Introduction to Algorithms
1. Define an algorithm and explain its properties.
2. What are the factors that define the efficiency of an algorithm?
3. If an algorithm takes 5 ms for an input size of 100, estimate the time for an input size of 10,000 assuming $O(n^2)$ complexity.
Asymptotic Notations
1. Explain the difference between Big-O, Omega (Ω), and Theta (Θ) notations.
2. Prove that $f(n) = 4n^3 + 10n^2 + 5n + 1$ is $O(n^3)$.
3. Prove that if $f(n) = \Theta(g(n))$, then $g(n)$ is also $\Theta(f(n))$.
Recurrence Relations
1. Solve the recurrence relation $T(n) = T(n-1) + \log(n)$ using the substitution method.
2. Solve the recurrence relation $T(n) = 2T(n/4) + \sqrt{n}$ using the master theorem.
Searching Algorithms (Linear Search and Binary Search)
1. What is the time complexity of linear and binary search in the best, worst, and average cases?
2. Given the sorted array arr[] = {3, 4, 5, 12, 21, 23, 34, 43, 45, 67} and the target element x = 5, perform a linear search and binary search and determine the number of comparisons required to find the target.
3. Calculate the time complexity of linear and binary search for an array of size N in the worst-case scenario.
4. If the middle element in binary search is calculated as mid = L + (H - L) / 2, explain why this formula is used instead of mid = (L + H) / 2.
Sorting Algorithms
1. Compare the time complexities of Bubble Sort, Insertion Sort, and Selection Sort.
2. Given an unsorted array of size n, how many swaps are performed in Selection Sort in the worst case?
3. If an array of size 100 takes 1 second to sort using Bubble Sort, estimate the time required for an array of size 1000.
4. Given an array [5, 3, 8, 4, 2], show the step-by-step execution of Insertion Sort.
Divide and Conquer

1. Explain the three steps of the **Divide and Conquer** technique with an example.
2. How does **parallelism** improve Divide and Conquer algorithms?
3. Compare **Merge Sort** and **Quick Sort** in terms of their **time complexity** and **space complexity**.
4. Consider an array **A = [5, 2, 8, 1, 9, 3, 7]**.
    a. Show the step-by-step **Merge Sort** process.
    b. Show the step-by-step **Quick Sort** process using the **last element as pivot**
5. Given an array of $\mathbf{10^6}$ random numbers, which sorting algorithm (**Merge Sort** or **Quick Sort**) would you prefer? Justify your answer with complexity analysis.
6. If an array is already sorted, what will be the recursion depth of Quick Sort when using the last element as a pivot?
7. Suppose you modify **Merge Sort** so that it divides the array into **three parts** instead of two. Write the new recurrence relation and find the time complexity.

Maximum Subarray Sum

1. What is the Maximum Subarray Sum problem? Explain with an example
2. Compare the complexities of the brute force, Divide and Conquer, and Kadane's Algorithm approaches for solving the Maximum Subarray Sum problem.
3. Given the array **arr[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4}**, use Kadane's Algorithm to find the maximum subarray sum. Show each step of the algorithm.