U/D/TY

Reg. No: 23BCE0969

Final Assessment Test – April 2025

Course: BCSE204L - Design and Analysis of Algorithms

Class NBR(s): 1459/1472/1489/1496/1504/1510/1518/1525/1530/1534/1537/1543/1547/1555/1561/1567/1578/1586/1594/1600/1608

Slot: A1+TA1

Time: Three Hours

Max. Marks: 100

- KEEPING MOBILE PHONE/ANY ELECTRONIC GADGETS, EVEN IN 'OFF' POSITION IS TREATED AS EXAM MALPRACTICE
- DON'T WRITE ANYTHING ON THE QUESTION PAPER

Answer ALL Questions
(10 X 10 = 100 Marks)

1. The **Karatsuba algorithm** is a divide-and-conquer approach for fast integer multiplication. Break down the **Karatsuba algorithm** into its constituent stages: problem decomposition, recursive subproblems, and recombination. Derive and analyze its time complexity, comparing it with the standard multiplication method. Illustrate the algorithm for the multiplication of two integers 21 and 143.

2. Construct the optimal Huffman code for a given set of character frequencies derived from the first 8 prime numbers.

Character	a	b	c	d	e	f	g	h
frequency	2	3	5	7	11	13	17	19

Derive the time and space complexity of the Huffman code. Define the prefix free property and the need for prefix free code in Huffman coding. Show that Huffman coding takes less space as compared to fixed size code.

3. A logistics company wants to optimize the delivery route for a salesperson who must visit four cities exactly once before returning to the starting city. The company has recorded the travel distances between cities in the given matrix in Table 2.

As a software engineer, develop an efficient route-planning algorithm using **Dynamic Programming (DP)**.

a) Formulate the **Traveling Salesman Problem (TSP)** using the DP approach and explain its **optimal substructure.**

b) Using the **Held-Karp Algorithm (Dynamic Programming)**, compute the minimum travel cost for the given distance matrix and show step-by-step calculations.

c) Compare the **time and space complexity** of the dynamic programming approach with the brute-force approach.

	A	**B**	**C**	**D**
A	0	4	1	9
B	3	0	6	4
C	4	1	0	2
D	6	5	-4	0

Table 2: Distance matrix between cities.

4. A delivery company wants to optimize the selection of delivery packages to maximize profit while ensuring that the total weight of selected packages does not exceed the weight limit of their delivery van. Each package has a specific weight and value associated with it.

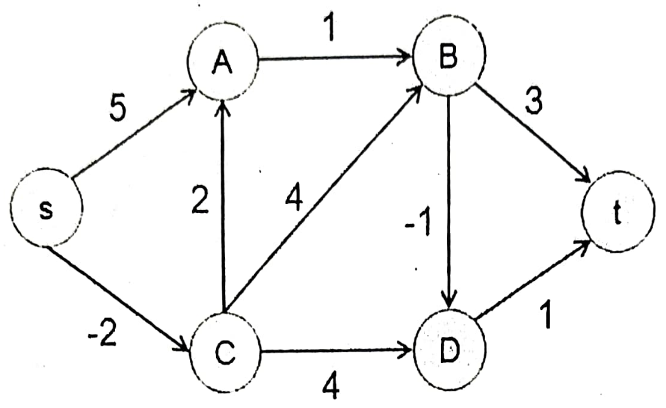
As an algorithm specialist, apply the **FIFO Branch & Bound** approach to solve the **0-1 Knapsack Problem** with the following items:

Package	**Weight (kg)**	**Value (Rs.)**
1	2	40
2	3	50
3	4	70
4	5	80

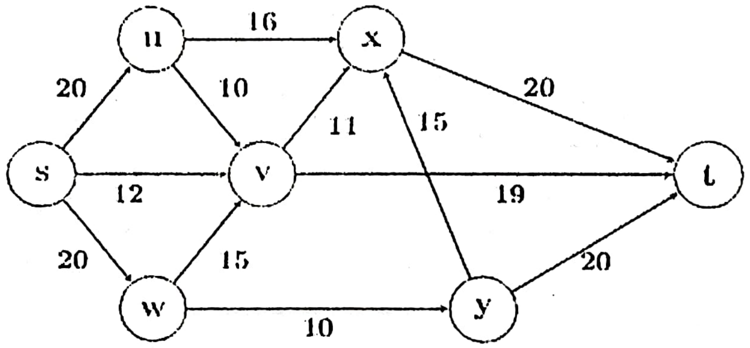
Contrast the Backtracking and Branch & Bound paradigms in solving the 0-1 Knapsack Problem, highlighting their differences in terms of search strategy, pruning mechanism, efficiency, and applicability.

5. As a bioinformatics researcher, you are tasked with developing an efficient algorithm to identify specific genetic markers in DNA sequences. You need to search for the genetic marker pattern P = "**ACGTACGT**" within the target DNA sequence T = "**GCACGTACGTGACGTACGTA**". Apply the KMP and Rabin-Karp algorithm for finding all occurrences of the pattern P in the given text T. Trace the execution of your chosen algorithm when searching for pattern P in sequence T, highlighting key comparisons and shifts. For Rabin Karp, consider A=1, C=2, G=3, T=4, base 4, and mod 101 and show the number of spurious hits. Compare the efficiency of your solution by calculating the total number of character comparisons performed by both algorithms for this specific example. Identify the algorithm that runs faster as compared to the other.

6. Apply the Bellman Ford algorithm to compute the all-pairs shortest paths for the given weighted graph. Show the step-by-step execution of the algorithm. Discuss the strength and weakness of the algorithm.



7. A logistics company is planning the most efficient way to transport goods from a central warehouse (*source* **s**) to a major distribution center (destination ***t***) through a network of cities: *u, v, w, x,* and *y*. Each road connecting these cities has a specific maximum carrying capacity, representing the amount of goods that can be transported at a time, represented by the graph given below.



Currently, no goods are flowing through the network. The company wants to maximize the shipment flow using the **Ford-Fulkerson algorithm** while following these conditions:

A. In the first step, the route with the **longest augmenting path** (largest number of cities) should be chosen for transportation.
B. In the second step, the route with the **highest residual capacity** should be prioritized.

In your answer, clearly show (i) the original graph with all capacities filled up, (ii) the augmenting path chosen in the first step (write this as a sequence of vertices, do not mark on the graph), (iii) the flow network after the first augmentation, (iv) the corresponding residual graph after the first augmentation, (v) the augmenting path chosen in the second step, (vi) the flow network after the second augmentation, (vii) the corresponding residual graph after the second augmentation. viii) Also write the time and space complexity analysis of the algorithm.

8.a) Given the set of points **P={(1,2),(3,5),(4,4),(6,2),(7,6),(8,3)}** and line segments formed by connecting $(1,2) \rightarrow (4,4), (3,5) \rightarrow (7,6), (6,2) \rightarrow (8,3), and\ (3,5) \rightarrow (6,2)$ Apply the sweep line algorithm to detect intersections among these segments. Demonstrate the step-by-step execution of the algorithm and construct the event queue and status structure at each step. Finally, analyze its time complexity and discuss its significance in geometric algorithms.

OR

8.b) Given the set of points:

$$P = \{(1,2), (3,5), (4,4), (6,2), (7,6), (8,3), (2,7), (5,1)\}$$

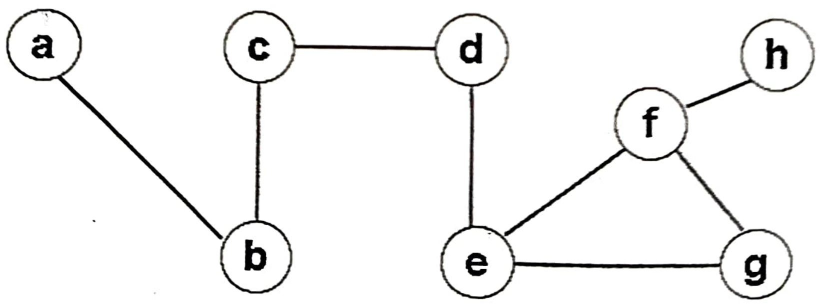
Apply **Graham's Scan** and determine the convex hull of the given points. Illustrate the step-by-step execution of the algorithm, showing the intermediate hull formations. Compare computational efficiency of **Graham's Scan** and **Jarvis' March** in terms of **time complexity** and discuss the conditions under which one method may be preferred over the other.

9. Apply the **Randomized QuickSort algorithm** to sort the array **[9, 4, 7, 2, 6, 3, 8]**. Show the step-by-step execution, including the selection of random pivots, partitioning, and recursive calls. Analyze its expected time complexity and discuss how randomization helps in avoiding the worst-case scenario.

10. a) Provide a summarized explanation of the complexity classes P, NP, NP-Hard, and NP-Complete, highlighting their role in assessing the hardness of real-world computational problems concerning algorithmic efficiency. Additionally, demonstrate the NP-completeness of the 3-SAT problem by formally reducing SAT to 3-SAT, outlining the key steps involved in the proof.

OR

10. b) Provide a summarized explanation of the Vertex Cover Problem, its computational complexity, and its significance in real-world applications. Discuss why it is classified as NP-complete and demonstrate the proof that Vertex Cover has a 2-approximation algorithm, outlining the key steps involved in the approximation technique. Obtain a vertex cover for the following graph.



⇔⇔⇔ U/D/TY ⇔⇔⇔