Assignment-12

Task-1:

Prompt:

Write Python code for Bubble Sort with inline comments explaining passes, comparisons, swaps, and early termination. Include a brief explanation and time & space complexity.

Code and Output:

Explanation:

Task 2:

Prompt:

Write Python code for Bubble Sort and Insertion Sort. Explain why Insertion Sort is more efficient for partially sorted arrays and compare their performance on a nearly sorted input.

Code and Output:

Bubble Shot Implementation:

Insertion Sort Implementation:

Explanation:

Task 3:

Prompt:

Write Python code for Linear Search and Binary Search with docstrings and performance notes. Test both on sorted and unsorted data and explain when Binary Search is preferable.

Code and Output:

Linear Search Implementation:

Binary Search Implementation:

Explanation:

Task 4:

Prompt:

Implement recursive Quick Sort and Merge Sort with partially completed functions. Complete missing logic, add docstrings, and compare performance on random, sorted, and reverse-sorted lists.

Code & Output:

Explanation:

Task 5:

Prompt:

I've written a naive duplicate-finder function that checks each element against every other ($O(n^2)$ time). Please optimize it using a more efficient approach like sets or dictionaries to achieve O(n) time complexity. Also, compare the execution times of both versions on large input lists and explain how the complexity was improved.

Code & Output:

Explanation: