Name: Timothy Lee

Student ID: 47724083

Date: 09/12/19

Lab 1 Report Document

**Section 1: Tables**

Table 1: All algorithm performance with dataset of size 10.

|  |  |  |  |
| --- | --- | --- | --- |
| size- 10 | Algorithms |  |  |
| Data Types | Bubble | Merge | Insertion |
| Random | 4.00000 ms | 1.00000 ms | 1.00000 ms |
| Reversed | 1.00000 ms | 1.00000 ms | 0.00000 ms |
| Unique | 0.00000 ms | 1.00000 ms | 0.00000 ms |
| Partial Sort | 0.00000 ms | 1.00000 ms | 0.00000 ms |

Table 2: All algorithm performance with dataset of size 1000.

|  |  |  |  |
| --- | --- | --- | --- |
| size-1000 | Algorithms |  |  |
| Data Types | Bubble | Merge | Insertion |
| Random | 6895.00 ms | 173.00 ms | 1757.00 ms |
| Reversed | 7249.00 ms | 145.00 ms | 1732.00 ms |
| Unique | 6483.00 ms | 143.00 ms | 1895.00 ms |
| Partial Sort | 2037.00 ms | 114.00 ms | 234.00 ms |

Table 3: All algorithm performance with dataset of size 10,000.

|  |  |  |  |
| --- | --- | --- | --- |
| size-10000 | Algorithms |  |  |
| Data Types | Bubble | Merge | Insertion |
| Random | 686666 ms | 2266 ms | 194318 ms |
| Reversed | 672466 ms | 2034 ms | 213344 ms |
| Unique | 675518 ms | 2006 ms | 187394 ms |
| Partial Sort | 196032 ms | 1386 ms | 17758 ms |

Table 4: All algorithm performance with dataset of size 100,000.

|  |  |  |  |
| --- | --- | --- | --- |
| size-100000 | Algorithms |  |  |
| Data Types | Bubble | Merge | Insertion |
| Random | 67820339 ms | 23650 ms | 19986539 ms |
| Reversed | 68323596 ms | 25315 ms | 18836859 ms |
| Unique | 68383766 ms | 24482 ms | 18623167 ms |
| Partial Sort | 20706015 ms | 18799 ms | 1654059 ms |

**Section 2: Graphs**

Table representation of graph “Random”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Size | Bubble Sort | Merge Sort | Insertion Sort |
| 10 | 4 | 1 | 1 |
| 1000 | 6895 | 173 | 1757 |
| 10000 | 686666 | 2266 | 194318 |
| 100000 | 67820339 | 23650 | 19986539 |

Table representation of graph “Reversed”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Size | Bubble Sort | Merge Sort | Insertion Sort |
| 10 | 1 | 1 | 0 |
| 1000 | 7249 | 145 | 1732 |
| 10000 | 672466 | 2034 | 213344 |
| 100000 | 68323596 | 25315 | 18836859 |

Table Representation of graph “Unique”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Size | Bubble Sort | Merge Sort | Insertion Sort |
| 10 | 0 | 1 | 0 |
| 1000 | 6483 | 143 | 1895 |
| 10000 | 675518 | 2006 | 187394 |
| 100000 | 68383766 | 24482 | 18623167 |

Table representation of graph “Partial Sort”

|  |  |  |  |
| --- | --- | --- | --- |
| Data Size | Bubble Sort | Merge Sort | Insertion Sort |
| 10 | 0 | 1 | 0 |
| 1000 | 2037 | 114 | 234 |
| 10000 | 196032 | 1386 | 17758 |
| 100000 | 20706015 | 18799 | 1654059 |

**Section 3: Analysis**

There were a few results that were not expected but upon further research gave insight into how these three sorting algorithms work under the hood. First, It was surprising to see that with a small dataset (10 elements), bubble and insertion sort outperformed merge sort on three out of four different types of data. Besides the completely random dataset, bubble sort and insertion sort were noticeably faster than merge sort. The reason for this odd finding was that with a small dataset that is either partially sorted, reversed, or partially unique, bubble sort and insertion sort perform relatively low numbers of operations while merge sort spends a lot of time and resources dividing and conquering a given dataset. However, when datasets were big enough that the operational cost of the divide and conquer approach became negligible, merge sort performed exponentially better than bubble or insertion sort in every dataset type. Second, insertion sort performed incredibly well –compared to its performance of other dataset types with the same number of elements—when sorting a partially sorted dataset regardless of its size. Even though it was still slower than merge sort, it is very clear that insertion sort performs well when the dataset is already partially sorted. Third, even though both bubble and insertion sort have the same time complexity (O(n^2)), insertion sort still performed much better than bubble sort. The graphs tell a clear story. With all types of datasets, insertion sort performed much better than bubble sort. Fourth, all three sorting algorithms seemed to struggle most with a random dataset. It was assumed that the reversed dataset would take longest to sort for all three algorithms. However, upon gathering actual data, it was made apparent that it is actually the random datasets that take the longest to sort. The reversed datasets were sorted faster than random sets, partially unique datasets were sorted faster than reversed sets, and finally, partially sorted sets performed faster than all other types of datasets.