**Report on Sorting Algorithms**

# Data collection with vector sizes and running time on each sorting algorithm

Table

Description automatically generated with medium confidence

# Graph that compares the performance of different sorting algorithms (QuickSort, MergeSort, IterativeMergeSort, ShellSort, BubbleSort and InsertionSort) with increasing data size

Chart, line chart

Description automatically generated

From this graph, we see that BubbleSort is the slowest. InsertionSort is the second lowest.

# Graph that compares the performance of QuickSort, MergeSort, IterativeMergeSort, ShellSort with increasing data size

This graph provides a closer look at the performances of QuickSort, MergeSort, IterativeMergeSort, and ShellShort since they are much faster than BubbleSort and InsertionSort:

Chart

Description automatically generated

From this graph, we see that QuickSort is the fastest, then comes MergeSort, ShellSort, and IterativeMergeSort.

# Overall

The performances of each sorting algorithm based on the time running with increasing data size, in order from the fastest to the slowest is:

1. QuickSort
2. MergeSort
3. ShellSort
4. IterativeMergeSort
5. InsertionSort
6. BubbleSort

# Big-O Analysis

## QuickSort:

Worst case: O(n2)

Average case: O(n log n)

## MergeSort:

Worst case: O(n log n)

Average case: O(n log n)

## ShellSort:

Worst case: O(n2)

Average case: O(n3/2 )

## IterativeMergeSort:

Worst case: O(n log n)

Average case: O(n log n)

## InsertionSort:

Worst case: O(n2)

Average case: O(n2)

## BubbleSort:

Worst case: O(n2)

Average case: O(n2)

We know that O(n log n) is faster than O(n2): n log n < n2

O(n3/2) = O(n \* n1/2). n \* n1/2 < n2

Depending on the vector size, different sorting algorithms can have different time complexity:

All algorithms take approximately the same time for vector size less than 10.

O(n2) works fine with vector size 10,000, but slower with size 1,000,000.

O(n log n) are helpful with vector up to size 1,000,000.