|  |  |
| --- | --- |
| Merge Sort Ascending | time (s) |
| 1000 | 0.01344710000557825 |
| 5000 | 0.07853209998575039 |
| 10000 | 0.16731624999374617 |
| 15000 | 0.25208375000511296 |

|  |  |
| --- | --- |
| Merge sort  decending | time (s) |
| 1000 | 0.03166195000812877 |
| 5000 | 0.0854727000114508 |
| 10000 | 0.17945489998965058 |
| 15000 | 0.2623077500174986 |

|  |  |
| --- | --- |
| Merge sort  Nearly Sorted | time (s) |
| 1000 | 0.03129939999780618 |
| 5000 | 0.09324400000332389 |
| 10000 | 0.20487329999741632 |
| 15000 | 0.2957026499789208 |

|  |  |
| --- | --- |
| quickSort left most Ascending | time (s) |
| 100 | 0.0010923000081675127 |
| 300 | 0.009515300014754757 |
| 400 | 0.015592649986501783 |
| 900 | 0.07533449999755248 |

|  |  |
| --- | --- |
| quickSort left most Decending | time (s) |
| 100 | 0.004818650020752102 |
| 300 | 0.02839184999174904 |
| 400 | 0.045622850011568516 |
| 900 | 0.19995530000596773 |

|  |  |
| --- | --- |
| quickSort left most Nearly Sorted | time (s) |
| 100 | 0.003015649985172786 |
| 300 | 0.009532049982226454 |
| 400 | 0.012707899993984029 |
| 900 | 0.0261939999909373 |

|  |  |
| --- | --- |
| Quicksort 3 Med Ascending | time (s) |
| 1000 | 0.005192799988435581 |
| 5000 | 0.031982700020307675 |
| 10000 | 0.06963940001151059 |
| 15000 | 0.10589160000381526 |

|  |  |
| --- | --- |
| Quicksort 3 Med Decending | time (s) |
| 1000 | 0.025441950012464076 |
| 5000 | 0.07036080000398215 |
| 10000 | 0.1414688499935437 |
| 15000 | 0.21933595000882633 |

|  |  |
| --- | --- |
| QUicksort 3 med nearly sorted | time (s) |
| 1000 | 0.021671800001058728 |
| 5000 | 0.04395604999444913 |
| 10000 | 0.08793074999994133 |
| 15000 | 0.1367661499971291 |

|  |  |
| --- | --- |
| Quicksort Random Ascending | time (s) |
| 1000 | 0.007294699986232445 |
| 5000 | 0.04532355001720134 |
| 10000 | 0.10491739999270067 |
| 15000 | 0.14912550001463387 |

|  |  |
| --- | --- |
| Quicksort Random Decending | time (s) |
| 1000 | 0.024501599997165613 |
| 5000 | 0.058452800003578886 |
| 10000 | 0.10860354999022093 |
| 15000 | 0.168555699987337 |

|  |  |
| --- | --- |
| QUicksort Random nearly sorted | time (s) |
| 1000 | 0.02722005000396166 |
| 5000 | 0.048029049998149276 |
| 10000 | 0.10112995000963565 |
| 15000 | 0.152238299997407 |

From the gathered results, we can conclude that with all O(N log N) sorting algorithms, the larger the N value, the longer it takes for the values to be sorted. From the results of the merge sort, we can see splitting in half the N value decreases the number of sub arrays exponentially and hence the time it takes to sort decreases because during merges, there are lesser copying of all N elements at each level in the recursion tree. Merge sorts are very stable and always result in a consistent O(N logN) complexity. Quick sorts depends heavily on how good the pivots are, the speed of sorting will vary depending on how many pivots split sub arrays exactly in half. The lesser splits every pivot creates, the slower the sorting speed. More splits will result in an O(N log N) complexity while if there are no splits, it results in a O(N^2) complexity.