Government Polytechnic, Balangir

Department of Computer Science Engineering

LESSON PLAN 2026-27(WINTER)

NAME OF THE TEACHER : ANJALI MUNDA, GUEST FACULTY

Subject: **ALGORITHMS**(Course Code: CSEPC 209/TH5)
Program: Diploma in Computer Science and Engineering
Semester: 3rd
Total Contact Hours: 45
Total Marks: 100
Assessment: Internal Assessment – 30, End Term – 70

After completion of the course, the students will be able to:
CO1: Define Algorithm with its characteristics.
CO2: Write algorithms with pseudocode.
CO3: Implement algorithms for sorting and searching using appropriate data structures.
CO4: Analyze the time and space complexity of algorithms.
CO5: Design solutions using advanced data structures for real-world applications, such as shortest path problems or flow-based algorithms.

Period	Unit	Topic / Sub-Topic	Learning Objectives (Bloom's Aligned)	Activity	Course Outcome	Learning Methodology	Homework
1	I	Introduction to Algorithms	Define the concept and necessity of an algorithm.	Concept Mapping	CO1	Lecture & Discussion	Read textbook introduction on computational problems.
2	I	Criteria of Algorithms: Input/Output	Identify the required structure for data ingestion and return.	Input/Output Modeling	CO1	Demonstration	Write input/output specs for a calculator tool.
3	I	Criteria: Finiteness, Definiteness, Effectiveness	Evaluate logical boundaries and clarity of steps.	Logic Fault Hunting	CO1	Problem Solving	Identify infinite loop risks in standard tasks.
4	I	Algorithms and Programs	Differentiate between abstract logic and syntax-specific code.	Concept Comparison	CO1	Lecture & Visuals	Chart the differences between design and execution.
5	I	Writing Algorithms: Pseudocode Basics	Construct foundational sequential logic in pseudocode.	Syntax Translation	CO2	Demonstration	Write pseudocode for swapping two variables.
6	I	Writing Algorithms: Control Structures	Formulate conditional and iterative loops in pseudocode.	Flowchart to Pseudocode	CO2	Problem Solving	Draft pseudocode for finding the maximum of three numbers.
7	I	Pseudocode Practice	Design end-to-end pseudocode for mathematical problems.	Dry Run Exercises	CO2	Lecture & Coding	Write pseudocode to generate a prime number list.
8	I	Unit I Review & Synthesis	Consolidate foundational algorithm criteria and drafting skills.	Peer Review	CO1, CO2	Discussion	Review Unit I notes for upcoming complexity analysis.
9	II	Algorithmic Complexity: Concept & Space Complexity	Assess memory overhead required during execution.	Memory Tracing	CO4	Lecture & Demo	Calculate variable footprint for a simple script.

10	II	Time Complexity Introduction	Define execution time growth relative to input size.	Graph Plotting	CO4	Lecture & Visuals	Read about linear vs. exponential growth.
11	II	Worst-Case Analysis	Evaluate the maximum execution boundary of a program.	Scenario Analysis	CO4	Problem Solving	Identify the worst case for sequential search.
12	II	Average and Best-Case Analysis	Differentiate standard operational speeds from ideal scenarios.	Statistical Comparison	CO4	Demonstration	Outline the best case for a pre-sorted array search.
13	II	Big-O Notation Fundamentals	Apply asymptotic notation to classify algorithmic growth.	Notation Mapping	CO4	Lecture	Memorize common Big-O classifications.
14	II	Finding Complexity (Simple Loops)	Calculate Big-O for single iterative structures.	Code Tracing	CO4	Problem Solving	Determine Big-O for 5 provided loop structures.
15	II	Finding Complexity (Nested Loops)	Calculate Big-O for multi-dimensional operations.	Algorithmic Teardown	CO4	Demonstration	Calculate time complexity for a matrix addition logic.
16	II	Unit II Review & Practice	Validate mastery of complexity evaluation.	Quiz / Live Test	CO4	Assessment	Complete worksheet on Big-O calculation.
17	III	Concept of Iteration and Recursion	Contrast loop-based repetition with self-referential functions.	Call-Stack Visualizing	CO4	Lecture & Demo	Outline the base case necessity in recursion.
18	III	Examples: Fibonacci & Factorial	Trace standard mathematical recursive patterns.	Diagram Drawing	CO2, CO4	Problem Solving	Draw the recursive tree for Fibonacci(4).
19	III	Tower-of-Hanoi Problem	Solve multi-step puzzle logic using recursive calls.	Logical Simulation	CO2, CO4	Demonstration	Trace the moves for a 3-disk Tower of Hanoi.
20	III	Complexities of Recursive Algorithms	Formulate recurrence relations for recursive space/time.	Math Derivation	CO4	Lecture & Coding	Calculate the Big-O for the factorial function.
21	III	Conversion: Recursive to Iterative	Refactor self-referential logic into standard memory loops.	Code Refactoring	CO2, CO4	Problem Solving	Convert recursive factorial to a while loop.
22	III	Unit III Review & Synthesis	Ensure comprehension of memory overhead in recursion.	Q&A Session	CO2, CO4	Discussion	Prepare for algorithm design paradigms.
23	IV	Algorithm Paradigms: Greedy Method	Design solutions based on localized optimum choices.	Logic Modeling	CO5	Lecture & Visuals	Read chapter on the Fractional Knapsack problem.
24	IV	Greedy Method Implementations	Trace execution of greedy logic on resource allocation.	Dry Run Exercises	CO5	Problem Solving	Trace greedy coin change logic on paper.

25	IV	Divide and Conquer Concept	Formulate solutions by breaking problems into independent sub-problems.	Architecture Mapping	CO5	Demonstration	Draft the split phase of an array division.
26	IV	Divide and Conquer Implementations	Assess the recombination phase of subdivided tasks.	Merge Simulation	CO5	Problem Solving	Trace a binary search divide pattern.
27	IV	Dynamic Programming	Design optimization strategies utilizing overlapping sub-problems.	Memoization Tracing	CO5	Lecture & Coding	Compare recursive Fibonacci vs. DP Fibonacci speeds.
28	IV	Branch and Bound	Evaluate state-space search trees for optimization.	Tree Pruning Exercise	CO5	Demonstration	Outline how bounding limits unnecessary computation.
29	IV	Backtracking	Implement trial-and-error search with *constraint* checking.	Maze Navigation Game	CO5	Problem Solving	Trace the N-Queens problem *on a 4x4 grid.*
30	IV	Unit IV Review & Synthesis	Correlate correct paradigm strategies to real-world problems.	Strategy Matching	CO5	Discussion	Categorize 5 mock problems by ideal paradigm.
31	V	Sorting: Bubble & Selection Sort	Implement adjacent swapping and minimum-value selection.	*Array Sorting* Simulation	CO3	*Lecture* & Demo	Trace Bubble Sort on a 5-element unsorted array.
32	V	Sorting: Insertion & Merge Sort	Contrast basic insertion with divide-and-conque r sorting.	Code Tracing	CO3	Problem Solving	Draft the merge logic for two sorted sub-arrays.
33	V	Sorting: Quicksort	Utilize pivot elements to partition and sort datasets.	Pivot Simulation	CO3	Demonstration	Trace Quicksort partitioning on paper.
34	V	Sorting: Heap & Radix Sort	Apply priority queues and non-comparative digit sorting.	Digit Grouping Exercise	CO3	Lecture & Visuals	Diagram a Max-Heap structure.
35	V	Searching: Symbol Tables & BST	Structure data dynamically for rapid logarithmic retrieval.	Tree Node Insertion	CO3	Demonstration	Insert 6 values into an empty Binary Search Tree.
36	V	Balanced Search Trees	Analyze methods to prevent tree degradation (e.g., AVL concepts).	Rotation Tracing	CO3	Lecture & Visuals	Read about Left/Right tree rotations.
37	V	Hashing Principles	Map keys to array indices using mathematical hash functions.	Hash Calculation	CO3	Problem Solving	Compute hash values using modulo arithmetic.
38	V	Hash Tables & Collisions	Resolve index overlaps using chaining or open addressing.	Array Mapping	CO3	Demonstration	Resolve a collision using linear probing.

39	VI	Graphs: Directed/Undirected, Paths, Cycles	Define vertex relationships and continuous routes.	Network Diagramming	CO5	Lecture & Visuals	Draw a 5-node graph containing a cycle.
40	VI	Spanning Trees & Directed Acyclic Graphs (DAGs)	Evaluate sub-graphs that connect all vertices without cycles.	Edge Reduction Exercise	CO5	Problem Solving	Identify a spanning tree from a provided graph.
41	VI	Topological Sorting	Sequence dependencies sequentially within a DAG.	Task Scheduling Map	CO5	Demonstration	Topologically sort a mock prerequisite chart.
42	VI	Minimum Spanning Tree (MST) Algorithms	Apply Greedy logic (Kruskal/Prim) to find lowest-cost network connections.	Weight Calculation	CO5	Lecture & Coding	Trace Prim's algorithm on a weighted graph.
43	VI	Shortest Path: Dijkstra's Algorithm	Calculate the most efficient route from a single source.	Table Updating Exercise	CO5	Problem Solving	Execute Dijkstra's on a 6-node weighted map.
44	VI	Flow-Based Algorithms	Analyze capacity limits and flow optimization in networks.	Pipeline Simulation	CO5	Demonstration	Trace maximum flow through a simple network graph.
45	VI	Course Review & Optimization Strategies	Synthesize complexity, paradigms, and structures.	Q&A / System Design	CO1-CO5	Discussion	Complete full syllabus revision for the End Term Exam.

Anjali Munda
13/07/26
Signature of Teacher

Deepak kumar Dash
13.07.2026
Signature of HOD