DOCUMENTATION AND COMPLEXITY ANALYSIS

CODE DESCRIPTION :

The code that I have written is in C++ and I've used File I/O to read the files given and written the sorted arrays into another file which is present in the respective sort's folder. I have displayed on the console the time taken for computation for the same and presented below is the complexity analysis for the three kinds of sorting algorithms implemented.

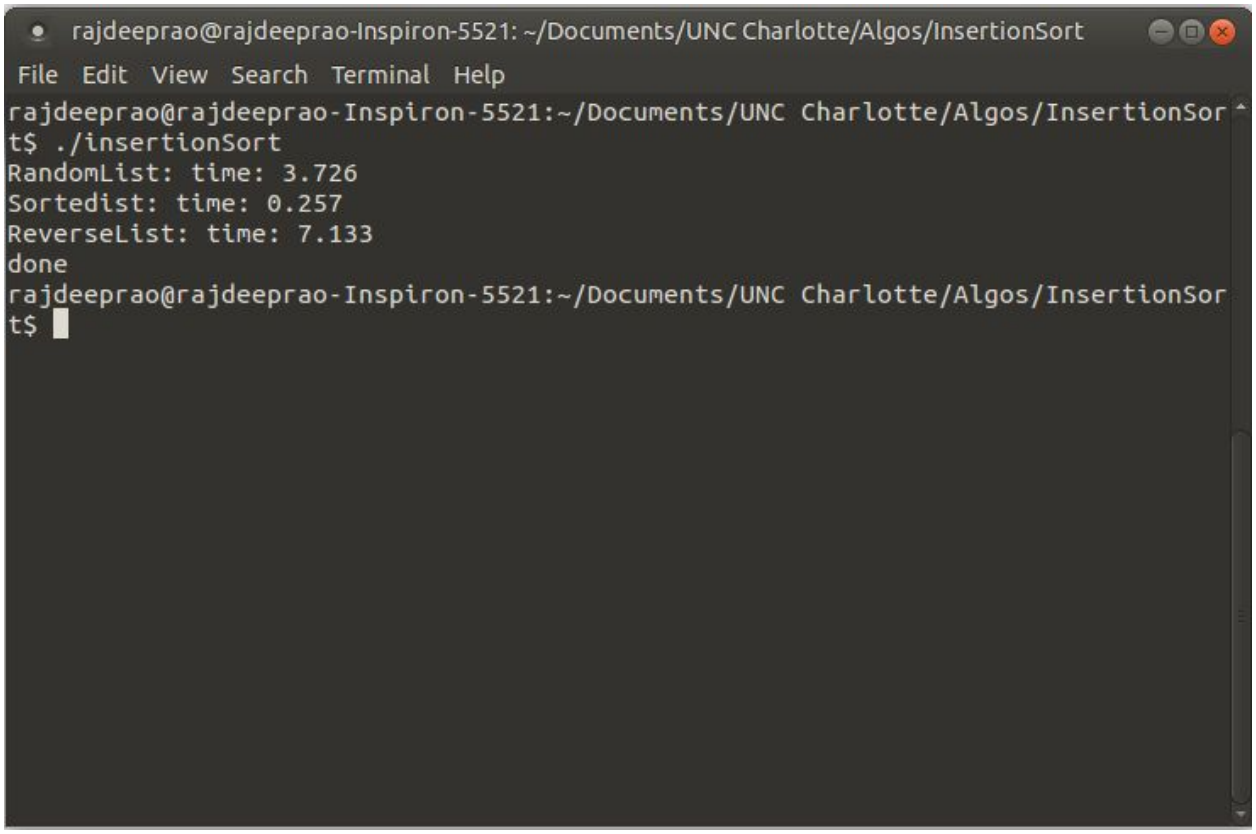
INSERTION SORT :

- Stable
- $O(n^2)$ comparisons and swaps

Best case scenario is when the array is sorted/almost sorted in the first place which is backed by the time taken by the code I have written. Sorted array took 0.257 : O(n)

Average case scenario when the array elements distribution is random which is backed by the time taken by the code I have written. Random array took 3.726 : $O(n^2)$

Worst case scenario when the array is reverse sorted which is backed by the time taken by the code I have written. Sorted array took 7.133 : $O(n^2)$

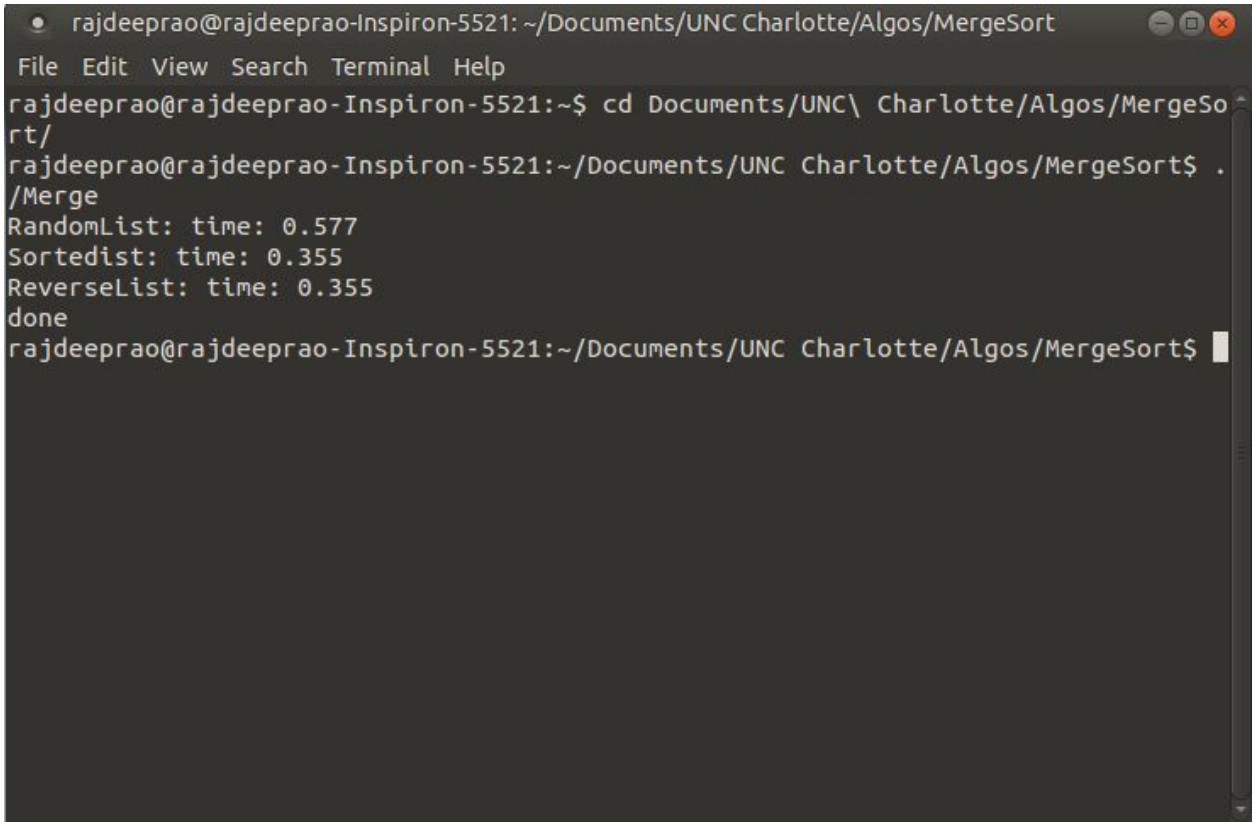


MERGE SORT :

- Stable
- Θ(n) extra space for arrays
- Θ(n·lg(n)) time

It makes between 0.5lg(n) and lg(n) comparisons per element, and between lg(n) and 1.5lg(n) swaps per element. The minima are achieved for already sorted data; the maxima are achieved, on average, for random data. If using Θ(n) extra space is of no concern, then merge sort is an excellent choice: It is simple to implement, and it is the only stable O(n·lg(n)) sorting algorithm

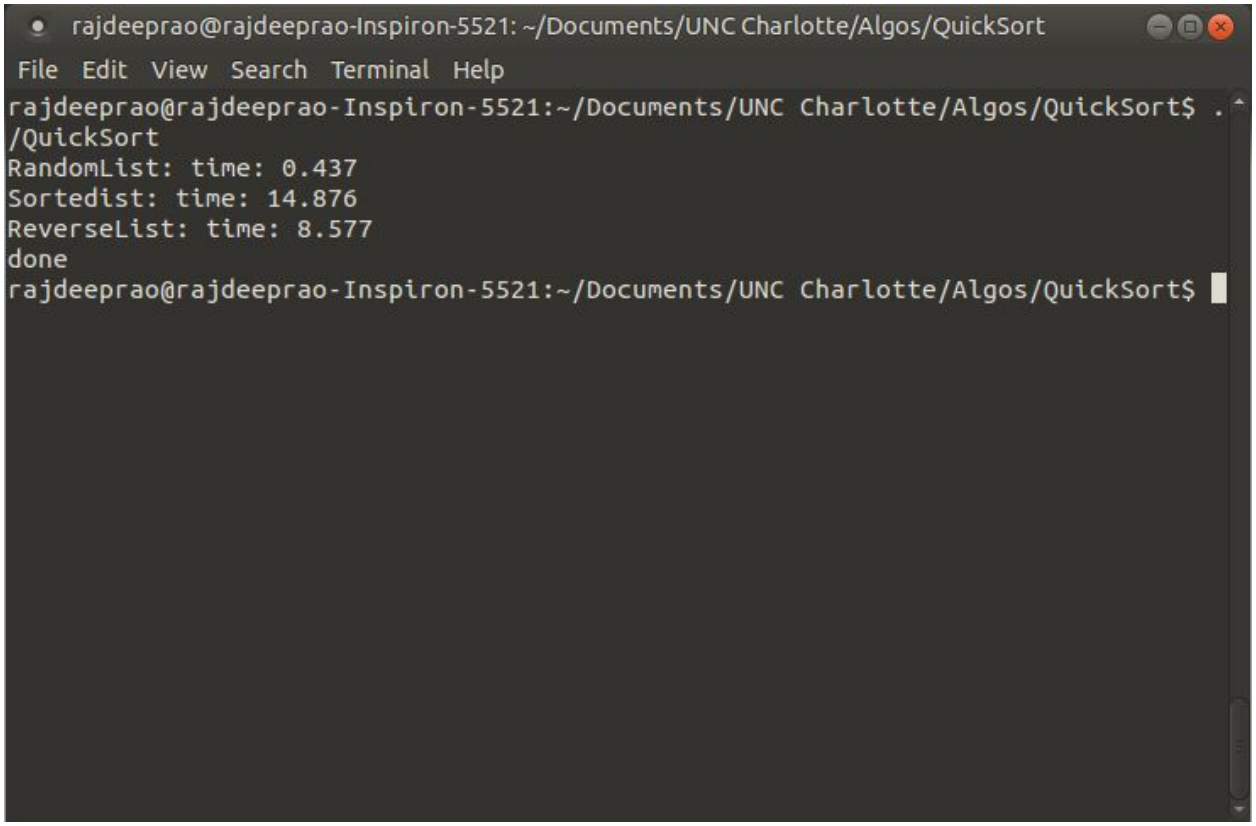
Sorted and Reverse both took about the same time of 0.355 while random took a little bit more of 0.577 but it has performed much better than Insertion Sort. Only trade of is that more space is required for this.



QUICK SORT :

- Not stable
- O(lg(n)) extra space (see discussion)
- $O(n^2)$ time, but typically $O(n \cdot lg(n))$ time

When carefully implemented, quick sort is robust and has low overhead. When a stable sort is not needed, quick sort is an excellent general-purpose sort.
Using a normal quick sort performs very badly when presented an almost sorted array as is evident from the attached code. It took a total of 14.876 : $O(n^2)$.


rajdeeprao@rajdeeprao-Inspiron-5521: ~/Documents/UNC Charlotte/Algos/QuickSort
File Edit View Search Terminal Help
rajdeeprao@rajdeeprao-Inspiron-5521:~/Documents/UNC Charlotte/Algos/QuickSort$ ./QuickSort
RandomList: time: 0.437
Sortedist: time: 14.876
ReverseList: time: 8.577
done
rajdeeprao@rajdeeprao-Inspiron-5521:~/Documents/UNC Charlotte/Algos/QuickSort$