**CSS 342: Program 4**

**Sorting Algorithm Runtimes**

**Disclaimer:** 32GB RAM Laptop with AMD Ryzen AI 9 CPU was used

As can be seen by the graph, Bubble sort and Insertion sort have the longest run time for higher inputs. They are appropriately given a run-time complexity of O(n^2) while all other sorting algorithms are O(n *log* n). This is also reflected by the graph. Merge Sort, Merge Iterative Sort, and Shell Sort have about the same run time efficiency for higher inputs, however, shell sort can be seen as slightly more efficient for input sized less 20000. Quick sort is the most efficient amongst this bunch of 6. It consistently produces the lowest run-time even for almost all input values except 10.

The tables below were used to graph with x=input size & y=average output, on MS Excel. Output times are the result of ./sorter [name]Sort [input size]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Size | Output Time 1 (microsecs) | Output Time 2 (microsecs) | Output Time 3 (microsecs) | Average Output (microsecs) |
| 10 | 3 | 1 | 2 | 2.00 |
| 100 | 52 | 36 | 48 | 45.33 |
| 1000 | 3926 | 3184 | 3163 | 3424.33 |
| 5000 | 121458 | 90079 | 90399 | 100312.00 |
| 10000 | 404592 | 370683 | 379387 | 384554.00 |
| 50000 | 10252512 | 10055291 | 10073020 | 10126807.67 |
| 100000 | 34588584 | 34408770 | 34297628 | 34431627.33 |

1. Bubble Sort - O(n^2)
2. Insertion Sort - O(n^2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Size | Output Time 1 (microsecs) | Output Time 2 (microsecs) | Output Time 3 (microsecs) | Average Output (microsecs) |
| 10 | 2 | 2 | 1 | 1.67 |
| 100 | 14 | 12 | 11 | 12.33 |
| 1000 | 1024 | 829 | 841 | 898.00 |
| 5000 | 26778 | 23436 | 20978 | 23730.67 |
| 10000 | 100637 | 83336 | 82314 | 90562.33 |
| 50000 | 1860955 | 1833603 | 1874882 | 1853146.67 |
| 100000 | 4749059 | 4784497 | 4869157 | 4804237.67 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Size | Output Time 1 (microsecs) | Output Time 2 (microsecs) | Output Time 3 (microsecs) | Average Output (microsecs) |
| 10 | 4 | 4 | 4 | 4.00 |
| 100 | 32 | 26 | 22 | 26.67 |
| 1000 | 254 | 255 | 203 | 237.33 |
| 5000 | 1579 | 1414 | 1156 | 1233.00 |
| 10000 | 2953 | 2984 | 2491 | 2809.33 |
| 50000 | 17446 | 17107 | 13883 | 16112.00 |
| 100000 | 36338 | 28783 | 28097 | 31072.67 |

1. Merge Sort - O(n *log* n)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Size | Output Time 1 (microsecs) | Output Time 2 (microsecs) | Output Time 3 (microsecs) | Average Output (microsecs) |
| 10 | 5 | 5 | 4 | 4.67 |
| 100 | 30 | 26 | 21 | 25.67 |
| 1000 | 254 | 252 | 502 | 336.00 |
| 5000 | 1388 | 1414 | 1274 | 1245.33 |
| 10000 | 2930 | 3698 | 2460 | 3029.33 |
| 50000 | 17652 | 16841 | 13626 | 16039.67 |
| 100000 | 37725 | 27758 | 31452 | 32378.33 |

1. Iterative Merge Sort - O(n *log* n)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Size | Output Time 1 (microsecs) | Output Time 2 (microsecs) | Output Time 3 (microsecs) | Average Output (microsecs) |
| 10 | 2 | 3 | 1 | 2.00 |
| 100 | 9 | 10 | 8 | 9.00 |
| 1000 | 119 | 116 | 90 | 108.33 |
| 5000 | 724 | 814 | 584 | 707.33 |
| 10000 | 1577 | 2119 | 1799 | 1831.67 |
| 50000 | 9654 | 9656 | 8192 | 9167.33 |
| 100000 | 19097 | 15796 | 16903 | 17265.33 |

1. Quick Sort - O(n *log* n)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input Size | Output Time 1 (microsecs) | Output Time 2 (microsecs) | Output Time 3 (microsecs) | Average Output (microsecs) |
| 10 | 2 | 1 | 1 | 1.33 |
| 100 | 10 | 11 | 9 | 10.00 |
| 1000 | 150 | 150 | 121 | 140.33 |
| 5000 | 1079 | 1293 | 1235 | 1202.33 |
| 10000 | 2323 | 2331 | 2032 | 2228.67 |
| 50000 | 15852 | 15942 | 12533 | 14775.67 |
| 100000 | 37545 | 26501 | 27269 | 30405.00 |

1. Shell Sort - O(n *log* n)