Problem 1: QuickSort

Time Complexity: The average and best-case time complexity is **O(n log n)**. In the worst case, when the pivot is the smallest or largest element every time, the time complexity is **$O(n^2)$**.

Problem 2: Nested Loops Example

Time Complexity: The time complexity is **O(rows * cols)**, where `rows` is the number of rows and `cols` is the number of columns in the matrix

Problem 3: Simple Loop Example

Time Complexity: The time complexity is **O(n)**, where `n` is the length of the array.

Problem 4: Longest Increasing Subsequence (LIS)

Time Complexity: The time complexity is **$O(n^2)$** due to the nested loops iterating over the array.

Problem 5: Mysterious Function

Time Complexity: The time complexity is **$O(n^2)$** due to the nested loops iterating over the array.

Problem 6: Sum of Digits (Recursive)

Time Complexity: The time complexity is **O(log n)** because the number of recursive calls is proportional to the number of digits in `n`.

Problem 7: Fibonacci Series (Recursive)

Time Complexity: The time complexity is **$O(2^n)$** due to the exponential growth of recursive calls.

Problem 8: Subset Sum (Recursive)

Time Complexity: The time complexity is **$O(2^n)$** due to the recursive exploration of all subsets.

Problem 9: Word Break (Recursive) **Time Complexity**: The time complexity is **$O(2^n)$** in the worst case due to the recursive checking of every possible substring.

Problem 10: N-Queens (Recursive)

Time Complexity: The time complexity is **O(n!)** due to the factorial growth of possible queen placements on the chessboard.