COURSE PLAN

A. COURSE OVERVIEW

Degree:	BE	Programme:	CSE
Academic Year:	2019-20	Semester:	4
Course Title:	**Design And Analysis of Algorithms**	Course Code:	18CS42
L-T-P-S:	3-2-0-0	Duration of SEE	3 Hrs
Total Contact Hours:	50 Hrs	SEE Marks:	60*
CIE Marks:	40	Assignment	1 / Module
Credits:	4		
Lesson Plan Author:	Harivinod N	Sign	Date 10/12/2019
Checked By:	Nithin Kurup U G	Sign	Date 10/12/2019

*The SEE will be conducted for 100 marks and proportionally reduced to 60 marks.

B. PREREQUISITES

- Programming in C and Data Structures (18CPS13/23)
- Data structures and Applications (18CS32)

C. COURSE DESCRIPTION

i) Course Outcomes

At the end of the course, the student will be able to;

- Estimate the computational complexity of different algorithms along with its representation notations.
- Design and analyze problem solving using divide and conquer strategy
- Apply greedy method to solve problems.
- Apply dynamic programming to solve problems using the solutions of similar subproblems.
- Design and apply backtracking technique for problem solving

ii) Relevance of the Course

- Design & Analysis of Algorithms Laboratory (18CSL47)
- Data Communication (18CS46)
- Computer graphics (18CS62)

iii) Applications areas

- Sorting and searching the information
- Geographic information systems (Maps. Graph theory)
- Information security (Encryption, Hashing, Cryptography)
- Link analysis (analysis of relationships between different entities)
- Data compression (JPEG, MPEG, Random number generation)
- Various stages in system and application programming.

Prepared by:	Checked by:	HOD

COURSE PLAN

D1. ARTICULATION MATRIX, CO v/s PO

Mapping of CO to PO												
	POs											
COs	1	2	3	4	5	6	7	8	9	10	11	12
1. Estimate the computational complexity of different algorithms along with its representation notations.	3	3	3	3	-	-	-	-	2	3	-	3
2. Design and analyse problem solving using devide and conquer stategy	3	3	3	3	-	-	-	-	2	3	-	2
3. Apply greedy method to solve problems.	3	3	3	3	-	-	-	-	2	3	-	2
4. Apply dynamic programming to solve problems using the solutions of similar subproblems.	3	3	3	3	-	-	-	-	2	3	-	2
5. Design and apply backracking technique for problem solving	3	3	3	3	-	-	-	-	2	3	-	2

Note: Mappings in the Tables D1 (above) and D2 (below) are done by entering in the corresponding cell the Correllation Levels in terms of numbers. For Slight (Low): 1, Moderate (Medium): 2, Substantial (High): 3 and for no correllation: " - ".

D2. ARTICULATION MATRIX, CO v/s PSO

Mapping of CO to PSO				
	PSOs			
COs	1	2	3	
1. Estimate the computational complexity of different algorithms along with its representation notations.	3	2	2	
2. Design and analyse problem solving using devide and conquer stategy	3	2	2	
3. Apply greedy method to solve problems.	3	2	2	
4. Apply dynamic programming to solve problems using the solutions of similar subproblems.	3	2	2	
5. Design and apply backracking technique for problem solving	3	2	2	

E. MODULE PLANS

MODULE - I

Title:	**Introduction**	Appr. Time:	10 Hrs
MO:			RBT
At the end of the Module, the student will be able to:			
1 Understand the sequence of steps involved in designing and analyzing an algorithm.			L1
2 Compute the efficiency of algorithm using asymptotic notation.			L3
3 Know the well known problem types where algorithms are used.			L4
4 Describe the fundamental data structures and with the manner in which these data structures can best be implemented.			L2
Lesson Schedule:			

COURSE PLAN

Lecture No.	Portion to be covered	CO
1	Introduction to algorithm and its specification	CO1
2	Analysis framework	CO1
3	Performance analysis: space complexity, time complexity	CO1
4	Asymptotic notations: big-oh, omega, theta and little-oh notation	CO1
5	Mathematical analysis of non-recursive algorithms with Examples	CO1
6	Mathematical analysis of recursive algorithms	CO1
7	Important Problem Types: Sorting, Searching, String processing,	CO1
8	Important Problem Types: Graph Problems, Combinatorial Problems.	CO1
9	Fundamental Data Structures: Stacks, Queues, Graphs	CO1
10	Fundamental Data Structures: Trees, Sets and Dictionaries.	CO1

Application Areas:

- In problem solving
- Analyzing the Performance of Algorithm / Program
- Simple searching and sorting techniques for small input size.

Review Questions / Questions Appeared in the Previous Years (CO):

	Introduction
1	Define algorithm. Discuss the criteria's that an algorithm must satisfy with an example. (CO1)
2	Define best case, worst case and average case efficiency. Give these efficiencies for sequential search. (CO1)
3	Explain space complexity and time complexity with an example. (CO1)
4	Explain with an example how a new variable count introduced in a program can be used to find the number of steps needed by a program to solve a particular problem instance. (CO1)
	Asymptotic Notations
5	Consider the following algorithm. (CO1) Algorithm GUESS (A[ ] [ ]) for i ← 0 to n – 1 for j ← 0 to i A [i] [j] ← 0 i) What does the algorithm compute? ii) What is basic operation? iii) What is the efficiency of this algorithm?
6	Explain asymptotic notations Big O, Big Ω and Big θ,that are used to compare the order of growth of an algorithm with example. (CO1)
7	Describe basic efficiency classes. (CO1)
8	Prove the following statements. (CO1) 1. $n^2 + 5n + 7 = \Theta(n^2)$ 2. $\frac{1}{2} n(n-1) = \Theta(n^2)$ 3. $\frac{1}{2} n^2+3n= \Theta(n^2)$ 4. $100n + 5 = `O(n^2)$ 5. $n^2 + n = O(n^3)$ 6. $5n^2 + 3n + 20 = O(n^2)$ 7. $n^3 + 4n^2 = \Omega(n^2)$
9	Define Little Oh. Compare the orders of growth of following functions (CO1) i) (½) n (n-1) and n^2

	ii) $3n+2$ and n^2
10	Prove that (CO1) If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$.
	Mathematical Analysis of Non-Recursive Algorithms
11	Explain general plan of mathematical analysis of non-recursive algorithms with example. (CO1)
12	Write the algorithm to find maximum element in the given array and explain the mathematical analysis of this non-recursive algorithm. (CO2)
13	Write the algorithm to check whether all the elements in the given array are distinct and explain the mathematical analysis of thisnon-recursive algorithm. Derive its worst-case time complexity (CO2)
14	Write the algorithm to perform matrix multiplication and explain the mathematical analysis of this non-recursive algorithm (CO1)
	Mathematical Analysis of Recursive Algorithms
15	Explain general plan of mathematical analysis of recursive algorithms with example. (CO1)
16	Illustrate mathematical analysis of recursive algorithm for Towers of Hanoi (CO1)
17	Illustrate mathematical analysis of recursive algorithmto find the factorial of a given number. (CO1)
18	State the recursive algorithm to count the bits of a decimal number in its binary representation. Give its mathematical analysis. (CO1)
19	Write a recursive function to find and print all possible permutations of a given set of n elements (CO1)
20	Solve the recurrence relation M(n) = 2M(n-1) +1 for n>1; M(1)=1 (CO1)
	Problem Types and Data structures
21	Briefly explain the important problem types coming under design and analysis of algorithms. (CO1)
22	Briefly explain important fundamental data structures used in algorithm design. (CO1)
23	Explain two common ways to represent the graph with example (CO1)

MODULE - II

Title: **Divide and Conquer**	Appr. Time:	10 Hrs
MO:		RBT
At the end of the Module, the student will be able to:		
1 Apply Divide & Conquer technique to solve problems of large input size		L4
2 Solve Recurrence relation and state complexity of solution.		L3
3 Illustrate how Binary Search, MaxMin algorithms works and compute its efficiency.		L2
4 Perform Merge sort, Quick sort and Strassen's matrix multiplication.		L3
5 Describe the decrease and conquer paradigm and concepts of task graph & work sequence using topological sorting.		L2
Lesson Schedule:		

COURSE PLAN

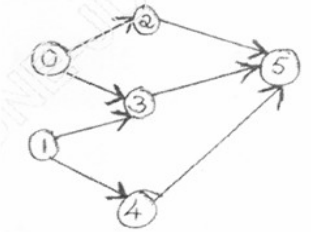
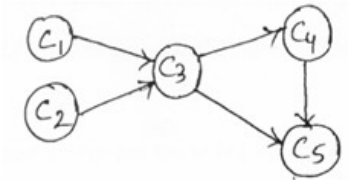
Lecture No.	Portion to be covered	CO
1	Introduction to Divide and Conquer, General method	CO2
2	Binary search	CO2
3	Recurrence equation for divide and conquer-1	CO2
4	Recurrence equation for divide and conquer-2	CO2
5	Finding the maximum and minimum	CO2
6	Merge sort	CO2
7	Quick sort	CO2
8	Strassen's matrix multiplication	CO2
9	Advantages and Disadvantages of divide and conquer.	CO2
10	Decrease and Conquer Approach: Topological Sort.	CO2

Application Areas:

- Solving problems of large input size.
- To divide a large task into sub task in a team and combine for solution

Review Questions / Questions Appeared in the Previous Years (CO):

	General Method, Solving Recurrence equations
1	Explain the general concept of divide and conquer method. Give the general algorithm DAndC(P) [Where P is the problem to solve] to illustrate this technique. (CO2)
2	List the advantages and disadvantages of Divide and Conquer. (CO2)
3	Find the upper bound of recurrences given below by substitution method. (CO2) i) T(n) = 2 T(n/2)+n ii) T(n) = T(n/2) + 1
4	Explain the general method of substitution method to solve the recurrence equation. (CO2)
5	State and explain master theorem to solve the recurrence equation. (CO2)
6	Consider Tower of Hanoi puzzle. Derive the recurrence relation for the total movement of disk. Solve the recurrence relation using substitution method (CO2)
	Binary Search, Max and Min
7	Show how binary search problem can be solved using the divide and conquer method. Write an algorithm for binary search and find average case efficiency. (CO2)
8	Design an algorithm to find the maximum and minimum element in a given list of n numbers using divide and conquer method. (CO2)
	Merge Sort, Quick sort
9	Write the algorithm for Merge Sort. Illustrate with an example. Derive the time efficiency (best case, average case, worst case) of the algorithm. (CO2)
10	Sort the following elements using merge sort. Write the recursion tree. (CO2) 70, 20, 30, 40, 10, 50, 60 Alternatively : Use D & C method to sort the numbers which divides problem size by considering position
11	Write an algorithm for sorting the numbers using Quick sort. Derive the best case, worst case, average case time efficiency of the algorithm. (CO2)
12	Apply quick sort algorithm to the following set of numbers. (CO2) 65, 70, 75, 80, 85, 60, 55, 50, 45
13	Illustrate the tracing of quick sort algorithm for the following set of numbers: (CO2)

	25, 10, 72, l8, 40, 11, 64, 58, 32, 9
14	Sort the list E, X, A, M, P, L, E in alphabetical order using the Quick Sort algorithm. Draw the tree of recursive call. (CO2)
	Stressen's Matrix Multiplication
15	Explain the algorithm which is used to perform matrix multiplication in an efficient way. OR Briefly explain Strassen's matrix multiplication and how its uses divide and conquer method. Obtain the time complexity. (CO2)
16	Explain topological sorting with example. (CO2)
17	Apply source removal method to obtain topological sort for the given graph.(CO2) 
18	Illustrate the topological sorting for the following graph. (CO2) 

MODULE - III

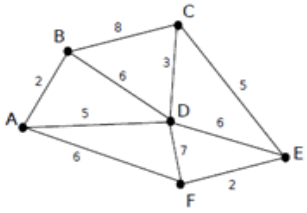
Title:	**Greedy Method**	Appr. Time:	10 Hrs
MO:			RBT
At the end of the Module, the student will be able to:			
1 Understand the concepts of GREEDY method and identify greedy criterion to solve given problem.			L2
2 Understand the significance of spanning trees and to find minimum cost spanning trees.			L3
3 Apply and implement Dijkstra's Algorithm to find single source shortest paths.			L3
4 Apply Huffman trees to minimize the weighted path length and Huffman codes which is used for lossless data compression.			L3
5 Illustrate Heaps and Heap Sort in the light of transform and conquer design.			L2
Lesson Schedule:			
Lecture No.	Portion to be covered		CO
1	Greedy Method: General method,		CO3
2	Coin Change Problem,		CO3
3	Knapsack Problem,		CO3
4	Job sequencing with deadlines		CO3
5	Minimum cost spanning trees: Prim's algorithm		CO3
6	Minimum cost spanning trees: Kruskal's algorithm		CO3
7	Single source shortest paths: Dijkstra's algorithm		CO3
8	Optimal Tree problem: Huffman trees and codes		CO3

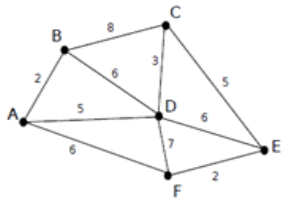
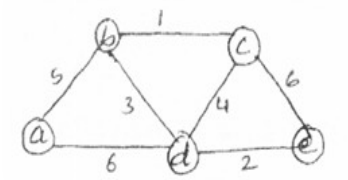
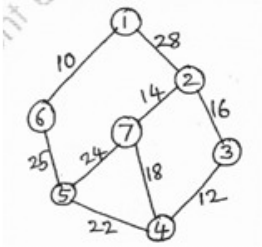
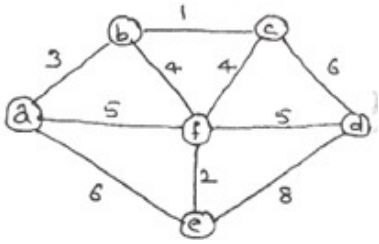
9	Transform and conquer approach: Heaps	CO3
10	Transform and conquer approach: Heap Sort	CO3

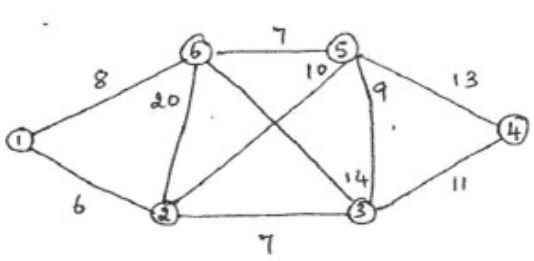
Application Areas:

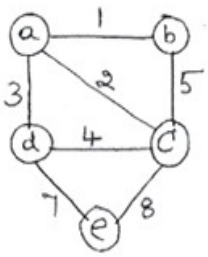
- Computer Networking,
- To solve the puzzles
- Game theory

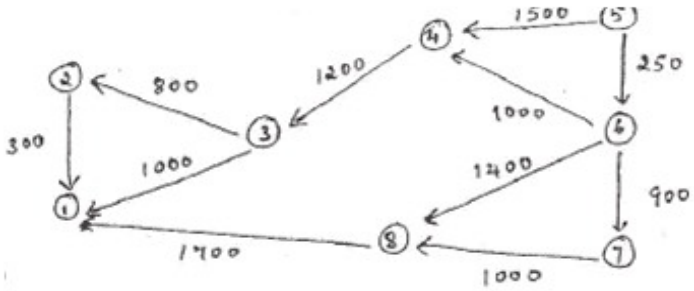
Review Questions / Questions Appeared in the Previous Years (CO):

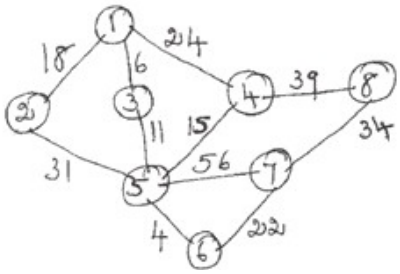
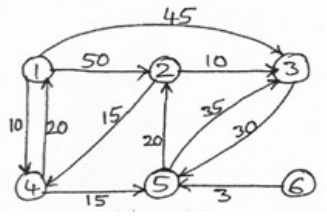
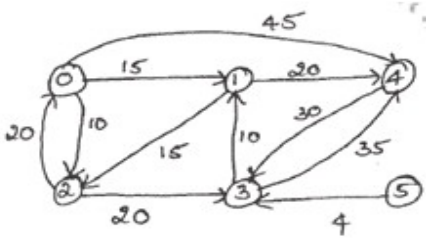
	Introduction
1	Define Optimal solution and feasible solution. (CO3)
2	Define coin change Problem. State the greedy method to solve the coin change problem. For 49 rupees, find the denominations with least no. of coins. The available denominations in rupees are { 1, 2, 5, 10} (CO3)
3	Define coin change problem. If coins available are of values { 2, 5, 3, 6 }, find the least denominations for a) 55 b)77 (CO3)
	Job Sequencing with deadlines
4	What is job sequencing with deadline problem. (CO3)
5	Obtain the optimal solution for the job sequencing problem with deadline where (CO3) n = 4 $(P_1, P_2, P_3, P_4) = (100, 10, 15, 27)$ (d1, d2, d3, d4) = (2, 1, 2, 1)
6	What is the solution generated by job sequencing when (CO3) n = 5 $(P_1, P_2, P_3, P_4, P_5) = (20, 15, 10, 5, 1)$ (d1, d2, d3, d4, d5) = (2, 2, 1, 3, 3)
7	Find solution generated by job sequencing problem with deadlines for 7 jobs given profits 3, 5, 20, 18, 1, 6, 30 and deadlines 1, 3, 4, 3, 2, 1, 2 respectively. (CO3)
8	Let n = 5, profits [10, 3, 33, 11, 40] and deadlines [3, 1, 1, 2, 2] respectively. Find the optimal solution using greedy algorithm. (CO3)
	Kanpsack Problem
9	What is knapsack problem? (CO3)
10	Obtain solution for a knapsack problem using greedy method for n = 3, capacity m=20 values 25, 24, 15 and weights 18, 15, 10 respectively. (CO3)
11	Apply greedy method to obtain an optimal solution to the knapsack problem given M = 60, (w1, w2, w3, w4, w5) = (5, 10, 20, 30, 40) (p1, p2, p3, p4, p5) = (30, 20, 100, 90, 160). Find the total profit earned. (CO3)
12	Solve the greedy knapsack problem where m=10, n=4, P=(40, 42, 25, 12), W=(4, 7, 5, 3) (CO3)
	Minimum Spanning Trees
13	Define MST. Write Prim's algorithm to construct minimum cost spanning tree. (CO3) 
14	Write Kruskals algorithm to construct MST. Show that the time efficiency is O(|ε|log|ε|).

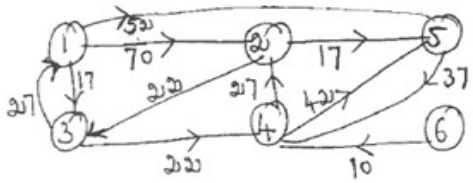
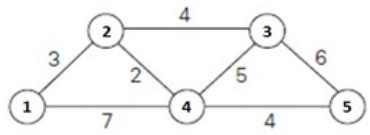
	(CO2,4)
15	Apply PRIMS and KRUSKAL algorithm for the following graph to get MST. Show the intermediate steps. (CO3) 
16	Obtain minimum cost spanning tree for the graph whose weight matrix is given below. (CO3) $\begin{bmatrix} 0 & 3 & \infty & 7 & \infty \\ 3 & 0 & 4 & 2 & \infty \\ \infty & 4 & 0 & 5 & 6 \\ 7 & 2 & 5 & 0 & 4 \\ \infty & \infty & 6 & 4 & 0 \end{bmatrix}$
17	Apply Prims and Kruskal algorithm to find the MST. Show the intermediate steps. (CO3) 
18	Apply Prims and Kruskal algorithm to find the MST. Show the intermediate steps. (CO4) 
19	Apply Prims and Kruskals algorithm to find the MST of the graph given below. (CO3) 
20	Apply Prim's algorithm to obtain a minimum spanning tree for the given weighted connected graph. (CO3)

<table>
<tr><td></td><td>


a)


b)


c)</td></tr>
<tr><td>21</td><td>Apply Prim's algorithm to obtain a minimum spanning tree for the given weighted connected graph.(CO3)

</td></tr>
<tr><td></td><td>Dijikstra's algorithm</td></tr>
<tr><td>22</td><td>Write Dijikstra's algorithm to find single source shortest path OR Write an algorithm to find single source shortest path. (CO3)</td></tr>
<tr><td>23</td><td>Apply Dijikstra's algorithm to find single source shortest path for the graph given below. Consider Node 6 as source. (CO3)

</td></tr>
<tr><td>24</td><td>Apply Dijikstra's algorithm to find single source shortest path for the graph given below. Source vertex is 5. (CO3)

</td></tr>
</table>

25	Determine the shortest paths from vertex 1 to all other vertices. (CO3) 
26	Apply Dijikras algorithm to find single source shortest path. Consider source vertex as (1) (CO3) 
	Hufman Coding
27	Explain Huffman coding algorithm. With an example show the construction of Huffman tree and generate the Huffman code using this tree. (CO3)
28	Construct the Huffman code for the following data. (CO3) symbol: A B C D _ frequency: 0.35 0.1 0.2 0.2 0.15 Also i) encode DAD and ADD ii) Decode 10011011011101
29	Construct a Huffman code for the following data:(CO3) symbol: A B C D _ frequency: 0.4 0.1 0.2 0.15 0.15 Encode ABACABAD using the code. Decode 100010111001010
	Heap, Heap Sort
30	Explain Bottom-up heap construction algorithm with an example. Give the worst case efficiency. (CO3)
31	Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by the bottom-up algorithm. (CO3)
32	Sort the following lists by heapsort by using the array representation of heaps.5, 2, 4, 1, 3 (in increasing order) (CO3)
33	Sort the array 2, 9, 7, 6, 5, 8 by heapsort. (CO3)

MODULE - IV

Title:	**Dynamic Programming**	Appr. Time:	10 Hrs
MO:			RBT
At the end of the Module, the student will be able to:			
1 Develop dynamic recurrence relation for solving problems with overlapping sub problems.			L2
2 Solve all -pair shortest path problem using Floyd's algorithm and transitive closure problem using Warshall's method.			L3
3 Describe the method of finding optimal binary search tree by dynamic programming.			L2
4 Solve Knapsack problem and Travelling Salesman problem using dynamic programming technique.			L3
5 Design and implement Bellman-Ford algorithm to compute shortest paths.			L5
Lesson Schedule:			

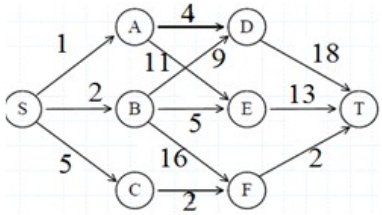
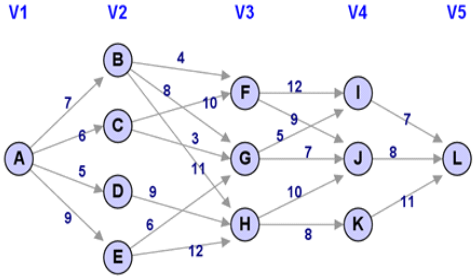
COURSE PLAN

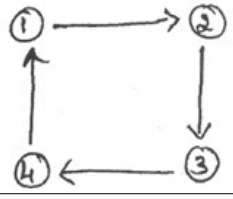
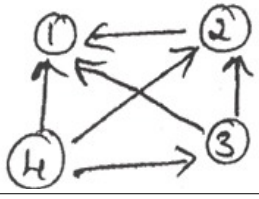
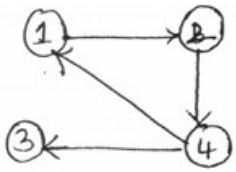
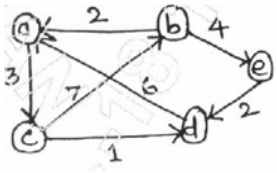
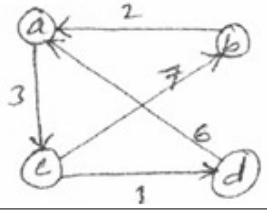
Lecture No.	Portion to be covered	CO
1	Dynamic programming: General method with examples	CO4
2	More examples	CO4
3	Multistage graphs	CO4
4	Transitive closure: Warshall's algorithm,	CO4
5	All pairs shortest paths: Floyd's algorithm	CO4
6	Optimal Binary Search Trees,	CO4
7	Knapsack problem,	CO4
8	Bellman-Ford Algorithm	CO4
9	Travelling Sales Person problem,	CO4
10	Reliability design	CO4

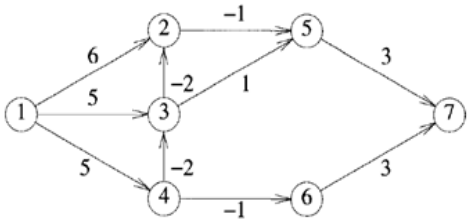
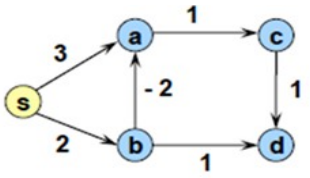
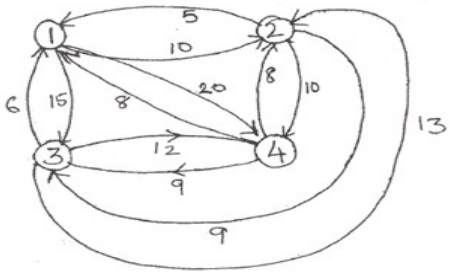
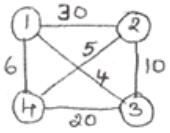
Application Areas:

- Solving problems with overlapping sub problems
- TSP model in PCB design and networking problems

Review Questions / Questions Appeared in the Previous Years (CO):

	Introduction to DP
1	Briefly explain the concept of dynamic programming with example. (CO4)
	Multistage Graph
2	Explain multistage graph with example. (CO4)
3	Write multistage graph algorithm using forward approach. (CO4)
4	Write multistage graph algorithm using backward approach. (CO4)
5	Find the shortest path from S to T in the following multistage graph using dynamic programming. Use forward approach to solve the problem. (CO4) 
6	Find the shortest path from A to L, in the following multistage graph, using dynamic programming. Use forward approach to solve the problem. (CO4) 
	Transitive Clossure – Warshalls algorithm
7	Define transitive closure. Write Warshall's algorithm to compute transitive closure. Find its efficiency. (CO4)
8	Generate transitive clossure of the graph given below. (CO4)

	 
9	Trace the following graph using Warshalls algorithm to find transitive clossure. (CO4) 
	Floyds Algorithm – All Pair Shortest Path
10	What is dynamic programming. Explain how would you solve all pair shortest path problem using dynamic programming. (CO4)
11	Apply floyds algirthm to find the all pair shortest path for the graph given below.(CO4)  $\begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix}$
12	Apply floyds algirthm to find the all pair shortest path for the graph given below.(CO4) 
	Optimal Binary Search Tree
13	Define optimal Binary Serach Tree. Write a pseudocode to find an optimal binary search tree using dynamic programming.(CO4)
14	Find the optimal binary search tree for the keys given below. (CO4) key A B C D probability 0.1 0.2 0.4 0.3
15	Find the optimal binary search tree for the keys A, B, C, D, E with search probabilities 0.1, 0.1, 0.2, 0.2, 0.4 respectively. (CO4)
	0/1 knapsack problem using dynamic programming.
16	Solve the following instance of 0/1 knapsack problem using dynamic programming. Knapsack capacity is W=5 and n=4 Item | Weight | Value 1 | 2 | 12 2 | 1 | 10 3 | 3 | 20 4 | 2 | 15 Capacity W = 5 (CO4)
17	Solve the Knapsack instance n=3, {w1, w2, w3} = {1, 2, 2} and {p1, p2, p3} = {18, 16, 6} and M=4 by dynamic programming. (CO4)

	Bell-ford algorithm
18	Explain Bellman-ford algorithm to find shortest path from single source to all destinations for a directed graph with negative edge cost. (CO4)
19	Apply Bellman-ford algorithm to the graph given below, to find shortest path to all the vertices from vertex 1. (CO4) 
20	Apply Bellman-ford algorithm to the graph given below, to find shortest path to all the vertices from vertex s. (CO4) 
	Travelling Salesman Problem
21	For the given graph obtain optimal cost tour using dynamic programming.(CO4) 
22	Solve the following TSP which is represented as a graph using dynamic programming. Start city is 1. (CO4) 
23	Solve the following TSP which using dynamic programming.(CO4) $\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$ starting city 1
24	Solve the following TSP problem using dynamic programming. The start city is 1.(CO4) $= \begin{pmatrix} 0 & 2 & 9 & 10 \\ 1 & 0 & 6 & 4 \\ 15 & 7 & 0 & 8 \\ 6 & 3 & 12 & 0 \end{pmatrix}$
	Reliability Design
25	Design a 3-stage system with device types A, B, C whose costs are 30, 15, 20 and reliability

	are 0.9, 0.8, 0.5 respectively. Budget available is 105. Design a system with highest reliability. (CO4)
26	Design a 3-stage system with device types A, B whose costs are 30, 20 and reliability are 0.7, 0.5 respectively. Budget available is 110. Design a system with highest reliability.(CO4)

MODULE - V

Title:	**Backtracking**	Appr. Time:	10 Hrs
MO:			RBT
At the end of the Module, the student will be able to:			
1Illustrate Backtracking, Branch and Bound technique to solve Combinatorial problems.			L3
2 Familiarize with approximation algorithm which returns near optimal solutions			L3
2 Solve 0/1 Knapsack problem using Branch and Bound technique.			L3
4 Understand NP-Complete and NP-Hard problems.			L2

Lesson Schedule:

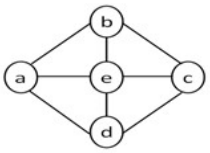
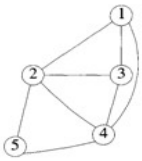
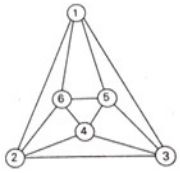
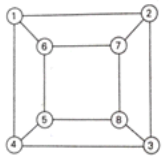
Lecture No.	Portion to be covered	CO
1	Backtracking: General method	CO5
2	N-Queens problem	CO5
3	Sum of subsets problem	CO5
4	Graph coloring, Hamiltonian cycles	CO5
5	Branch and Bound: Assignment Problem, Travelling Sales Person problem	CO5
6	0/1 Knapsack problem	CO5
7	LC Branch and Bound solution	CO5
8	FIFO Branch and Bound solution	CO5
9	NP-Complete and NP-Hard problems: Basic concepts, non-deterministic algorithms,	CO5
10	P, NP, NP-Complete, and NP-Hard classes	CO5

Application Areas:

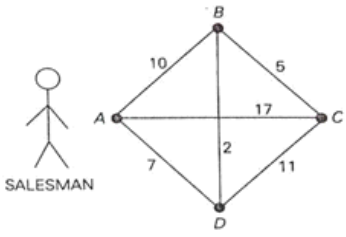
- Combinatorial problems
- Approximation algorithm

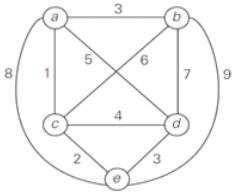
Review Questions / Questions Appeared in the Previous Years (CO):

	Backtracking, General Method
1	What is backtracking. Give the general Procedure. OR Write the pseudocode for backtracking algorithm (CO5)
	N-Queens Problem
2	Give the problem statement of n-queens problem. Explain the solution for 4-queens problem using state space tree. (CO5)
	Sum of Subset problem
3	Let w = {3, 5, 6, 7} and m =15. Find all possible subsets of w that sum to m. Draw the state space tree that is generated. (CO5)
4	Apply backtracking to solve the following instance of the subset-sum problem : S={1,3,4,5} and d=11. Draw the state space tree. (CO5)
5	Apply backtracking to solve the following instance of the subset-sum problem : S={5,10,12,13,15,18} and d=30. Give all possible solutions. (CO5)
	Graph Coloring

6	Define Graph coloring problem. Apply backtracking to solve the 3-cloring problem for the graph given below. (CO5) 
7	Apply backtracking based graph coloring algorithm for the graph given below with m=4. Give state space tree showing first 3 valid assignments. (CO5) 
8	Draw the portion of the state space tree for m – colorings of a graph when n=4 and m=3, (CO5)
	Hamiltonian cycles
9	What is Hamiltonian cycle? Give the backtracking based algorithm to find the Hamiltonian cycle in the graph. Write the functions used to generating next vertex and for finding Hamiltonian cycles. (CO5)
10	Apply the backtracking to the problem of finding Hamiltonian cycle in the following graphs. (CO5) (a)  (b) 
	Branch-and-bound assignment problem
11	What branch and bound method. How it is different from backtracking. (CO5)
12	Apply best-first branch and bound method for the following instance of assignment problem to find the optimal solution. Give the complete state space tree. (CO5)

Job 1	Job 2	Job 3	Job 4	
9	2	7	8	Person a
6	4	3	7	Person b
5	8	1	8	Person c
7	6	9	4	Person d

	Branch-and-bound Travelling salesperson problem
13	Explain how TSP can be solved using branch and bound technique. (CO5)
14	Apply the branch-and-bound algorithm to solve the travelling sales man problemfor the following graph. Consider start city as A. Give the statespace tree. (CO5) 
15	Apply the branch-and-bound algorithm to solve the travelling sales man problemfor the

	following graph. Start city is *a*. Give the statespace tree. (CO5) 
	0/1 Knapsack problem
16	Explain LC branch & bound and FIFO branch and bound. (CO5)
17	With the help of a state space tree, solve the following instance of Knapsack problem by the branch and bound algorithm. Knapsack Capacity W = 10 (CO5) Item No. 1 2 3 4 Weight 4 7 5 3 Value 40 42 25 12
18	With the help of a state space tree, solve the following instance of Knapsack problem by the branch and bound algorithm. Knapsack Capacity W = 15 Item No. 1 2 3 4 5 6 Weight 5 7 2 4 5 1 Value 40 35 18 4 10 2
19	Apply Least Cost Branch and Bound (LCBB) method for the following instance of 0/1 Knapsack problem to get the optimal solution. Knapsack Capacity W = 15(CO5) Item No. 1 2 3 4 Weight 2 4 6 9 Value 10 10 12 18
20	Apply FIFO Branch and Bound method for the following instance of 0/1 Knapsack problem to get the optimal solution.Knapsack Capacity W = 15 (CO5) Item No. 1 2 3 4 Weight 2 4 6 9 Value 10 10 12 18
	NP-Complete, NP Hard Problems
21	Define deterministic and non deterministic algorithms. (CO5)
22	Explain the following with examples (CO5) a. Class P Problems b. Class NP Problems c. NP complete problem d. NP hard problem.

F. INTERNAL ASSESSMENT TEST MODEL QUESTION PAPER

Dept: CSE	Sem / Div: 4 CS A & B	Course: Design and Analysis of Algorithms	Course Code: 18CS42
Date: dd/mm/yyyy	Time: 90 Min.	Max Marks: 50	Elective: N

Note: Answer any 2 full questions, choosing one full question from each part.

QN		Questions	Marks	RBT	CO's
PART A					
1	a	Define best case, worst case and average case efficiency. Give these efficiencies for sequential search.	8	L2	CO2

	b	Briefly explain the important problem types coming under design and analysis of algorithms.	8	L2	CO1
	c	Illustrate mathematical analysis of recursive algorithm for Towers of Hanoi	9	L3	CO2
		OR			
2	a	Explain asymptotic notations Big O, Big Ω and Big θ,that are used to compare the order of growth of an algorithm with example.	8	L2	CO2
	b	Explain two common ways to represent the graph with example	8	L2	CO1
	c	Write the algorithm to check whether all the elements in the given array are distinct and explain the mathematical analysis of thisnon-recursive algorithm. Derive its worst-case time complexity	9	L3	CO2
		PART B			
3	a	Explain the general concept of divide and conquer method. Give the general algorithm DAndC(P) [Where P is the problem to solve] to illustrate this technique.	8	L2	CO1,4
	b	Show how binary search problem can be solved using the divide and conquer method. Write an algorithm for binary search and find average case efficiency.	8	L2	CO3
	c	Write an algorithm for sorting the numbers using Quick sort. Derive the best case, worst case, average case time efficiency of the algorithm.	9	L2	CO3
		OR			
4	a	Explain topological sorting with example.	8	L2	CO1,4
	b	Design an algorithm to find the maximum and minimum element in a given list of n numbers using divide and conquer method.	8	L2	CO3
	c	Sort the following elements using merge sort. Write the recursion tree. 70, 20, 30, 40, 10, 50, 60	9	L2	CO3

G. CONTINUOUS INTERNAL EVALUATION

Evaluation	Weightage in Marks
IA Test – 1	30
IA Test – 2	30
IA Test – 3	30
Marks for Different Components : Assignments ,Additional Assesment Tool (AAT)- (Seminar/ CLT/ Quiz/ Mini Project)	10 To deepen student's understanding and increase his/her confidence in the topics studied. Further, to improve the oral, written skills and engineering aptitudes.