**Merge Sort**

Random Sorted

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 59.67 |
| 1000 | 990.00 |
| 10000 | 34398.33 |
| 100000 | 1342945.33 |

Ascending Order

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 60.66 |
| 1000 | 1061.33 |
| 10000 | 35786.33 |
| 100000 | 1310168.0 |

Shows the consistency of Merge Sort, time taken is around the same for random and sorted data types. However, the time taken is fairly slow compared to Quicksort but still faster than O(n^2) sorting algorithms. This is mainly due to being not an inplace sorting algorithm, hence copying data back from the temp array requires extra computational time .

**Quick Sort(Left Most Pivot)**

Random Sorted

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 29.67 |
| 1000 | 63.0 |
| 10000 | 547.0 |
| 100000 | 6493.33 |

Ascending Order

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 86.33 |
| 1000 | 953.0 |
| 10000 | StackOverflow |
| 100000 | StackOverflow |

This shows the inconsistency of Quicksort if a bad pivot point is picked. It works well on random sorted arrays, however when given a near sorted or sorted array the time complexity goes up to O(n^2) due to the pivot not partitioning the data at all. In larger data sets, the stack can overflow.

**QuickSort(Random)**

Random Sorted

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 32.67 |
| 1000 | 70.33 |
| 10000 | 635.33 |
| 100000 | 7398.00 |

Ascending Order

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 85 |
| 1000 | 173.67 |
| 10000 | 677.00 |
| 100000 | 4221.00 |

Quicksort with random pivot selection also has inconsistencies in time taken but not as significant as the left most index. It is also fairly quick due to requiring less computational steps to calculate pivot index.

**QuickSort(Median of Three)**

Random Sorted

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 40.33 |
| 1000 | 83.00 |
| 10000 | 1027.33 |
| 100000 | 6757.33 |

Ascending Order

|  |  |
| --- | --- |
| Input Size | Time Taken(Micro Seconds) |
| 100 | 42.33 |
| 1000 | 121.33 |
| 10000 | 506.67 |
| 100000 | 2483.33 |

Quicksort with a better pivot selection shows a more consistent O(n log n) sorting times for different sets of data. However having a more consistent sorting time makes the sorting algorithm slightly slower compared to the random and left most pivot selection due to needing extra computational steps when determining the median.

Overall, both Quicksort and Mergesort scale well with large data sets due to its O( n log(n) ) time complexity, with Quicksort being much faster than Mergesort due to being inplace and requiring less computational steps. Both Quicksort and Mergesort are also not stable.

The selection of the pivot in Quicksort is extremely important in maintaining O(n log(n) ) time complexity for different data sets, however trades its speed for consistency.