Data Structures and algorithms

Specify the sorting algorithm chosen for (c) and state its asymptotic speed in terms of the Best, Average and Worst Case.

I had utilized bubble sort as the chosen custom algorithm

Best Case: O(n)

This would happen when the array is already sorted, it would loop through the whole array with one full pass without swapping the values, so therefore the best case would only happen if no swipes occurred and the algorithm is stopped early.

Average Case: O(n²)

Occurs when the array is randomized on where each element may need to be compared with every other element.

Best Case: O(n²)

In the situation where the array is reversed so each element needs to be bubbled all the way to the correct positioning, which leads to the maximum number of comparisons and swaps.

Compare the asymptotic speeds above to those of the other 2 sorting algorithms implemented i.e. Merge Sort and Quick Sort and state which algorithm would be the most ideal one for sorting objects

Merge Sort:

Best Case: O(n log n)

Average Case: O(n log n)

Worst Case: O(n log n)

Constant fast and stable, utilizes more memory

Quick Sort:

Best Case: O(n log n)

Average Case: O(n log n)

Worst Case: O(n²)

Fast on average but can result into O(n²) in the worst case.

Most Ideal is Merge Sort:

It cannot degrade to O(n²) as all cases are O(n log n), keeps the original order of equal elements.

C: Written task for PRNG

The prng implementation is correct based on the conditions based in question B which states that the prng implementation must generate a specified range [1,1000],

So therefore, the numbers cannot be sorted both in ascending and descending order.

The prng implementation is not intractable, as the constant time performance per number generated is (o(1) so therefore it’s tractable.

A screenshot of a computer

AI-generated content may be incorrect.

Github Repo: https://github.com/Dfvgdev111/DSAHOMEAssignment