Array Size = 100

Selection Sort Average Time: 11007.55

Insertion Sort Average Time: 6785.01

Merge Sort Average Time: 4033.91

Quick Sort Average Time: 1966.18

Array Type: Sorted

Selection Sort Average Time: 1704.89

Insertion Sort Average Time: 815.94

Merge Sort Average Time: 2273.50

Quick Sort Average Time: 2568.82

Array Type: Reverse-sorted

Selection Sort Average Time: 2501.90

Insertion Sort Average Time: 941.98

Merge Sort Average Time: 2672.54

Quick Sort Average Time: 2326.5

Array Size: 1000

Selection Sort Average Time: 163939.82

Insertion Sort Average Time: 48604.44

Merge Sort Average Time: 28566.69

Quick Sort Average Time: 9988.44

Array Type: Sorted

Selection Sort Average Time: 148211.78

Insertion Sort Average Time: 692.68

Merge Sort Average Time: 22578.29

Quick Sort Average Time: 200982.88

Array Type: Reverse-sorted

Selection Sort Average Time: 205168.39

Insertion Sort Average Time: 88190.80

Merge Sort Average Time: 25468.66

Quick Sort Average Time: 174533.5

Array Size: 10000

Selection Sort Average Time: 15164910.88

Insertion Sort Average Time: 4423911.43

Merge Sort Average Time: 563595.83

Quick Sort Average Time: 376665.27

**Discussion**

Selection sort has the longest execution times for all array sizes and types, matching its theoretical time complexity of O(n^2). This makes it inefficient for large datasets. While insertion sort performs better than Selection Sort, it still shows slower performance for larger arrays. With a time complexity of O(n^2), it tends to be slower, especially for bigger datasets. Merge sort is much more consistent performance -wiseacross different array sizes and types. It has a time complexity of O(n log n) which makes it much more efficient for larger data sets. Lastly, quick sort has the best performance among the sorting algorithms tested because it has an average time complexity of O(n log n).



