**Test Results**

Console output for test1.txt

This is using the test functions displayArrayTestIS(), displayArrayTestQS(), and displayArrayTestNS().

|  |  |  |
| --- | --- | --- |
| **Insertion Sort** | **Quick Sort** | **New Sort** |
| Values in file: test1.txt  029 081 089 050 038 060  091 088 064 035 052 090  065 061 009 065 053 017  074 004 014 040 047 071  064 078 095 006 037 064  026 007 060 046 043 099  002 076 033 013 080 067  089 065 026 012 062 064  019 012  After Insertion Sort  002 004 006 007 009 012  012 013 014 017 019 026  026 029 033 035 037 038  040 043 046 047 050 052  053 060 060 061 062 064  064 064 064 065 065 065  067 071 074 076 078 080  081 088 089 089 090 091  095 099  Comparisons: 732 | Values in file: test1.txt  029 081 089 050 038 060  091 088 064 035 052 090  065 061 009 065 053 017  074 004 014 040 047 071  064 078 095 006 037 064  026 007 060 046 043 099  002 076 033 013 080 067  089 065 026 012 062 064  019 012  After Quick Sort  002 004 006 007 009 012  012 013 014 017 019 026  026 029 033 035 037 038  040 043 046 047 050 052  053 060 060 061 062 064  064 064 064 065 065 065  067 071 074 076 078 080  081 088 089 089 090 091  095 099  Comparisons: 412 | Values in file: test1.txt  029 081 089 050 038 060  091 088 064 035 052 090  065 061 009 065 053 017  074 004 014 040 047 071  064 078 095 006 037 064  026 007 060 046 043 099  002 076 033 013 080 067  089 065 026 012 062 064  019 012  After New Sort  002 004 006 007 009 012  012 013 014 017 019 026  026 029 033 035 037 038  040 043 046 047 050 052  053 060 060 061 062 064  064 064 064 065 065 065  067 071 074 076 078 080  081 088 089 089 090 091  095 099  Comparisons: 2111 |

Console output for test1.txt – test6.txt

This is using the test functions testIS(), testQS(), and testNS().

|  |  |
| --- | --- |
| **test1.txt** | **test2.txt** |
| Insertion Sort Completed on file: test1.txt  Comparisons: 732  Quick Sort Completed on file: test1.txt  Comparisons: 412  New Sort Completed on file: test1.txt  Comparisons: 2111 | Insertion Sort Completed on file: test2.txt  Comparisons: 49  Quick Sort Completed on file: test2.txt  Comparisons: 558  New Sort Completed on file: test2.txt  Comparisons: 680 |
| **test3.txt** | **test4.txt** |
| Insertion Sort Completed on file: test3.txt  Comparisons: 799  Quick Sort Completed on file: test3.txt  Comparisons: 454  New Sort Completed on file: test3.txt  Comparisons: 680 | Insertion Sort Completed on file: test4.txt  Comparisons: 244286  Quick Sort Completed on file: test4.txt  Comparisons: 15767  New Sort Completed on file: test4.txt  Comparisons: 811677 |
| **test5.txt** | **test6.txt** |
| Insertion Sort Completed on file: test5.txt  Comparisons: 3616  Quick Sort Completed on file: test5.txt  Comparisons: 223565  New Sort Completed on file: test5.txt  Comparisons: 822334 | Insertion Sort Completed on file: test6.txt  Comparisons: 250455  Quick Sort Completed on file: test6.txt  Comparisons: 17366  New Sort Completed on file: test6.txt  Comparisons: 238691 |

Observations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File** | **Size** | **Sorted/Unsorted** | **Duplicates** | **Range of Values** | **Spread** |
| test1.txt | 50 | Unsorted | Few | 0 – 100 | Random but evenly spread |
| test2.txt | 50 | Almost Sorted | Multiple | 0 – 100 | Evenly spread |
| test3.txt | 50 | Unsorted | Multiple | 0 – 100 | Random but evenly spread |
| test4.txt | 1000 | Unsorted | Few | 0 – 2500 | Random but evenly spread |
| test5.txt | 1000 | Almost Sorted | Few | 0 – 2500 | Random large numbers unsorted |
| test6.txt | 1000 | Unsorted | Multiple | 0 - 250 | Random but evenly spread |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File** | **Size** | Comparisons | **Insertion Sort** | **Quick Sort** | **New Sort** |
| test1.txt | 50 | 732 | 412 | 2111 |
| test2.txt | 50 | 49 | 558 | 680 |
| test3.txt | 50 | 799 | 454 | 680 |
| test4.txt | 1000 | 244286 | 15767 | 811677 |
| test5.txt | 1000 | 3616 | 223565 | 822334 |
| test6.txt | 1000 | 250455 | 17366 | 238691 |

Quick Sort was the best sorting algorithm for these files as it required the least comparisons on average. Quick Sort could sort all those files with ~260000 comparison, whereas Insertion Sort took ~500000 and New Sort took ~1800000 comparisons

However, for Quick Sort, test2 *(with 558 comparisons)* *)* had much higher comparison results compared to test1 and test3, despite there being the same number of values – likewise with test5. Test2 and test5 contained arrays that were nearly sorted and therefore that shows that quick sort doesn’t have the best performance in this situation.

Insertion sort is the most efficient algorithm when the data is almost/already sorted and duplicate values have

little to no effect on the number of comparisons it takes as shown through test2 *(49)* and test5 *(3616)*.

In all cases the New Sort had the worst performance for sorting the 6 test files as it always searched to the end

of the array from where the last sorted value was. Therefore, in general, the larger the file the more comparisons

it took.

However, duplicate values in the data resulted in better performance from the New Sort as shown through the

reduced number of comparisons for test2, test3 (both has the same number of duplicates), and test6 compared

to the other test file results.