CODE	COURSE NAME	CATEGORY	L	T	P	CREDIT
ITT304	ALGORITHM ANALYSIS AND DESIGN	PCC	3	1	0	4

Preamble: The syllabus is prepared with a view to equip the Engineering Graduatesto learn basic concepts in algorithms, and to instil the confidence to solve non-conventional problems using different problem solving strategies.

Prerequisite:

- ITT201 Data Structures

Course Outcomes: After completion of the course the student will be able to

CO No.	Course Outcome (CO)	Bloom's Category Level
CO 1	Explain asymptotic notations used in the performance analysis of algorithms and to solve recurrence equations	Level 2: Understand
CO 2	Apply divide and conquer strategy to solve practical problems efficiently	Level 3: Apply
CO 3	Apply greedy and dynamic programming techniques in algorithm design	Level 3: Apply
CO 4	Apply backtracking and branch and bound techniques in algorithm design	Level 3: Apply
CO 5	Interpret sophisticated algorithms such as string matching and approximation algorithms	Level 2: Understand

Mapping of Course Outcomes with Program Outcomes

POs / COs	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO 1	3	-	-	-	-	-	-	-	-	-	-	2
CO 2	3	3	3	-	3	-	-	-	-	-	-	2
CO 3	3	3	3	-	3	-	-	-	-	-	-	2
CO 4	3	3	3	-	3	-	-	-	-	-	-	2
CO 5	3	3	-	-	-	-	-	-	-	-	-	2

3/2/1: high/medium/low

Assessment Pattern

Bloom's Category	Continuous Assessment Tests		End Semester Examination
	Test 1 (Marks)	Test 2 (Marks)	Marks
Remember			
Understand	30	30	60
Apply	20	20	40
Analyze			
Evaluate			
Create			

Mark distribution

Total Marks	Continuous Internal Evaluation (CIE)	End Semester Examination (ESE)	ESE Duration
150	50	100	3 hours

Continuous Internal Evaluation Pattern:

Attendance	: 10 marks
Continuous Assessment Test (2 numbers)	: 25 marks
Assignment/Quiz/Course project	: 15 marks

End Semester Examination Pattern: There will be *two* parts; **Part A** and **Part B**. Part A contain 10 questions with 2 questions from each module, having 3 marks for each question. Students should answer *all* questions. Part B contains 2 questions from each module of which student should answer *any one*. Each question can have maximum 2 sub-divisions and carry 14 marks.

Sample Course Level Assessment Questions

Course Outcome 1 (CO 1):

1. What is an asymptotic notation? Give the different notations used to represent the complexity of algorithms.
2. Write the recurrence equation for your algorithm and solve it to estimate the time complexity of the algorithm.

Course Outcome 2 (CO 2):

1. Give the divide and conquer solution for binary search and analyse its complexity.

2. What is the notion behind the divide and conquer method? Apply divide and conquer strategy to perform merge sort on an array of integers.

Course Outcome 3 (CO 3):

1. Why Kruskal's minimum cost spanning tree construction method is considered as a Greedy method for problem solving?
2. State fractional knapsack problem. Give an algorithm for solving the fractional knapsack problem using greedy strategy.

Course Outcome 4 (CO 4):

1. Draw the state space tree corresponding to 4-Queens problem.

Course Outcome 5 (CO 5):

1. Write an algorithm based on Rabin Karp method to find all the occurrences of pattern P[0..m-1] from a given string str[0....n-1], where n>m. Compare the time complexity of this algorithm with the naive approach.
2. Suggest an algorithm for finding the vertex cover of a graph.

Model Question Paper

Course Code: ITT304

Course Name: Algorithm Analysis and Design

Max.Marks:100 **Duration: 3 Hours**

Part A

Answer all questions. Each question carries 3 marks. (10 * 3 = 30 Marks)

1. What are the properties of a good algorithm?
2. Write down the control abstraction of divide and conquer.
3. List and explain the characteristic properties associated with a problem that can be solved using dynamic programming.
4. What is backtracking? Give one problem that can be solved by backtracking.
5. Differentiate Fixed Tuple and Variable Tuple formulation.
6. Define Least Common Sequence Problem.
7. What is principle of optimality?
8. Write an algorithm for matrix multiplication using Divide and conquer method.

9. What are approximation algorithms?

10. Differentiate between deterministic and nondeterministic algorithm.

Part B

Answer all questions. Each question carries 14 marks. (5 * 14 = 70 Marks)

11. (a) Compare asymptotic notations with examples (6 marks)

    (b) What is amortized analysis? Explain any one method to perform amortized analysis with an example. (8 marks)

OR

12. (a) Compare the growth of time complexity for the following set of functions. (6 marks)

    (i) 2n and n^2
    ii) √log n and log log n
    iii) n√n and n log n

    (b) Solve the recurrence equation using recursive tree method: T(n)= 2T(n/2)+c (8 marks)

13. (a) How we can prove that Strassen's matrix multiplication is advantageous over ordinary matrix multiplication? (8 marks)

    (b) Explain why worst case complexity of Quick sort is O(n^2) and average case complexity is O(n log n) (6 marks)

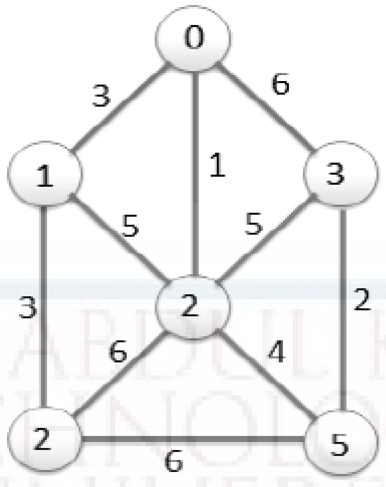
OR

14. (a) Write a recursive algorithm for implementing Binary Search. Illustrate the divide and conquer approach through this algorithm. (7 marks)

    (b) Design a recursive algorithm to find the maximum and minimum from a set of *n* numbers .Illustrate with an example. (7 marks)

15. (a) Explain Kruskal's algorithm. Find the minimum cost spanning tree of the graph whose vertices are v1,v2,v3,v4,v5,v6 and v7. Cost of graph edges are (v1,v2)= 28, (v1,v6)=10, (v6,v5)=25, (v5,v4)=22, (v7,v2)=14, (v2,v3)=16, (v3,v4)=12, (v4,v7)=18 (8 marks)

    (b) Draw and explain each stages of execution of Prim's algorithm in the following graph. (6 marks)



OR

16. (a) Find an optimal solution to the fractional knapsack problem for an instance with number of items 7, Capacity of the sack W=15, profit associated with the items (p1,p2,…,p7) = (10,5,15,7,6,18,3) and weight associated with each item (w1,w2,…,w7)= (2,3,5,7,1,4,1). (5 marks)
(b) Define Travelling Salesman Problem (TSP). Explain the basic steps that are to be followed to solve TSP using branch and bound. Illustrate with an example. (9 marks)

17. (a) Define the following tree organization for representing solution spaces.

(i) Problem state (ii) State space (iii) Solution state (6 marks)

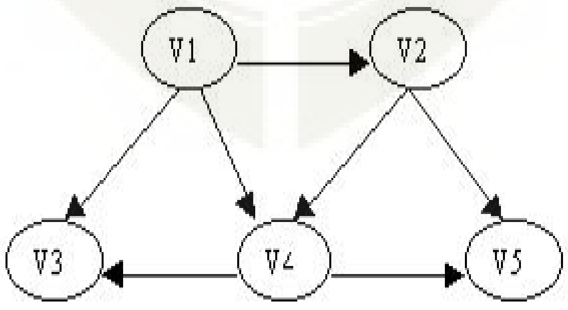
(b) Describe Branch and Bound technique. Demonstrate a problem that can be solved by branch and bound method. (8 marks)

OR

18. (a) What is 15-puzzle problem? How can it be solved? (8 marks)

(b) What is the relevance of Least cost search? Give the control abstraction for Least Cost Search. (6 marks)

19. (a) Perform Topological sorting on the given graph. (5 marks)



(b) Explain vertex cover problem using an example. Suggest an algorithm for finding vertex cover of a graph. (9 marks)

OR

20. (a) Write an algorithm based on Rabin Karp method to find all the occurrences of the pattern P[0..m-1] from a given string str[0....n-1], where n>m. Analyse and Compare the time complexity of this algorithm with the naive approach. (10 marks)

(b) Give an approximate algorithm for graph coloring problem. (4 marks)

Syllabus

Module 1: Introduction to Algorithms (9 Hours)
Properties of a good Algorithm, Development of an Algorithm, Pseudo-code Conventions, Recursive Algorithms Performance Analysis - Space and Time Complexity, Running Time Comparison - Worst, Best and Average Case Complexity, Asymptotic Notations, Common Complexity Functions Recurrence Relations – Solving Recurrences using substitution and recurrence trees Amortized Complexity – aggregate analysis, cost-accounting and potential methods
Module 2: Divide and Conquer (8 Hours)
Divide and Conquer - Control Abstraction, Finding Maximum and Minimum, Binary Search, Strassen's Matrix Multiplication, Quick Sort, Merge Sort
Module 3: Greedy Strategy and Dynamic Programming (9 Hours)
Greedy Strategy- Control Abstraction, Fractional Knapsack Problem, Minimum Cost Spanning Trees – Prim's and Kruskal's Algorithm, Job sequencing with deadlines Dynamic Programming- Principle of Optimality, DP solution for traveling salesman and 0/1 Knapsack problems, Least Common Subsequence problem
Module 4: Backtracking & Branch and Bound (10 Hours)
Backtracking– State Space Tree, Fixed Tuple and Variable Tuple Formulation - Control Abstraction, Monte Carlo Method – N-Queens Problem, Sum of Subsets Branch and Bound Techniques– FIFO, LIFO, and LC Branch and Bound, Control Abstraction, 15-puzzle problem
Module 5: Sophisticated Algorithms (9 Hours)
Topological sort, string matching: KMP algorithm, Rabin-Karp algorithm, Introduction to Computational Complexity – complexity classes, Determinism and Non-determinismm, Approximation Algorithms – Planar Graph Colouring, Vertex cover

Text Books

1. Introduction to Algorithms – Cormen, Leiserson, Rivest, Stein – 3/e, PHI
2. Fundamentals of Computer Algorithms – Horowitz and Sahni, 2/e, Universities Press

Reference Books

1. Computer Algorithms – Introduction to Design and Analysis – Sara Baase& Allen Van Gelder, 3/e, Pearson Education
2. Introduction to the Design and Analysis of Algorithms – Anany Levitin, 3/e, Pearson
3. Foundations of Algorithms – Richard Neapolitan, 5/e, Jones and Barlett Learning

Course Contents and Lecture Schedule

No.	Topic	No. of Lectures
1	**Introduction to Algorithms**	**9 Hours**
1.1	Properties of a good Algorithm, Development of an Algorithm, Pseudo-code Conventions, Recursive Algorithms	3 Hours
1.2	Performance Analysis - Space and Time Complexity, Running Time Comparison - Worst, Best and Average Case Complexity, Asymptotic Notations, Common Complexity Functions	3 Hours
1.3	Recurrence Relations – Solving Recurrences using substitution and recurrence trees Amortized Complexity – aggregate analysis, cost-accounting and potential methods	3 Hours
2	**Divide and Conquer**	**8 Hours**
2.1	Divide and Conquer - Control Abstraction, Finding Maximum and Minimum	2 Hours
2.2	Binary Search, Strassen's Matrix Multiplication	3 Hours
2.3	Quick Sort, Merge Sort	3 Hours
3	**Greedy Strategy and Dynamic Programming**	**9 Hours**
3.1	Greedy Strategy- Control Abstraction, Fractional Knapsack Problem	2 Hours
3.2	Minimum Cost Spanning Trees – Prim's and Kruskal's Algorithm, Job sequencing with deadlines	2 Hours
3.3	Dynamic Programming- Principle of Optimality	2 Hours
3.4	DP solution for traveling salesman and 0/1 Knapsack problems, Least Common Subsequence problem	3 Hours
4	**Backtracking & Branch and Bound**	**10 Hours**
4.1	Backtracking– State Space Tree, Fixed Tuple and Variable Tuple	3 Hours

	Formulation - Control Abstraction	
4.2	Monte Carlo Method – N-Queens Problem, Sum of Subsets	2 Hours
4.3	Branch and Bound Techniques– FIFO and LIFO	2 Hours
4.4	LC Branch and Bound, Control Abstraction, 15-puzzle problem	3 Hours
5	**Sophisticated Algorithms**	**9 Hours**
5.1	Topological sort, string matching: KMP algorithm	3 Hours
5.2	Rabin-Karp algorithm, Introduction to Computational Complexity – complexity classes, Determinism and Non-determinismm	3 Hours
5.3	Approximation Algorithms – Planar Graph Colouring, Vertex cover	3 Hours