B.Tech – AI&DS (R2023)

Year/Semester: II/III

2302AS301 – Design and Analysis of Algorithms

	PART B (Marks : 20)		Marks	CO	BTL
	Course Outcome : 01				
1	(a)	An e-commerce company is tasked with optimizing delivery routes for a fleet of drones. Currently, a simple greedy approach is employed, always dispatching a drone to the closest unvisited delivery point. a. Analyze the time complexity of the current greedy approach using Big O notation. State any assumptions made about the number of delivery points (N). (5 marks) b. Identify a specific scenario where this greedy approach would lead to a suboptimal solution (i.e., a much longer total route or more fuel consumption) compared to an ideal route. Provide a small example (e.g., 3-5 delivery points) to illustrate the point. (7 marks) c. Briefly propose an alternative algorithmic approach (e.g., a known algorithm or a high-level strategy) that could potentially yield a more optimal solution for drone delivery routes, even if it is com0putationally more intensive. Explain why the proposed approach might be better. (8 marks)	20	CO1	L3
2	(a)	Consider the development of a secure system requiring rapid verification of the integrity of large files downloaded by users. One proposed solution involves employing a hashing technique to generate a unique "fingerprint" for each file. a. Explain the fundamental concept of a hash function and its role in data integrity verification. (5 marks) b. Describe a scenario where a collision in a hash function could lead to a security vulnerability or data corruption within such a system. (7 marks) c. Propose a method or a characteristic that a robust hash function should possess to minimize the likelihood and impact of collisions in this file integrity context. (8 marks)	20	CO1	L3
3	(a)	Consider the problem of performing a depth-first search (DFS) on a binary tree to find a specific node. A common way to implement DFS is using a recursive algorithm. a. Write down a recurrence relation that describes the time complexity of a recursive DFS algorithm for a binary tree in the worst-case scenario (e.g., a skewed tree where the target node is at the deepest level). Define your variables clearly. (7 marks) b. Using the substitution method, solve the recurrence relation derived in part (a) to find the Big O efficiency class of the recursive DFS algorithm. Show your steps. (10 marks) c. Briefly explain why the time complexity might differ in the best-case scenario compared to the worst-case scenario for DFS. (3 marks)	20	CO1	L3
4	(a)	A new data analytics platform is under development, necessitating the selection of an efficient sorting algorithm for large datasets. Two candidates are Merge Sort and a hypothetical "Simple Sort" that exhibits N3 comparisons in the worst case, where N represents the number of elements. a. State the asymptotic notation (Big O) for the worst-case time complexity of Merge Sort. (3 marks) b. Explain the practical implications of selecting an algorithm with an efficiency class of O(N3) versus O(NlogN) when processing a dataset of 1 million elements	20	CO1	L3

		($N=10^6$). Be specific about the difference in operations. (10 marks) c. In what very specific and highly constrained scenario might the "Simple Sort" (or any algorithm with a higher polynomial complexity) be considered acceptable over Merge Sort, even if theoretically less efficient? (Think about extremely small N). (7 marks)			
5	(a)	The task is to write a function that identifies the maximum product of two distinct numbers within a given unsorted array of positive integers. The solution must be non-recursive. a. Describe a step-by-step non-recursive algorithm to find the maximum product of two distinct numbers in an unsorted array. (8 marks) b. Perform a mathematical analysis of the non-recursive algorithm to determine its time complexity in terms of Big O notation. Justify the answer by counting the dominant operations. (10 marks) c. Could this problem be solved with a single pass through the array? Briefly explain why or why not. (2 marks)	20	CO1	L3
6		The backend for a new online multiplayer game is being designed. A critical component is a leaderboard system that requires frequent updates and rapid display of top players. a. If the leaderboard needs to maintain a sorted list of all players (potentially millions) and allow for quick insertion/update of scores, what abstract data type and underlying data structure would be initially considered for this task? Justify the choice based on its efficiency for these operations. (7 marks) b. Describe a scenario where the chosen data structure/algorithm in part (a) might become a bottleneck in terms of performance as the number of players scales up drastically. (7 marks) c. Propose a different algorithmic approach or a modification to the initial idea that could mitigate the bottleneck identified in part (b), even if it introduces new complexities or trade-offs (e.g., slightly delayed updates for faster reads). (6 marks)	20	CO1	L3
	Course Outcome : 02				
1	(a)	A software system is being developed for a large retail chain to manage their product inventory. The system frequently needs to display products based on various criteria (e.g., price, quantity in stock, product ID). a. Compare and contrast Bubble Sort and Selection Sort in terms of their time complexity (best, average, and worst case) and their stability. Discuss a scenario where one might be marginally preferred over the other, even though both have O(N2) worst-case complexity. (8 marks) b. The retail chain decides to implement a more efficient sorting algorithm for their daily inventory reports, which can involve millions of products. Between Merge Sort and Quick Sort, which one would you recommend for this task and why? Justify your choice by discussing their typical performance characteristics, memory requirements, and stability properties. (8 marks) c. Describe a specific, realistic scenario in this inventory management system where the stability of a sorting algorithm would be a critical requirement. Provide an example of how instability could lead to undesirable outcomes for the retail chain. (4 marks)	20	CO2	L3
2		A university library is building a new digital catalog system. They have a massive database of books, journals, and articles, and users need to be able to search for specific titles or keywords. a. Explain the fundamental difference between Sequential Searching (Linear Search) and Binary Search. Discuss the preconditions that *must* be met for	20	CO2	L3

		Binary Search to be applicable and efficient. (6 marks) b. If the library's database is unindexed and unstructured (i.e., just a simple list of text entries), why would Sequential Searching be the only practical search method? Analyze its worst-case time complexity in this context. (6 marks) c. The library decides to sort its database by title to enable faster searching. Design a scenario where Binary Search would significantly outperform Sequential Searching in terms of query time. Quantify the difference in the number of comparisons for a database of 100,000 entries if the target item is found at roughly the midpoint. (8 marks)			
3	(a)	A cyber-security team is analyzing network traffic for malicious patterns. One task involves searching for specific "signature" strings (patterns) within large incoming data streams (texts). a. Describe the mechanism of the Brute Force String Matching algorithm. Illustrate its operation with a small example: search for the pattern "ABC" in the text "ABABDABACDABABCABA". Show the steps involved. (7 marks) b. Analyze the worst-case time complexity of Brute Force String Matching in terms of the text length (N) and pattern length (M). Provide an example of a specific text and pattern combination that would lead to this worst-case scenario. (7 marks) c. Discuss a practical limitation of using solely Brute Force String Matching for high-volume, real-time network traffic analysis. Briefly suggest why more advanced string matching algorithms (not necessarily from this syllabus, but known to exist) are preferred in such scenarios. (6 marks)	20	CO2	L3
4	(a)	The "General Method" for algorithm design often refers to the Divide and Conquer paradigm. Many efficient algorithms, including Merge Sort and Quick Sort, are based on this principle. a. Explain the three main steps of the Divide and Conquer general method. Use Merge Sort as an example to illustrate how each step is applied in its operation. (8 marks) b. Discuss the advantages of using a Divide and Conquer approach for problems that can be naturally broken down into smaller, independent subproblems. Consider aspects like parallelism and efficiency. (6 marks) c. Identify a type of problem (not necessarily from sorting or matrix multiplication) that would be *unsuitable* for a Divide and Conquer approach. Justify your answer by explaining why the problem's structure doesn't align with the paradigm's requirements. (6 marks)	20	CO2	L3
5	(a)	Development of a data processing tool involves datasets that are extremely large and fit entirely in memory, while others are so large they must be processed from disk.. a. Describe the recursive nature of Quick Sort. Explain how its partitioning step works and why it's crucial for the algorithm's performance. (7 marks) b. Compare the memory requirements of Merge Sort and Quick Sort. In which scenario (in-memory large dataset vs. disk-based extremely large dataset) would one of these algorithms be significantly more advantageous than the other, specifically concerning memory usage? Explain your reasoning. (8 marks) c. Quick Sort's performance is heavily influenced by the choice of pivot. Discuss a strategy for pivot selection that aims to mitigate the worst-case scenario for Quick Sort. (5 marks)	20	CO2	L3
6	(a)	In scientific computing and graphics, operations involving large matrices are common.	20	CO2	L3

		Standard matrix multiplication is computationally intensive, and specialized algorithms like Strassen's are often employed. a. State the standard (brute force) method's time complexity for multiplying two N×N matrices using Big O notation. (3 marks) b. Explain the core idea behind Strassen's Matrix Multiplication that allows it to achieve better asymptotic complexity than the standard method. You don't need to list all 7 recursive products, but explain the key insight (i.e., reducing the number of multiplications). (7 marks) c. While Strassen's algorithm has a better asymptotic complexity, it is not always used in practice for all matrix sizes. Discuss at least two practical reasons (e.g., overhead, constant factors, matrix size thresholds) why standard matrix multiplication might still be preferred over Strassen's for certain applications or matrix dimensions. (10 marks)			
	Course Outcome : 03				
1	(a)	Imagine you are managing resource allocation for a disaster relief organization. You have a single cargo plane with a limited weight capacity, and various aid packages available, each with a specific weight and a humanitarian value. You need to maximize the total humanitarian value of the aid packages transported. a. Distinguish between the Fractional Knapsack problem and the 0/1 Knapsack problem. Explain which of these problems is directly applicable to the scenario described above (aid packages with weight and humanitarian value, single plane with limited capacity), and why. (7 marks) b. Outline the greedy strategy used to solve the Fractional Knapsack problem. Apply this strategy to the following example: Cargo plane capacity = 100 kg. Aid packages: Package A (30 kg, value 60), Package B (50 kg, value 150), Package C (20 kg, value 40). Show your steps to determine the maximum humanitarian value. (8 marks) c. Consider a slightly modified scenario where aid packages cannot be split (e.g., medical kits or tents). Would the greedy approach still guarantee an optimal solution? Justify your answer by explaining the limitation of the greedy approach for this modified problem. (5 marks)	20	CO3	L3
2		A telecommunications company needs to connect several cities with fiber optic cables to form a robust communication network. Each possible connection between two cities has a different installation cost. The goal is to connect all cities such that the total installation cost is minimized, ensuring that every city can communicate with every other city, directly or indirectly. a. Identify the algorithmic problem that best models this scenario. Briefly explain why this problem is relevant to minimizing connection costs while ensuring full connectivity. (5 marks) b. Compare and contrast Prim's Algorithm and Kruskal's Algorithm for solving this problem. Discuss their core strategies (e.g., how they build the tree), and their typical data structures used. (8 marks) c. Suppose there are 100 cities. In a practical implementation, which algorithm (Prim's or Kruskal's) might be more advantageous if the network is very dense (many possible connections)? Conversely, which might be better if the network is very sparse (few possible connections)? Justify your reasoning in terms of their typical time complexities given different graph densities. (7 marks)	20	CO3	L3
3	(a)	A logistics company needs to find the most cost-effective route for deliveries from their	20	CO3	L3

		central warehouse (source) to various retail stores across a city. The city's road network has varying travel times/costs between intersections. a. Define the Single Source Shortest Path problem. Explain how the Principle of Optimality applies to this problem, providing a foundational basis for dynamic programming or greedy algorithms like Dijkstra's. (7 marks) b. Consider a simplified road network represented by a directed graph where edges have non-negative weights (travel times/costs). If you were to use an algorithm to find the shortest path from the warehouse to *all* other retail stores, what algorithm would you likely choose (e.g., Dijkstra's algorithm)? Describe the main steps of this algorithm. (8 marks) c. What if some road sections could have negative travel times/costs (e.g., due to subsidies or "time travel" shortcuts, which are not usually realistic but illustrate a concept)? Would your chosen algorithm from part (b) still guarantee a correct solution? Explain why or why not, and briefly mention an alternative algorithm if needed. (5 marks)			
4		Management of a complex software development project involves several distinct stages. At each stage, there are multiple possible tasks or modules that can be chosen, each leading to a different set of options for the next stage and having an associated cost. The objective is to find the sequence of tasks from the start to the end that minimizes the total cost. a. Describe the characteristics of a problem that makes it suitable for modeling as a Multi-stage Graph problem. How does it differ from a general shortest path problem in a non-layered graph? (7 marks) b. Explain how the Principle of Optimality is applied in solving a Multi-stage Graph problem. Illustrate with a small conceptual example how subproblems are solved and combined to find the overall optimal path. (8 marks) c. Construct a small example of a 3-stage graph problem (e.g., 2 nodes in stage 1, 3 in stage 2, 2 in stage 3) with costs on edges. Using the general method for multi-stage graphs, show how you would find the minimum cost path from the start to the end stage. (5 marks)	20	CO3	L3
5	(a)	A large internet service provider (ISP) needs to analyze the latency (time delay) between any two points in their extensive network. They require a comprehensive understanding of the shortest path between every possible pair of routers to optimize data routing and troubleshoot issues. a. Identify the algorithmic problem that directly addresses the need to find the shortest path between all pairs of nodes in a graph. Explain why algorithms like Dijkstra's (repeatedly applied) might be less efficient for this specific requirement compared to a dedicated all-pairs algorithm. (7 marks) b. Describe the working principle of the Floyd-Warshall algorithm for solving the All Pair Shortest Path problem. Explain how it uses dynamic programming and iterates through intermediate vertices to find all-pairs shortest paths. (8 marks) c. Discuss the implications of negative cycles in the graph for the Floyd-Warshall algorithm. How does the algorithm behave in the presence of negative cycles, and what does this signify for the shortest path problem in such cases? (5 marks)	20	CO3	L3
6		A tour company needs to plan a tour for a group of tourists visiting several specific cities. The tour must start and end in the same city, visiting each other designated city exactly once, and the goal is to minimize the total travel distance. a. Formally define the Traveling Salesperson Problem (TSP). Explain why it is	20	CO3	L3

		considered a classic example of an NP-hard problem, distinguishing it from problems like the Shortest Path problem (which is P-time solvable). (7 marks) b. Briefly describe a brute-force approach to solve the Traveling Salesperson Problem. Analyze its time complexity in terms of the number of cities (N). Explain why this approach quickly becomes infeasible for even a moderate number of cities. (8 marks) c. Since TSP is NP-hard, exact solutions are computationally expensive for large instances. Propose a high-level strategy (not necessarily a specific algorithm from the syllabus, but a general approach) that a tour company might use in practice to find a "good enough" (near-optimal) solution for a large number of cities, even if it cannot guarantee the absolute minimum distance. (5 marks)			
	Course Outcome : 04				
1	(a)	Development of a puzzle game involves placing objects on a board under specific constraints. One such puzzle is a variation of the classic N-Queens problem. a. Describe the general method of backtracking as an algorithmic technique. Explain its core principle of building solutions incrementally and abandoning branches that cannot lead to a valid solution. (7 marks) b. How can the N-Queen problem be solved using the backtracking approach? Outline the recursive steps, including how constraints (e.g., no two queens attacking each other) are checked to prune the search space. Illustrate with a partial trace for N=4, showing how an invalid placement leads to backtracking. (8 marks) c. Consider a modification to the N-Queen problem: instead of placing queens, you need to place N "rooks" on an N x N chessboard such that no two rooks attack each other. Would backtracking be an efficient method for this modified problem? Justify your answer by discussing the nature of the constraints and the search space compared to the N-Queen problem. (5 marks)	20	CO4	L3
2	(a)	A famous chess puzzle is the Knight's Tour Problem, where a knight must visit every square on a chessboard exactly once. a. Explain why the Knight's Tour Problem is a suitable candidate for a backtracking solution. What defines a "state" in this problem, and how does the algorithm explore possible moves? (7 marks) b. Outline the steps of a backtracking algorithm to find a Knight's Tour. Discuss the conditions for a move to be valid and the base case for a successful tour. (8 marks) c. Why is the Knight's Tour often more challenging to solve with simple backtracking compared to finding *any* path in a graph? Discuss the concept of Warnsdorff's Rule (without necessarily detailing its implementation) and how it attempts to improve the efficiency of finding a Knight's Tour compared to pure, unguided backtracking. (5 marks)	20	CO4	L3
3	(a)	A small manufacturing company needs to decide which products to produce to maximize profit, given limited raw materials. Each product requires a specific amount of material and yields a certain profit. Only whole units of products can be produced. a. Distinguish between the Fractional Knapsack problem and the 0/1 Knapsack problem. Which of these more accurately models the manufacturing scenario described, and why? (7 marks) b. Explain the general method of Branch and Bound. How does it improve upon a pure backtracking approach by using bounds (e.g., upper bounds on potential	20	CO4	L3

		solutions) to prune search tree branches more effectively? (8 marks) c. Describe how Branch and Bound could be applied to solve the 0/1 Knapsack problem. Specifically, how would you calculate the upper bound for a node in the search tree to determine if a branch can be pruned? (5 marks)			
4	(a)	A courier company needs to plan the most efficient route for a delivery driver who must visit a set of specific customer locations, starting and ending at the depot. The goal is to minimize the total travel distance. a. Identify this classic algorithmic problem. Explain why a brute-force approach for solving it quickly becomes computationally infeasible as the number of customer locations increases. (6 marks) b. Explain how the general method of Branch and Bound can be applied to solve the Traveling Salesperson Problem (TSP). Describe what constitutes a "node" in the search tree and how "branching" typically occurs. (8 marks) c. The effectiveness of Branch and Bound for TSP heavily relies on finding good lower bounds. Describe a common method to calculate a lower bound for a partial tour in TSP (e.g., using a minimum spanning tree on the remaining unvisited nodes or reduced cost matrix methods). Explain why a tighter lower bound leads to more effective pruning. (6 marks)	20	CO4	L3
5	(a)	When choosing an algorithm for an optimization problem where exact solutions are required but the search space is potentially very large, both Backtracking and Branch and Bound are often considered. a. What are the primary similarities and differences between Backtracking and Branch and Bound as general algorithmic methods? Focus on their core search mechanisms and how they explore the solution space. (8 marks) b. In what type of scenario or for what specific problem characteristics would Branch and Bound typically outperform Backtracking significantly, especially when seeking an *optimal* solution? Provide a concrete example problem where this difference would be evident. (7 marks) c. Discuss the role of heuristic functions in both backtracking (sometimes used for variable ordering) and branch and bound (for calculating bounds). How does the quality of these heuristics impact the practical efficiency of each algorithm? (5 marks)	20	CO4	L3
6	(a)	Many real-world optimization problems, including the 0/1 Knapsack and Traveling Salesperson problems, are classified as NP-hard. This implies that finding an optimal solution in polynomial time is generally not possible. a. For the 0/1 Knapsack problem, explain why a simple greedy approach (like the one used for Fractional Knapsack) does not guarantee an optimal solution. Provide a small counterexample. (7 marks) b. Given that problems like the Traveling Salesperson Problem are NP-hard, what are the practical implications for businesses or researchers who need to solve large instances of these problems? Why are algorithms like Branch and Bound, despite their exponential worst-case complexity, still valuable tools? (8 marks) c. When faced with an NP-hard problem that cannot be solved exactly in a reasonable time for large inputs, what alternative strategies or approaches (beyond exact algorithms like Branch and Bound) are commonly employed to find good, albeit not necessarily optimal, solutions? (5 marks)	20	CO4	L3
	Course Outcome : 05				
1		A software engineer is tasked with designing a new sorting algorithm that can	20	CO5	L3

		efficiently sort very large datasets. The manager claims it can sort in $O(NloglogN)$ time. a. Explain the concept of a lower bound for a problem. Why is it important to understand lower bounds when designing algorithms? (5 marks) b. Describe how Decision Trees are used to establish the lower bound for comparison-based sorting algorithms. Illustrate with a small example (e.g., sorting 3 elements) to show how each comparison branches the tree. (8 marks) c. Based on the decision tree model, refute your manager's claim by explaining why any comparison-based sorting algorithm cannot achieve a time complexity better than $O(NlogN)$. (7 marks)			
2		A company is developing a new scheduling software for complex manufacturing processes. Some scheduling problems seem easy to solve quickly, while others are incredibly difficult, even for small inputs. a. Define the complexity classes P, NP, and NP-Complete. Provide a real-world example of a problem that falls into each category. (8 marks) b. Explain the fundamental relationship between P, NP, and NP-Complete problems. Why is proving $P=NP$ a major open problem in computer science? (7 marks) c. Consider the problem: "Given a set of jobs with durations and deadlines, can all jobs be scheduled to complete by their deadlines?" Is this problem likely to be in P, NP-Hard, or NP-Complete? Justify your reasoning by relating it to known problems. (5 marks)	20	CO5	L3
3		An audio processing application frequently needs to multiply large polynomials, which represent frequency spectra. Standard polynomial multiplication is too slow. a. Describe the standard (naive) algorithm for multiplying two polynomials of degree $N-1$. Analyze its time complexity using Big O notation. (5 marks) b. Explain the core idea behind using the Fast Fourier Transform (FFT) for polynomial multiplication. How does it transform the problem to achieve a more efficient solution? You don't need to detail the FFT algorithm itself, but focus on its role in polynomial multiplication. (8 marks) c. State the improved time complexity achieved by using FFT for polynomial multiplication. Discuss a practical scenario in digital signal processing (other than audio processing) where the speedup provided by FFT for polynomial operations would be critical. (7 marks)	20	CO5	L3
4		A startup provides route optimization services, and one client needs to efficiently plan delivery routes that visit many locations. This problem is known to be NP-hard. a. What is an Approximation Algorithm? Explain its purpose and why it is used for certain types of problems, particularly those in the NP-hard class. (6 marks) b. Consider the Traveling Salesperson Problem (TSP) with triangle inequality (distances satisfy $d(u,v) \leq d(u,w)+d(w,v)$). Describe a common approximation algorithm for this variant of TSP that provides a 2-approximation (e.g., using an MST-based approach). (9 marks) c. In a real-world setting, what are the trade-offs involved in using an approximation algorithm compared to trying to find an exact solution for an NP-hard problem? Discuss both benefits and potential drawbacks. (5 marks)	20	CO5	L3
5		Implementing a custom dynamic array (similar to Python's list or Java's ArrayList) that needs to support efficient append operations. When the array runs out of space, it doubles its capacity.	20	CO5	L3

		a. Explain the concept of Amortized Analysis. How does it differ from worst-case analysis for a sequence of operations? Why is it useful for analyzing dynamic data structures? (7 marks) b. Using amortized analysis, show why a sequence of N append operations on a dynamic array that doubles its capacity when full takes O(N) total time. Detail the cost of reallocations and copies. (8 marks) c. If, instead of doubling, the array increased its capacity by a fixed constant amount (e.g., +10 elements) each time it ran out of space, what would be the amortized time complexity for N append operations? Justify your answer. (5 marks)			
6		A hospital is developing an emergency room patient management system. Patients arrive with different severity levels, and the system needs to quickly identify and retrieve the patient with the highest priority (most severe condition). a. Describe the properties of a Binary Heap (specifically, a max-heap) and how these properties enable efficient retrieval of the maximum element. (6 marks) b. Explain the primary operations on a max-heap: insert (adding a new patient) and extract_max (retrieving the highest priority patient). Analyze the time complexity of these operations in terms of the number of elements in the heap, N. (8 marks) c. In the context of the emergency room system, why would a heap be a more suitable data structure for managing patient priorities than a simple sorted array or an unsorted linked list? Discuss the efficiency of the key operations required for this application. (6 marks)	20	CO5	L3