U.S.N.										

B.M.S. College of Engineering, Bengaluru-560019

Autonomous Institute Affiliated to VTU

June 2025 Semester End Main Examinations

Programme: B.E. **Semester: IV**

Branch: CSE(DS)/AI&DS/CSE(IoT) **Duration: 3 hrs.**

Course Code: 23DC4PCDAA **Max Marks: 100**

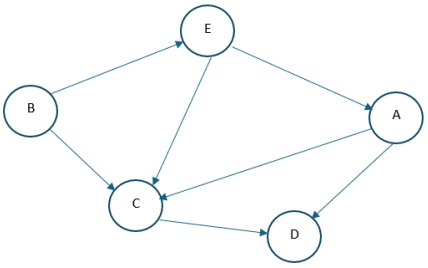
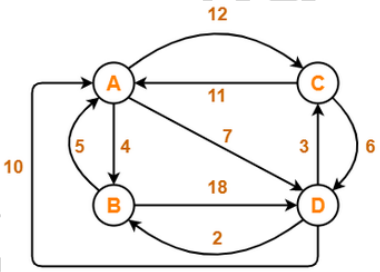
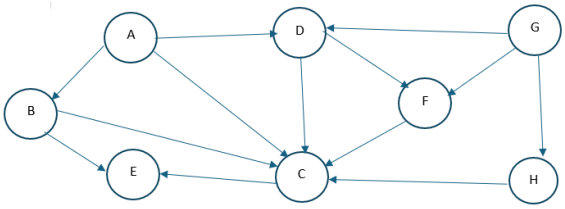
Course: Design and Analysis of Algorithms

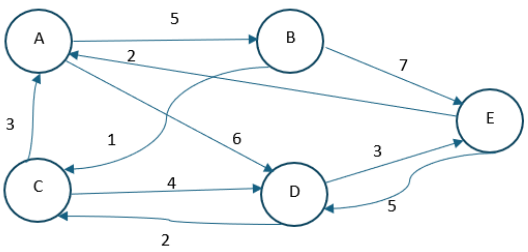
Instructions: 1. Answer any FIVE full questions, choosing one full question from each unit.
2. Missing data, if any, may be suitably assumed.

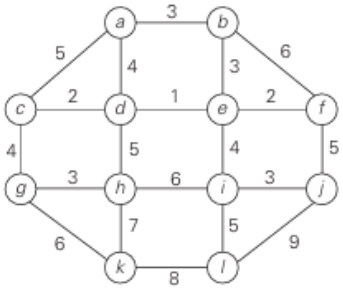
Important Note: Completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages. Revealing of identification, appeal to evaluator will be treated as malpractice.

		UNIT - I	CO	PO	Marks
1	a)	Explain the Fundamentals of Algorithmic problem solving with a neat diagram.	CO1	PO1	**06**
	b)	Find gcd(31415, 14142) by applying Euclid's algorithm.	CO1	PO1	**04**
	c)	Explain different asymptotic notations used to represent the time complexities and also determine whether the following assertions are true or false. i). $n(n+1) / 2 \in O(n^3)$ ii). $n(n+1) / 2 \in \Theta(n^3)$ iii). $n(n+1) / 2 \in O(n^2)$ iv). $n(n+1) / 2 \in \Omega(n)$	CO2	PO2	**10**
		OR			
2	a)	Suppose you have algorithms with the three running times listed below. (Assume these are the exact running times.) How much slower do each of these algorithms get when you (i) double the input size (ii) increase the input size by one? i) n^3 ii) $100n^2$ iii) n log n	CO2	PO2	**06**
	b)	Design an algorithm to perform linear search on an array of n elements. Analyze its time complexity in the best, worst and average cases.	CO1	PO1	**06**

	c)	Analyze the time complexity for the following codes by applying the step-by-step procedure for finding out the time complexity of non-recursive and recursive algorithms.
		i)
		int add(int A[], int n)
		{
		int sum = 0, i;
		for(i = 0; i < n; i++)
		sum = sum + A[i];
		return sum;
		}
		ii)
		Algorithm GE(A[0..n−1,0..n])
		for i ←0 to n−2 do
		for j ←i+1to n−1 do
		for k ←i to n do
		A[j,k]←A[j,k]−A[i,k]∗A[j,i]/A[i,i]
			CO2	PO2	**08**

			UNIT - II			
	3	a)	Apply source removal method to solve the following topological sorting problem. 	*CO1*	*PO1*	**08**
		b)	Solve the following assignment problem using Exhaustive Search technique, whose matrix for assigning four jobs to four persons are given: J1 J2 J3 J4 P1 9 2 7 8 P2 6 4 3 7 P3 5 8 1 8 P4 7 6 9 4	*CO1*	*PO1*	**06**
		c)	Solve Travelling Salesman problem using Exhaustive Search technique where following graph shows a set of cities and cost between every pair of cities. Show step by step procedure in finding out the route that gives minimum cost. 	*CO1*	*PO1*	**06**
			OR			
	4	a)	Find the BFS and DFS traversals starting from vertex **A** for the following Graph. Also, write the BFS Algorithm. What is the time complexity of the algorithm? 	*CO1*	*PO1*	**10**
		b)	i)What is the largest number of key comparisons made by binary search in searching for a key in the following array? 3 14 27 31 39 42 55 70 74 81 85 93 98 ii) List all the keys of this array that will require the largest number of key comparisons when searched for by binary search.	*CO2*	*PO2*	**04**

			iii) Find the average number of key comparisons made by binary search in a successful search in this array. Assume that each key is searched for with the same probability. iv) Find the average number of key comparisons made by binary search in an unsuccessful search in this array. Assume that searches for keys in each of the 14 intervals formed by the array's elements are equally likely.			
		c)	Determine the number of character comparisons made by the brute-force algorithm in searching for the pattern **NOT** in the text **NOBODY_NOTICED_HIM**	*CO2*	*PO2*	**06**
			UNIT - III			
	5	a)	Apply Merge Sort to sort the characters in the list **{P, A, C, G, E, U, H, B, Q}** in alphabetical order. Derive the time complexity of Merge Sort.	*CO3*	*PO3*	**10**
		b)	Apply Horspool's string matching algorithm to search for the Pattern string: **"LEADER"** in the Text string: **"JIMY_HAILED_THE_LEADER_TO_STOP"**. Clearly demonstrate all the steps. Write an algorithm for the same.	*CO3*	*PO3*	**10**
			OR			
	6	a)	Construct a Max Heap for the following list of keys and sort the list using Heap Sort technique. Write the algorithm for Max Heap. **{2,9,7,6,5,8}**	*CO3*	*PO3*	**10**
		b)	Demonstrate in terms of time complexity, how multiplication of two large integers using Divide and Conquer technique is efficient.	*CO2*	*PO2*	**5**
		c)	Apply Horner's rule to evaluate the following polynomial: $5x^4 + 2x^3 - 3x^2 + x - 7$ at the point x=3.	*CO1*	*PO1*	**5**
			UNIT - IV			
	7	a)	Apply Floyd's algorithm to compute all pairs shortest path for the following graph. 	*CO3*	*PO3*	**10**
		b)	Construct a Huffman tree and find the code word for the following data: Symbol: A B C D _ Frequency: 0.4 0.1 0.2 0.15 0.15	*CO3*	*PO3*	**10**

			Using above code, Encode the text **ABACABAD** and decode the text **10011011011101.**			
			OR			
	8	a)	Analyze time efficiency of Prim's algorithm. Apply Prim's algorithm to find the minimum cost spanning tree for the graph shown below:  	*CO3*	*PO3*	**10**
		b)	Solve the following instances of Knapsack problem by Dynamic Programming technique. Assume Knapsack capacity as 5. Show the steps clearly for computation of the values. Item 1 2 3 4 Weight 2 1 3 2 Profit 12 10 20 15	*CO3*	*PO3*	**10**
			UNIT - V			
	9	a)	Find any one solution to 4-queens problem using backtracking. Draw the state-space tree	*CO1*	*PO1*	**8**
		b)	Describe about approximation algorithm for Vertex Cover	*CO1*	*PO1*	**6**
		c)	Solve the following instance of 0/1 Knapsack problem using Branch and bound with capacity C=10. Items= {1,2,3,4} Weights= {4,7,5,3} and Values= {$40, $42, $25, $12}	*CO3*	*PO3*	**6**
			OR			
	10	a)	Describe the P and NP class problems with examples. Illustrate NP Completeness proof by Reduction.	*CO1*	*PO1*	**10**
		b)	Write the state-space tree for the following scenario: Given the set **S={3,5,6,7}** and the required sum value **d=15**. Find the possible subsets with sum value d using Backtracking. Also, write the pseudocode for the same.	*CO3*	*PO3*	**10**

****