Practice Assignment

Module 3	
Sr. No.	**Questions**
1	Explain the concepts of time complexity and space complexity in algorithm analysis. How do they impact the efficiency and resource utilisation of algorithms? Provide examples to illustrate the differences between time and space complexity.
2	Describe the significance of Big-O notation in algorithm analysis. How does it help in comparing the efficiency of algorithms and predicting their performance as input size grows? Provide real-world examples of different complexity classes using Big-O notation.
3	Explore the concept of recurrence relations in algorithm analysis. Provide a step-by-step explanation of solving a specific recurrence relation. Discuss the role of recursion in algorithm design and how it contributes to the overall efficiency of algorithms.
4	Delve into the use of Divide-and-conquer strategy in designing operations on data structures. Choose a specific data structure (e.g., arrays, trees) and explain how the divide-and-conquer approach can be applied to perform operations like search, insertion, or deletion efficiently.
5	Compare and contrast the time and space complexity of two different sorting algorithms. Choose one algorithm with better

	time complexity and another with better space complexity. Explain the trade-offs involved and situations where one algorithm might be preferred over the other.
6	Investigate the role of Big-O notation in analysing the performance of algorithms for different input sizes. Choose an algorithm and provide a step-by-step derivation of its time complexity using Big-O notation. Discuss how this notation helps in making algorithmic decisions.
7	Analyse a real-world problem and propose a divide-and-conquer algorithm to solve it efficiently. Describe the problem, the steps involved in dividing and conquering, and how the subproblems are combined to obtain the final solution. Discuss the time complexity of your algorithm.
8	Explore the application of recurrence relations in analysing the time complexity of recursive algorithms. Choose a specific recursive algorithm (e.g., Fibonacci sequence generation) and derive its time complexity using recurrence relations. Discuss any optimisation techniques that can be applied.
9	Discuss the importance of choosing the right data structure for specific algorithmic tasks. Choose a problem domain and explain how the choice of data structure can impact the time and space complexity of solving the problem. Provide examples to support your explanation.
10	Analyse a practical scenario where space complexity becomes a critical consideration in algorithm design. Explain how the algorithm's space complexity affects memory usage and overall performance. Discuss strategies to optimise space usage in such scenarios.