Gyanmanjari Innovative University

Syllabus
Gyanmanjari Institute of Technology
Semester-5(B. Tech)

Subject: Analysis and Design of Algorithms: Optimizing Performance and Efficiency-BET1CE15318

Type of course: Professional Core

Prerequisite: Basic knowledge of any object-oriented programming language.

Rationale:

This course provides a comprehensive understanding of algorithmic problem-solving techniques and computational efficiency. It covers fundamental concepts such as algorithm definition, pseudocode conventions, recursive algorithms, time and space complexity analysis, asymptotic notations, randomized algorithms, and the role of algorithms in computing. The syllabus includes major design strategies like Divide and Conquer, Greedy Method, Dynamic Programming, Backtracking, and Branch and Bound with applications such as Binary Search, Merge Sort, Quick Sort, Strassen's Matrix Multiplication, Knapsack Problem, Job Sequencing, Minimum Spanning Trees, Shortest Path Problems, Traveling Salesperson Problem, and Optimal Binary Search Trees. It also introduces graph theory concepts including DFS, BFS, graph coloring, Hamiltonian cycles, and multistage graphs. Finally, the course explores computational complexity through NP-Hard and NP-Complete problems, enabling students to analyze problem tractability and develop efficient algorithmic solutions for real-world computational challenges.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
4	0	2	5	100	50	150

Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation; CCE-Continuous and Comprehensive Evaluation.

Course Content:

Sr.No.	Course Content	Hrs.	% Weightage
1	**Introduction & Divide and Conquer Algorithms** ***Theory Topics:*** Introduction-Algorithm definition, Algorithm Specification (Pseudocode Conventions and Recursive Algorithms),Performance Analysis of Algorithms, Space complexity and Time complexity, Best, Average and worst-case Analysis, Asymptotic Notation, Randomized Algorithms, The Role of Algorithm in Computing, Divide and Conquer Algorithm: General method, Basic of Recursion and its complexity, applications - Binary search, Finding maximum and minimum–subarray problem, Merge sort, Quick sort, Strassen's algorithm for Matrix Multiplication, Tower-of-Hanoi Problem. ***Practical:*** 1. Write a program to perform Linear Search on a sorted array and analyze its time complexity. 2. Write a program to perform Binary Search on a sorted array and analyze its time complexity. 3. Implement Linear Search and Binary Search algorithms and compare their best, average, and worst-case complexities. 4. Write a recursive algorithm to calculate the factorial of a number and analyze the recursion complexity. 5. Develop a recursive program to generate Fibonacci series and compare recursive and iterative approaches. 6. Write a program to find the maximum element in an array using Divide and Conquer technique. 7. Write a program to find the minimum element in an array using Divide and Conquer technique. 8. Implement Merge Sort algorithm and analyze its time and space complexity. 9. Write a program to implement Quick Sort and compare its performance with Merge Sort. 10. Implement Strassen's Matrix Multiplication algorithm for multiplying two matrices. 11. Write a program to find the Maximum Subarray Sum using Divide and Conquer method. 12. Develop a program to measure the execution time of sorting algorithms for different input sizes. 13. Write a program to compare iterative and recursive implementations of finding power of a number. 14. Implement a recursive algorithm for matrix addition and subtraction operations.	18	20 %

15. Write a menu-driven program demonstrating different Divide and Conquer algorithms such as Binary Search, Merge Sort, and Quick Sort.
16. Implement recursive Tower of Hanoi problem and analyze the number of moves required.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Algorithm Implementation and Analysis:** Students are required to implement the given algorithms, compare their complexities, and compile the algorithm names, program codes, outputs, and complexity analysis into a single PDF for submission.	20	
2	**Active Learning Activity** **Algorithm Insight Challenge:** Students must implement any three algorithms using simulators, analyze their time and space complexities with best/average/worst cases in a single table, and submit all implementations and results in one PDF.		10
	Total	20	10

Examination Style:

Algorithm Implementation and Analysis(20 Marks)
Students received algorithm definitions from the faculty such as Recursive Algorithms, Randomized Algorithms, Binary Search, Finding Maximum and Minimum – Subarray Problem, Merge Sort, Quick Sort, Strassen's Algorithm for Matrix Multiplication, and Tower-of-Hanoi Problem. Students need to implement these algorithms in programs and compare their complexities based on the given algorithms. Algorithm name, program codes, outputs, and complexity comparisons must be compiled into a single PDF document and submitted.

Algorithm Insight Challenge (10 Marks)
Students are required to implement any three algorithms from the following—Binary Search, Merge Sort, Quick Sort, Strassen's Matrix Multiplication, Tower of Hanoi, Finding Maximum and Minimum, or Subarray Problem—using simulators such as VisuAlgo, Algorithm Visualizer, OnlineGDB, Replit, or Python Tutor. For each algorithm, students must analyze and compare time and space complexities along with best, average, and worst-case cases (where applicable) and present the results in a single table. All implementations, analysis, and results must be compiled into one PDF document and submitted.

2	**The Greedy Method** ***Theory Topics:*** The General method, applications- Knapsack problem, Job sequencing with deadlines, Minimum cost spanning trees, Prim's Algorithm, Kruskal's Algorithm, Dijkstra Algorithm, Single source shortest path problem. ***Practical:*** 17. Write a program Write a program to solve the Fractional Knapsack Problem using Greedy Method and calculate the maximum profit. 18. Implement the Job Sequencing with Deadlines algorithm to maximize total profit from given jobs. 19. Write a program to construct a Minimum Cost Spanning Tree using Prim's Algorithm. 20. Implement Kruskal's Algorithm to find the Minimum Spanning Tree of a weighted graph. 21. Implement Prim's Algorithm to find the Minimum Spanning Tree of a weighted graph. 22. Implement Dijkstra's Algorithm to find the shortest path from a source vertex to all other vertices in a graph. 23. Write a program to solve the Single Source Shortest Path problem using Dijkstra's Algorithm. 24. Write a program to display the edges selected during Prim's Algorithm step-by-step. 25. Write a program to calculate total minimum cost required to connect all cities in a network using MST. 26. Write a menu-driven program to perform operations of Prim's, Kruskal's, and Dijkstra's Algorithms. 27. Write a program to detect cycles while constructing Minimum Spanning Tree using Kruskal's Algorithm.	18	20 %

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Practical Exploration of Greedy Algorithms:** Students must implement and demonstrate any one algorithm given by faculty using a programming language, showing input, step-by-step execution, and output, and submit the code, explanation, and results in a single PDF.	20	
2	**Active Learning Activity Greedy Scheduling Lab: Job Sequencing with Deadlines Using Simulation Tools:** Students must simulate Job Sequencing with Deadlines using greedy approach, showing job selection, scheduling, output, and complexity analysis, and submit all results in one PDF.		10
	Total	20	10

Examination Style:

Practical Exploration of Greedy Algorithms(20 Marks)
Students are required to implement and demonstrate any one algorithm as per given by faculty from the following set: Knapsack Problem, Job Sequencing with Deadlines, Minimum Cost Spanning Tree, Prim's Algorithm, Kruskal's Algorithm, Dijkstra's Algorithm, or Single Source Shortest Path Problem. The algorithm must be executed using a suitable programming language and the student should clearly show input, step-by-step processing, and final output. All program code, output, and explanation must be compiled into single PDF document and submitted.

Greedy Scheduling Lab: Job Sequencing with Deadlines Using Simulation Tools (10 Marks)
Students are required to perform a simulator-based practical on the Job Sequencing with Deadlines problem using tools. In this activity, students must design a set of jobs with given profits and deadlines and simulate how the greedy algorithm schedules jobs to maximize profit while respecting deadlines. The simulation should clearly show job sorting by profit, slot allocation step-by-step, and final scheduled output. Students are also required to explain the greedy strategy used, real-world applications, and analyze the time and space complexity of the algorithm. All implementation steps, simulation results, and explanations must be compiled into a single PDF document and submitted for evaluation.

3	**Dynamic Programming and Search Techniques** ***Theory Topics:*** The General Method, applications- Multistage Graphs,all pairs shortest path problem, Optimal binary search trees, 0/1 knapsack problem, Reliability design, Traveling sales person problem. ***Practical:*** 28. Write a program to find the shortest path in a Multistage Graph using Dynamic Programming. 29. Write a program to find the minimum cost path in a weighted multistage graph. 30. Write a program to implement the All-Pairs Shortest Path Problem. 31. Write a program to construct an Optimal Binary Search Tree and calculate its minimum search cost. 32. Write a program to calculate the expected search cost of an Optimal Binary Search Tree. 33. Write a program to solve the 0/1 Knapsack Problem using Dynamic Programming and determine the maximum profit. 34. Develop a program to compare greedy and dynamic programming approaches for the Knapsack Problem. 35. Implement a Dynamic Programming solution for Reliability Design Problem to maximize system reliability. 36. Implement Traveling Salesperson Problem using adjacency matrix representation of graphs. 37. Implement the Traveling Salesperson Problem (TSP) using Dynamic Programming approach. **Evaluation Method:**	18	20%

Sr. No.	Evaluation Methods	SEE	CCE
1	**Dynamic Programming Problem Implementation:** Students must implement any one dynamic programming problem as assigned by faculty, using a programming language, and submit code, input/output, and explanation in a single PDF.	20	
2	**Active Learning Assignment Traveling Salesman Problem via Online Simulation:** Students must simulate the Traveling Salesman Problem (TSP) using online tools by modeling a weighted graph, finding the optimal tour using brute force or dynamic programming, and submitting the results, output, and complexity analysis in one PDF.		10
	Total	20	10

	Examination Style: **Dynamic Programming Problem Implementation (20 Marks)** Students are required to perform a programming-based practical on any one problem from the following set as per given by faculty: Multistage Graphs, All-Pairs Shortest Path Problem, Optimal Binary Search Trees, 0/1 Knapsack Problem, Reliability Design, or Traveling Salesperson Problem. The selected problem must be implemented using a suitable programming language, clearly demonstrating the dynamic programming approach. Students should also explain the general method of dynamic programming and its real-world applications. The program code, input, output must be compiled into a single PDF document and submitted. **Traveling Salesman Problem via Online Simulation (10 Marks)** Students are required to perform a simulator-based practical on the Traveling Salesman Problem (TSP) using online tools In this activity, students must define a weighted graph representing cities, simulate the TSP solution process using a suitable algorithm (such as brute force or dynamic programming), and demonstrate how the optimal tour is constructed step by step. The simulation should clearly show input graph creation, route evaluation, cost calculation, and final optimal path selection. Students are also required to explain the working principle of TSP, its real-world applications, and analyze time and space complexity. All implementation steps, simulation outputs, and explanations must be compiled into a single PDF document and submitted for evaluation.		
4	**Graph and Backtracking** ***Theory Topics:*** An introduction to Graph, Basic Definitions, Traversing Graphs – Depth First Searchand Traversal, Breadth First Searchand Traversal, Backtracking – The General Method, The Eight Queen Problem, Sum of Subsets, Graph Coloring, Hamiltonian cycles. ***Practical:*** 38. Write a program to represent a graph using adjacency matrix and adjacency list. 39. Implement Depth First Search (DFS) traversal for a given graph. 40. Write a program to perform Breadth First Search (BFS) traversal using queue data structure. 41. Develop a program to compare DFS and BFS traversal techniques on the same graph. 42. Write a recursive program for graph traversal using DFS approach. 43. Implement the Eight Queen Problem using Backtracking technique. 44. Write a program to display all possible solutions of the Eight Queen Problem.	18	20%

45. Implement the Sum of Subsets problem using Backtracking and display all valid subsets.
46. Write a program to solve the Graph Coloring problem using Backtracking method.
47. Implement Hamiltonian Cycle algorithm and determine whether a Hamiltonian cycle exists in a graph.
48. Write a program to find all Hamiltonian paths in a given graph.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Graph Traversal and Backtracking Problem:** Students must implement any one assigned problem using a programming language, showing input, steps, and output with brief explanation, and submit code and results in a single PDF.	20	
2	**BFS vs DFS Comparative Analysis:** Students must simulate BFS and DFS using online tools by constructing a graph, showing traversal steps and outputs, and comparing time and space complexities in a table, then submit all results in one PDF.		10
	Total	20	10

Examination Style:

Graph Traversal and Backtracking Problem (20 Marks)
Students are required to perform a programming-based practical on any one of the following topics as assigned by the faculty: Depth First Search (DFS), Breadth First Search (BFS), Eight Queen Problem, Sum of Subsets, Graph Coloring, or Hamiltonian Cycle. The selected problem must be implemented using a suitable programming language, clearly demonstrating the working process such as graph traversal steps or backtracking decision tree. Students should show input, intermediate steps, and final output, along with a brief explanation of the algorithm and its real-world applications. The program code, output, and input must be compiled into a single PDF document and submitted.

BFS vs DFS Comparative Analysis (10 Marks)
Students are required to perform a simulator-based practical on Breadth First Search (BFS) and Depth First Search (DFS) using online tools. In this activity, students must construct a sample graph and simulate both BFS and DFS step-by-step, clearly showing traversal order, visited nodes, and execution flow. After simulation, students must compare BFS and DFS based on time complexity, space complexity, memory usage, and traversal behavior in a tabular format. The submission must include

<table>
<tr><td></td><td>inputs, simulation outputs, comparison analysis, and explanations compiled into a single PDF document for evaluation.</td><td></td><td></td></tr>
<tr><td>5</td><td>Branch and Bound & NP Problems

<u>Theory Topics:</u>

Branch and Bound- General Method, applications-0/1 Knapsack problem, LC Branch and Bound solution, FIFO Branch and Bound solution, Traveling sales person problem. NP-Hard and NP-Complete problems- Basic concepts, non-deterministic algorithms, NP - Hard and NP-Complete classes.

<u>Practical:</u>

49. Write a program to solve the 0/1 Knapsack Problem using Branch and Bound technique.
50. Implement LC (Least Cost) Branch and Bound solution for the 0/1 Knapsack Problem.
51. Write a program to solve the 0/1 Knapsack Problem using FIFO Branch and Bound method.
52. Implement Traveling Salesperson Problem (TSP) using Branch and Bound technique.
53. Write a program to find the minimum cost tour in a Traveling Salesperson Problem using Branch and Bound.
54. Write a program to compare Dynamic Programming and Branch and Bound solutions for the Knapsack Problem.
55. Implement a non-deterministic algorithm simulation for solving computational problems.
56. Write a program to classify given computational problems into P, NP, NP-Hard, and NP-Complete categories.
57. Develop a case study explaining real-world examples of NP-Hard and NP-Complete problems.
Evaluation Method:
<table>
<tr><th>Sr. No.</th><th>Evaluation Methods</th><th>SEE</th><th>CCE</th></tr>
<tr><td>1</td><td>Branch and Bound for Complex Problem Solving:
Students must implement any one Branch and Bound problem, showing state space tree exploration, bounding strategy, input/output, and brief NP-Hard/NP-Complete explanation, and submit everything in one PDF.</td><td>20</td><td></td></tr>
<tr><td>2</td><td>Next-Gen Computing Algorithms: A Comprehensive Survey:
Teams of 2 students will review recent research papers on advanced algorithms and prepare a structured survey paper with comparative analysis and proper references for submission on the GMIU portal in PDF.</td><td></td><td>10</td></tr>
<tr><td></td><td>Total</td><td>20</td><td>10</td></tr>
</table></td><td>18</td><td>20 %</td></tr>
</table>

	Examination Style: **Branch and Bound for Complex Problem Solving (20 Marks)** Students are required to perform a programming-based practical on any one of the following topics as assigned by the faculty: 0/1 Knapsack Problem using Branch and Bound (LC or FIFO approach), Traveling Salesman Problem using Branch and Bound, or any relevant Branch and Bound application. The implementation must clearly demonstrate problem formulation, state space tree exploration, bounding strategy, and step-by-step selection of nodes leading to the optimal solution. Students should also include input, output, and a brief explanation of NP-Hard and NP-Complete concepts related to the selected problem. The program code, output, and explanation must be compiled into a single PDF document and submitted. **Next-Gen Computing Algorithms: A Comprehensive Survey (10 Marks)** Students will form teams of 2 members to study and review 5–8 recent research articles from reputed sources such as ACM, IEEE, or arXiv focusing on advanced next generation computational and algorithmic methodologies. Each team will choose a specialized domain (such as graph algorithms, optimization, machine learning algorithms, searching techniques, etc.), critically evaluate and compare the proposed methods, and prepare a survey paper discussing major contributions, emerging trends, challenges, limitations, and possible future enhancements. The report must follow a well-structured format of research paper with appropriate citations and references. The final survey paper in PDF format should be submitted through the GMIU web portal.		

Suggested Specification table with Marks (Theory):100

Distribution of Theory Marks (Revised Bloom's Taxonomy)						
Level	Remembrance (R)	Understanding (U)	Application (A)	Analyze (N)	Evaluate (E)	Create (C)
Weightage %	15%	15%	15%	25%	20%	10%

Course Outcome:

Afterlearningthecourse, the students should be able to:	
CO1	Apply algorithm design techniques such as recursion and divide-and-conquer to analyze and solve computational problems while evaluating their time and space complexities.
CO2	Design and implement greedy algorithms to solve optimization problems such as knapsack, job sequencing, minimum spanning trees, and shortest path problems efficiently.
CO3	Apply dynamic programming techniques to develop optimal solutions for graph, optimization, and resource allocation problems.
CO4	Analyze graph traversal and backtracking techniques to solve combinatorial and graph-based computational problems efficiently.
CO5	Evaluate Branch and Bound strategies and computational complexity concepts to solve optimization problems and classify NP-Hard and NP-Complete problems.

Instructional Method:

The course delivery method will depend upon the requirement of content and the needs of students.The teacher, in addition to conventional teaching methods byblack board, may also use anytools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

The internal evaluation will be done on the basis of continuous evaluation of students in the laboratory and class-room.

Practical examination will be conducted at the end of semester for evaluation of performance of students in the laboratory.

Students will use supplementary resources such as online videos, NPTEL videos, e-courses, Virtual Laboratory.

Reference Books:

[1] Fundamental of Computer Algorithms by Ellis Horowitz, Sartaz Sahni and Sanguthevar Rajasekarn, Computer Science Press.

[2] Introduction to Algorithms by Thomas H. Coreman, Charles E. Leiserson, Ronald Rivest and Clifford Stein, MIT Press.

[3] Fundamentals of Algorithms by Gilles Brassard and Paul Bratley, Prentice Hall of India.

[4] Introduction to Design and Analysis of Algorithms by Anany Levitin, Pearson.

[5] Design and Analysis of Algorithms by Parag Himanshu Dave and Himanshu Bhalchandra Dave, 1st Edition, PHI.

Suggested Assessment Guidelines:

Module-1: Introduction& Divide and Conquer Algorithms

Algorithm Implementation and Analysis (20 Marks)		
Criteria	**Description**	**Marks**
Algorithm Implementation	Correct implementation of assigned algorithms with proper logic and structure	5
Program Code Quality	Clean, readable, and well-documented code with proper naming and indentation	5
Output & Execution	Correct execution showing expected input and output results for all algorithms	5
Complexity Analysis & Comparison	Proper explanation and comparison of time and space complexities of all algorithms	5
	Total	20

Module-2: The Greedy Method

Practical Exploration of Greedy Algorithms (20 Marks)		
Criteria	**Description**	**Marks**
Algorithm Implementation	Correct implementation of the selected greedy algorithm with proper logic and structure	5
Input & Step-by-Step Execution	Clear representation of input data and step-by-step processing of the algorithm	5
Output & Result Verification	Correct final output with proper validation of solution	5
Code Quality	Well-structured, readable code with proper comments and formatting	5
	Total	20

Module-3: Dynamic Programming and Search Techniques

Dynamic Programming Problem Implementation (20 Marks)		
Criteria	**Description**	**Marks**
Problem Understanding & Modeling	Correct formulation of the problem and identification of DP approach	5
Algorithm Implementation	Proper implementation of the selected dynamic programming algorithm with correct logic	5
Input, Table Construction & Execution Steps	Clear input representation and step-by-step DP table filling or computation process	5
Output & Result Validation	Accurate final result with correct verification of optimal solution	5
	Total	20

Module-4: Graph and Backtracking

Graph Traversal and Backtracking Problem (20 Marks)		
Criteria	**Description**	**Marks**
Problem Formulation	Correct formulation of the selected problem with appropriate algorithm choice	5
Algorithm Design & Implementation	Proper implementation of DFS/BFS or backtracking algorithm with correct logic	5
Execution Trace	Clear demonstration of input and step-by-step processing (traversal/decision tree)	5
Result Analysis	Accurate final output with explanation of solution correctness	5
	Total	20

Module-5: Branch and Bound & NP Problems

Branch and Bound for Complex Problem Solving (20 Marks)		
Criteria	**Description**	**Marks**
Problem Understanding & Representation	Correct interpretation of the problem and suitable modeling using Branch and Bound approach	5
Algorithm Design & Implementation	Proper coding/implementation of LC or FIFO Branch and Bound strategy with correct logic	5
Search Process & Bounding Analysis	Clear demonstration of state space tree exploration and bounding technique	5
Final Solution & Verification	Accurate optimal result with proper validation and justification	5
	Total	20