Activity 4

Sort Name	Data size	Sorting order	Runtime results
Merge sort	10	Ascending	4.1
Merge sort	20	Descending	4.3
Merge sort	100	In random order	5.6
Merge sort	50	Nearly sorted	5.3
Quick sort (Left most Pivot selection)	10	Ascending	5.33
Quick sort (Left most Pivot selection)	20	Descending	5.67
Quick sort (Left most Pivot selection)	100	In random order	5.8
Quick sort (Left most Pivot selection)	50	Nearly sorted	5.9
Quick sort (median of three pivot selection)	10	Ascending	2.3
Quick sort (median of three pivot selection)	20	Descending	4.6
Quick sort (median of three pivot selection)	100	In random order	3.33
Quick sort (median of three pivot selection)	50	Nearly sorted	1.33
Quick sort (random pivot selection)	10	Ascending	2.33
Quick sort (random pivot selection)	20	Descending	5.0
Quick sort (random pivot selection)	100	In random order	5.0
Quick sort (random pivot selection)	50	Nearly sorted	4.0

Activity 4

When comparing and contrasting the runtime results of the merge sort, quick sort (left most pivot) , quick sort (median of three) and quick sort in every case (arranged in either ascending, descending, in random or nearly sorted merge sorts worst case to best case has (kind of) the same time complexity because no matter what the number of comparisons is we have to copy all the elements at each level in the recursion tree. Therefore, all worst case , best case and average cases have the same time complexity of O(N log N) .

<u>Quick sort (left most pivot)</u>

Compared to quick sort by selecting the median as the pivot or a random as the pivot element the Quick sort with the left most pivot is slower because there is no smaller element to the left of the array other than the pivot element which means there is no left sub array at all.

The left most pivot strategy will almost always lead to the worst-case scenario with a time complexity of O(N^2). The worst-case scenario would be not splitting the array at all. This is the reason that the nearly sorted array gets too longer than the ascending , descending and random .

<u>Quick sort (median of three pivot)</u>

Guarantees a split with at least one element. The nearly sorted takes less time than the ascending , descending and random because it is almost close to the best-case scenario. But still the time complexity of the best-case scenario would be O(N log N).

<u>Quick sort (random pivot element)</u>

This strategy is very unlikely to become a poor pivot element. So, the worst-case time complexity of O(N^2) is highly unlikely to hit. And compared to the above mentioned other two strategies selecting a random element as the pivot is the fastest for the quick sort methos.