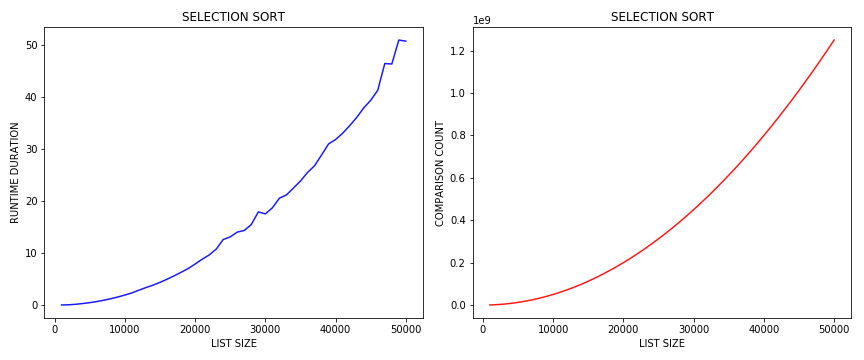
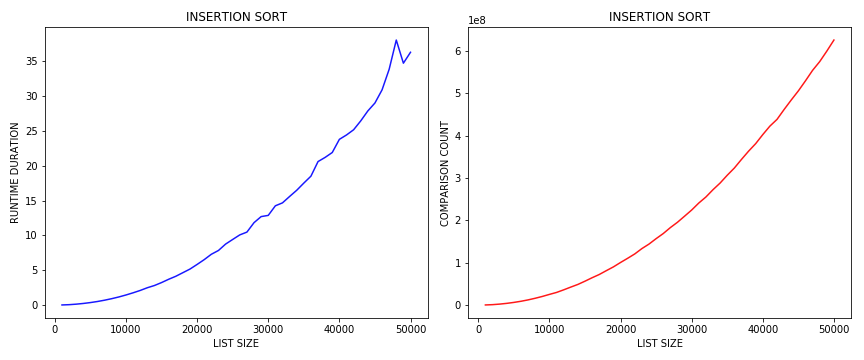
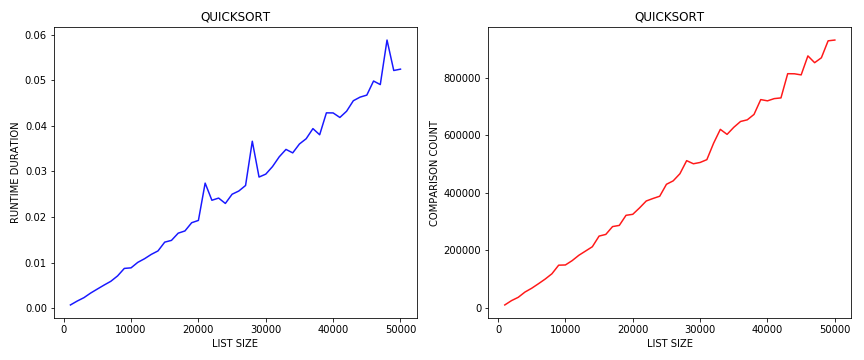
Laurence Aung

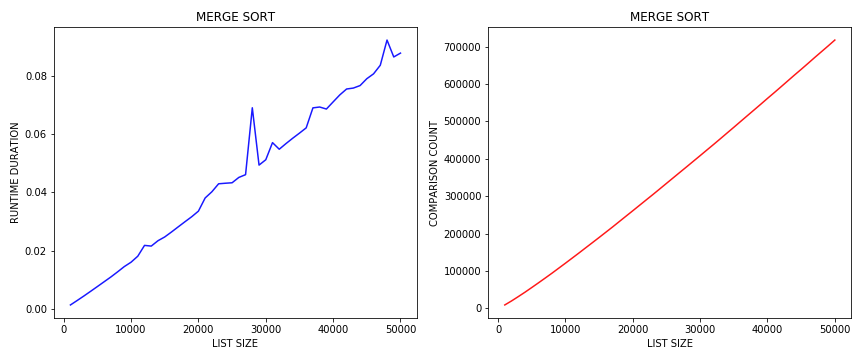
Analyzing Sorting Algorithms

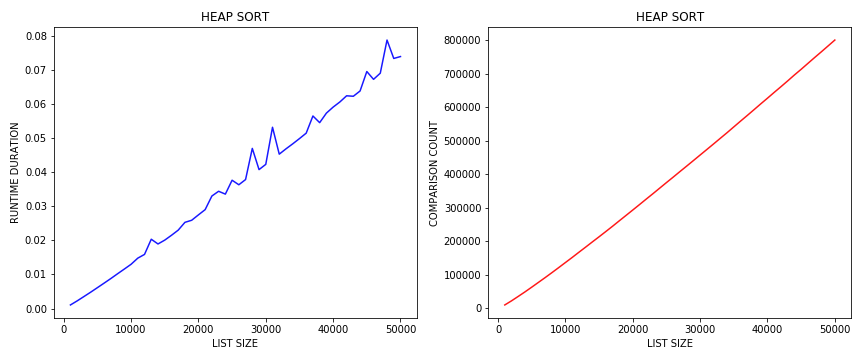
### GRAPHS OF DIFFERENT ALGORITHMS

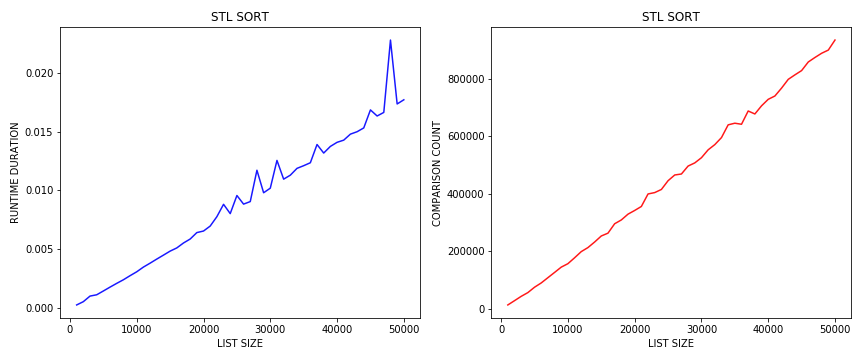


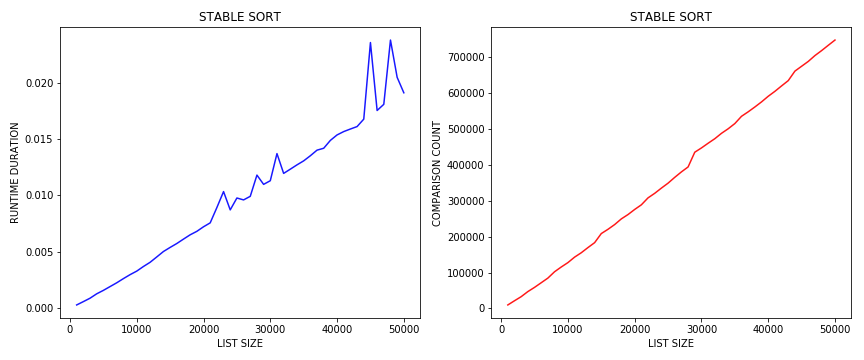












### HEAP SORT

In this type of sorting we first create a min heap (for sorting in ascending order) or max heap (for sorting in descending order). Time taken to create a heap from an array containing n elements is O(n). Then one by one we pop elements from the heap and store the elements in an array sequentially. When we pop elements from a min heap we get the next smaller element. So, we can see that we get the elements in a sorted order. Time taken to pop an element from the heap is O(logn). So, time taken to pop n elements is O(nlogn). So, the overall time complexity of our Heap Sort is O(nlogn).

### ANALYSIS

From the above graphs we can see that Selection sort and Insertion sort takes O() time complexity. Quick sort, Merge sort, Heap sort, STL sort and Stable sort takes O(nlogn) time complexity. The graphs of Selection sort and Insertion sort all have the form y = , while the graphs of the other sorting algorithms have a straight line form.

STL sort and STL stable\_sort all have a time complexity of O(nlogn).

STL sort is actually a hybrid sorting algorithm which uses three sorting algorithms (Quicksort, Heapsort, Insertion sort) to minimize the running time.

STL stable\_sort is actually implemented as merge sort. stable\_sort preserves the physical order of semantically equivalent values,i.e., it preserves the order in which the similar elements are present in the array.

The fastest algorithm here is the STL sort. As it is implemented in a very efficient way so it takes less time than the others.