
2	Write a program to implement Sum of subset problem.	KL5
3	Write a program to implement Task Scheduling Problem using Greedy technique.	KL5
4	Write a program to implement Activity Selection Problem using Greedy technique.	KL5
5	Compute the transitive closure of a given directed graph using Warshall's algorithm. Write a program to implement shortest path algorithm.	KL6
6	Write a program to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking principle.	KL5
7	Develop a program to implement solve LCS problem.	KL6
8	Develop a program to implement Huffman-code.	KL6
9	Find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm. Use Union-Find algorithms in your program.	KL6
10	Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.	KL6
11	Design a program to (a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm. (b) Implement Travelling Sales Person problem using Dynamic programming.	KL6
12	Design and implement to find a subset of a given set S = {Sl, S2,.... , Sn} of n positive integers whose SUM is equal to a given positive integer d. For example, if S ={1, 2, 5, 6, 8} and d= 9, there are two solutions {1,2,6}and {1,8}. Display a suitable message, if the given problem instance doesn't have a solution.	KL6
13	Transpose of Matrix	KL5
14	Given an (n x m) matrix, rotate the matrix by 90 degrees in clockwise direction.	KL6
15	Given an array arr[] of size N-1 with integers in the range of [1, N], the task is to find the missing number from the first N integers.	KL6
16	Given a string, the task is to count all palindrome sub string in a given string. Length of palindrome sub string is greater than or equal to 2.	KL6
17	Given a set of strings, find the longest common prefix.	KL5
18	Given two strings, the task is to find if a string can be obtained by rotating another string by two places.	KL6
19	Given an expression string exp, write a program to examine whether the pairs and the orders of "{", "}", "(", ")", "[", "]" are correct in the given expression.	KL5

20	Given a string, reverse it using stack.	KL5
21	Given an integer k and a queue of integers, The task is to reverse the order of the first k elements of the queue, leaving the other elements in the same relative order.	KL6
22	Given a singly linked list with N nodes and a number X. The task is to find the node at the given index (X)(1 based index) of linked list.	KL5
23	Given a Linked List and a number N, write a function that returns the value at the Nth node from the end of the Linked List.	KL5
24	Given a linked list, check if the linked list has a loop (cycle) or not. The below diagram shows a linked list with a loop. 	KL6
25	Given two sorted arrays arr1[] and arr2[] of sizes n and m in non-decreasing order. Merge them in sorted order without using any extra space. Modify arr1 so that it contains the first N elements and modify arr2 so that it contains the last M elements.	KL5
26	Given a string S of size N, the task is to sort the string without changing the position of vowels.	KL5
27	Given an array arr[] of size N and a number K, where K is smaller than the size of the array. Find the K'th smallest element in the given array. Given that all array elements are distinct.	KL5
28	Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop.	KL6
29	Given two strings 'str1' and 'str2' of size m and n respectively. The task is to remove/delete and insert the minimum number of characters from/in str1 to transform it into str2. It could be possible that the same character needs to be removed/deleted from one point of str1 and inserted at some another point.	KL6
30	Given are N ropes of different lengths, the task is to connect these ropes into one rope with minimum cost, such that the cost to connect two ropes is equal to the sum of their lengths.	KL6
31	Given a binary search tree, task is to find Kth largest element in the binary search tree.	KL6
32	Given a weighted graph and a source vertex in the graph, find the shortest paths from the source to all the other vertices in the given graph.	KL6

STUDENT-CENTERED LEARNING (Seff-Learning towards lifelong learning)

Self-Learning (it's a typical course-based project to be carried out by a whole class in groups of four students each; they should exhibit higher level BTLs)

The students, in a group, are expected to conceive an idea based on the content (objectives/outcomes) and apply the suitable knowledge to demonstrate their learning.

A)COURSE-BASED PROJECT (Psychomotor skills)

To enhance their skill set in the integrated course, the students are advised to execute course-based design projects. Some sample projects are given below:

S. No	Suggested Projects	KL
1	Comparing Kruskal and Prim's algorithm in MST.	KL5
2	Greedy and Backtracking Algorithm Comparison in Graph Coloring Problem.	KL5
3	Build the game of Capsa Susun using Greedy Algorithm	KL6
4	Solving Travelling Salesman Problem Using Greedy Algorithm and Brute Force Algorithm	KL3
5	Design and Analysis 0/1 Knapsack Problem.	KL6
6	Design Coin Change Problem using Brute Force and Greedy Algorithm	KL6
7	Comparing Exhaustive Search Algorithm and Greedy Algorithm in job Scheduling Problem	KL5
8	Build a Cash Flow Minimiser	KL6
9	Build a CB Mario game using Dynamic Programming Optimisation.	KL6
10	Build an application of Sudoku game using Backtracking method	KL6
11	Build a Snakes & Ladders game, challenge the player to win in minimum number of moves. You can use BFS to compute it.	KL6
12	Make an application like Google Maps - You can use Dijkshtra's algorithm to find the shortest paths, A* Search for more efficient & real time use.	KL6
13	Create a Binary Search Algorithm to Search for a Value with a Certain Precision	KL6
14	Create Space Efficient Algorithm for Subset Sum	KL6
15	Design the game- Place eight queens on an 8 × 8 chessboard so that no queen attacks another queen.	KL6
16	Build a game like SPACE- SHOOTER.	KL6
17	Make a PHONEBOOK using TRIE Data structure	KL6
18	Use a string compression algorithm, like run length encoding or Huffman coding	KL3
19	Build. Game like Flappy Bird.	KL6
20	Build a snake and Ladders game, challenge the player to win in minimum number of moves. You can use BFS to compute it.	KL6

B) SELF-LEARNING THROUGH MOOCs (Cognitive Skills):

1. Code tantra
2. https://www.coursera.org/courses?query=algorithm%20design
3. https://www.geeksforgeeks.org/learn-data-structures-and-algorithms-dsa-tutorial/?ref=shm

(Course Lead) **(Program Chair)** **(Dean)**