Problem Statement:

Given an arbitrary set of unordered numbers in an array, sort this array using all the sorting algorithms discussed in the class. Primarily, focus on analyzing computational time (i.e. CPU time) and the number of iterations, swap operations, and other similar operations performed by the algorithms.

For analysis construct a set of arrays such as a completely random array, almost sorted array, reversely sorted array, etc. Furthermore, you may also vary the size of the array to assess its effects on different algorithms.

Along with the source code, please also include an analysis MS Word document to include snapshot of different results.

Source Code:

There are 8 sorting algorithms on which the analysis is taken place since these are all the algorithms that were taught in class. These are:

1. Bubble Sort
2. Optimized Bubble Sort
3. Selection Sort
4. Insertion Sort
5. Merge Sort
6. Quick Sort
7. Count Sort
8. Radix Sort

First the general code of these algorithms is written in a class called simpAlgor. This doesn't have any additional code written to analyze the code in different ways since it'll also contribute to the runtime. This class is used to measure the computational time only. Another class is made called iterAlgor, in which additional code is written in each function of sorting algorithm to get the number of iterations and swaps. Similarly, code for different parameters can be written in this same class for comparison. The time measured is process time and not the total executed time, and this is platform dependent. This code cannot work on except windows. This code, after testing has been realized that is not good and inconsistent so this parameter should be taken with a grain of salt. These algorithms are then now analyzed under different cases, such as different sizes and ranges and different levels of sorting. The random function with a for loop is used to fill the array in a non-sorted manner. N is size of the array, and the range is the range of values that the array can have.

Case 1: Low N, Low Range

```
                BUBBLE SORT
Time measured: 0.000000 ms.
The number of iterations are 19900.
The number of swaps are 10052.
                OPTIMIZED BUBBLE SORT
Time measured: 0.000000 ms.
The number of iterations are 19855.
The number of swaps are 10052.
                SELECTION SORT
Time measured: 0.000000 ms.
The number of iterations are 19900.
The number of swaps are 195.
                INSERTION SORT
Time measured: 0.000000 ms.
The number of iterations are 199.
The number of swaps are 10052.
                MERGE SORT
Time measured: 0.000000 ms.
The number of iterations are 2963.
The number of swaps are 607.
                Quick SORT
Time measured: 0.000000 ms.
The number of iterations are 1771.
The number of swaps are 881.
                COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 1199.
The number of swaps are 200.
                RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 2030.
The number of swaps are 600.
```

The number of N is 200 and the range is also kept 200.

Case 2: Low N, Medium Range

```
                BUBBLE SORT
Time measured: 0.000000 ms.
The number of iterations are 19900.
The number of swaps are 10067.
                OPTIMIZED BUBBLE SORT
Time measured: 0.000000 ms.
The number of iterations are 19729.
The number of swaps are 10067.
                SELECTION SORT
Time measured: 0.000000 ms.
The number of iterations are 19900.
The number of swaps are 191.
                INSERTION SORT
Time measured: 0.000000 ms.
The number of iterations are 199.
The number of swaps are 10067.
                MERGE SORT
Time measured: 0.000000 ms.
The number of iterations are 2955.
The number of swaps are 599.
                Quick SORT
Time measured: 0.000000 ms.
The number of iterations are 1455.
The number of swaps are 834.
                COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 16773.
The number of swaps are 200.
                RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 2640.
The number of swaps are 800.
```

The number of N is 200 and the range is also kept 8,000.

Case 3: Low N, High Range

```
                BUBBLE SORT
Time measured: 0.000000 ms.
The number of iterations are 19900.
The number of swaps are 10111.
                OPTIMIZED BUBBLE SORT
Time measured: 0.000000 ms.
The number of iterations are 19795.
The number of swaps are 10111.
                SELECTION SORT
Time measured: 0.000000 ms.
The number of iterations are 19900.
The number of swaps are 197.
                INSERTION SORT
Time measured: 0.000000 ms.
The number of iterations are 199.
The number of swaps are 10111.
                MERGE SORT
Time measured: 0.000000 ms.
The number of iterations are 2985.
The number of swaps are 629.
                Quick SORT
Time measured: 0.000000 ms.
The number of iterations are 1372.
The number of swaps are 770.
                COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 66315.
The number of swaps are 200.
                RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 3250.
The number of swaps are 1000.
```

The number of N is 200 and the range is also kept 40,000.

Case 4: Medium N, Low Range

```
                BUBBLE SORT
Time measured: 812.500000 ms.
The number of iterations are 31996000.
The number of swaps are 15909684.
             OPTIMIZED BUBBLE SORT
Time measured: 906.250000 ms.
The number of iterations are 31988860.
The number of swaps are 15909684.
                SELECTION SORT
Time measured: 93.750000 ms.
The number of iterations are 31996000.
The number of swaps are 7943.
                INSERTION SORT
Time measured: 46.875000 ms.
The number of iterations are 7999.
The number of swaps are 15909684.
                  MERGE SORT
Time measured: 15.625000 ms.
The number of iterations are 202418.
The number of swaps are 46258.
                  Quick SORT
Time measured: 0.000000 ms.
The number of iterations are 240878.
The number of swaps are 36688.
                  COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 32399.
The number of swaps are 8000.
                  RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 80030.
The number of swaps are 24000.
```

The number of N is 8,000 and the range is also kept 200.

Case 5: Medium N, Medium Range

```
                BUBBLE SORT
Time measured: 890.625000 ms.
The number of iterations are 31996000.
The number of swaps are 15879191.
             OPTIMIZED BUBBLE SORT
Time measured: 875.000000 ms.
The number of iterations are 31985989.
The number of swaps are 15879191.
                SELECTION SORT
Time measured: 78.125000 ms.
The number of iterations are 31996000.
The number of swaps are 7994.
                INSERTION SORT
Time measured: 62.500000 ms.
The number of iterations are 7999.
The number of swaps are 15879191.
                  MERGE SORT
Time measured: 15.625000 ms.
The number of iterations are 202593.
The number of swaps are 46433.
                  Quick SORT
Time measured: 15.625000 ms.
The number of iterations are 117048.
The number of swaps are 59946.
                  COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 47999.
The number of swaps are 8000.
                  RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 104040.
The number of swaps are 32000.
```

The number of N is 8,000 and the range is also kept 8,000.

Case 6: Medium N, High Range

```
                BUBBLE SORT
Time measured: 781.250000 ms.
The number of iterations are 31996000.
The number of swaps are 15879825.
             OPTIMIZED BUBBLE SORT
Time measured: 828.125000 ms.
The number of iterations are 31992840.
The number of swaps are 15879825.
                SELECTION SORT
Time measured: 78.125000 ms.
The number of iterations are 31996000.
The number of swaps are 7991.
                INSERTION SORT
Time measured: 46.875000 ms.
The number of iterations are 7999.
The number of swaps are 15879825.
                  MERGE SORT
Time measured: 0.000000 ms.
The number of iterations are 202725.
The number of swaps are 46565.
                  Quick SORT
Time measured: 0.000000 ms.
The number of iterations are 117963.
The number of swaps are 63136.
                  COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 97531.
The number of swaps are 8000.
                  RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 128050.
The number of swaps are 40000.
```

The number of N is 8,000 and the range is also kept 40,000.

Case 7: High N, Low Range

```
                BUBBLE SORT
Time measured: 19453.125000 ms.
The number of iterations are 799980000.
The number of swaps are 397054650.
                OPTIMIZED BUBBLE SORT
Time measured: 17296.875000 ms.
The number of iterations are 799818404.
The number of swaps are 397054650.
                SELECTION SORT
Time measured: 1734.375000 ms.
The number of iterations are 799980000.
The number of swaps are 39779.
                INSERTION SORT
Time measured: 1015.625000 ms.
The number of iterations are 39999.
The number of swaps are 397054650.
                MERGE SORT
Time measured: 31.250000 ms.
The number of iterations are 1205974.
The number of swaps are 275478.
                Quick SORT
Time measured: 15.625000 ms.
The number of iterations are 4482072.
The number of swaps are 201556.
                COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 160399.
The number of swaps are 40000.
                RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 400030.
The number of swaps are 120000.
```

The number of N is 40,000 and the range is also kept 200.

Case 8: High N, Medium Range

```
                BUBBLE SORT
Time measured: 17765.625000 ms.
The number of iterations are 799980000.
The number of swaps are 399959099.
                OPTIMIZED BUBBLE SORT
Time measured: 17203.125000 ms.
The number of iterations are 799977299.
The number of swaps are 399959099.
                SELECTION SORT
Time measured: 1906.250000 ms.
The number of iterations are 799980000.
The number of swaps are 39984.
                INSERTION SORT
Time measured: 1015.625000 ms.
The number of iterations are 39999.
The number of swaps are 399959099.
                MERGE SORT
Time measured: 31.250000 ms.
The number of iterations are 1207105.
The number of swaps are 276609.
                Quick SORT
Time measured: 15.625000 ms.
The number of iterations are 758449.
The number of swaps are 348114.
                COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 175999.
The number of swaps are 40000.
                RADIX SORT
Time measured: 15.625000 ms.
The number of iterations are 520040.
The number of swaps are 160000.
```

The number of N is 40,000 and the range is also kept 8,000.

Case 9: High N, High Range

```
                BUBBLE SORT
Time measured: 17531.250000 ms.
The number of iterations are 799980000.
The number of swaps are 400941956.
                OPTIMIZED BUBBLE SORT
Time measured: 17218.750000 ms.
The number of iterations are 799979810.
The number of swaps are 400941956.
                SELECTION SORT
Time measured: 1750.000000 ms.
The number of iterations are 799980000.
The number of swaps are 39992.
                INSERTION SORT
Time measured: 1031.250000 ms.
The number of iterations are 39999.
The number of swaps are 400941956.
                MERGE SORT
Time measured: 31.250000 ms.
The number of iterations are 1207198.
The number of swaps are 276702.
                Quick SORT
Time measured: 15.625000 ms.
The number of iterations are 704571.
The number of swaps are 364068.
                COUNT SORT
Time measured: 0.000000 ms.
The number of iterations are 225535.
The number of swaps are 40000.
                RADIX SORT
Time measured: 0.000000 ms.
The number of iterations are 640050.
The number of swaps are 200000.
```

The number of N is 40,000 and the range is also kept 40,000.

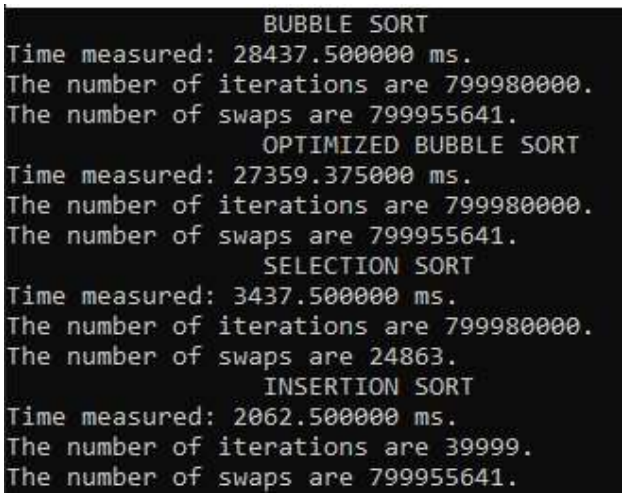
Analyzing the cases with Low N, it could be seen that even though that the array is so small, the first four algorithms are at a disadvantage. However, it could be noticed that they stay constant with the changing range since they don't care about how big the number is and how many digits it has. The effect of the increase in range could be clearly seen with the increasing swaps and iterations in radix and count sort. However, the fastest algorithm in this case is merge and quick sort, with the least number of swaps and iterations. The proper computational time was not comparable, as all of these algorithms were very fast so no time was there.

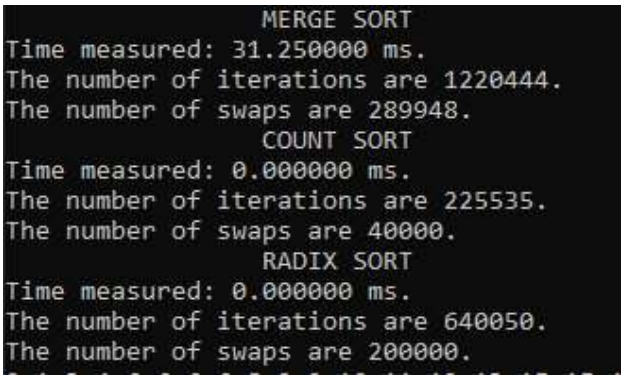
Analyzing the cases with Medium N, it could be seen that the general trend is the same here as in the cases of Low N. However, Merge sort takes significant amount of time, such that it is measurable. Also the number of iterations has already reached in millions for bubble sort and similar, with only 8000 elements in the array. The difference in time complexities is measurable here between the $O(n^2)$ and $O(n \log(n))$ now. For the high range cases, the difference between $O(n \log(n))$ and $O(n)$ is not significantly clear as of now.

Analyzing the cases with High N, it could be seen that the difference in all the time complexities of these algorithms is apparent. There is still no proper measurable processing time of count and radix sort, but it should be taken into consideration that the array size is still only 40,000. This number is about 100,000 times smaller than 1 billion and by assuming linear increase in time, so it would take about half an hour to sort the array with a billion elements. It would have been fun to test these algorithms at higher N.

The count sort is and will be superior when there is not much difference between the range of the data set and the size (N) of the array or when the range is smaller. The radix sort will be overtaking when the range is much larger than the size of the array, for example, Low N and Medium/High Range.

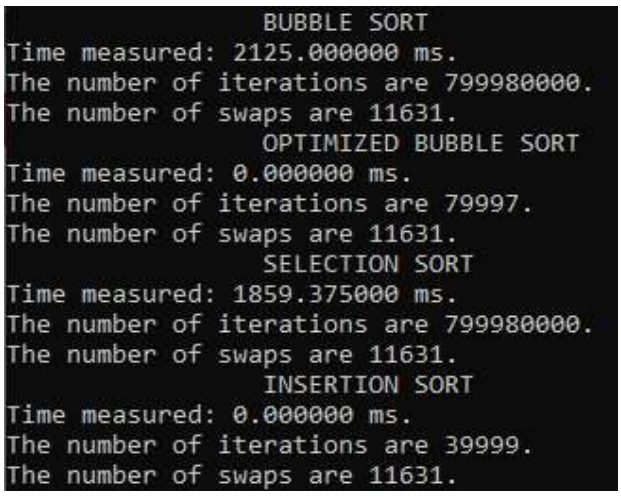
Case 10: Reverse Sorted (Worst Case)

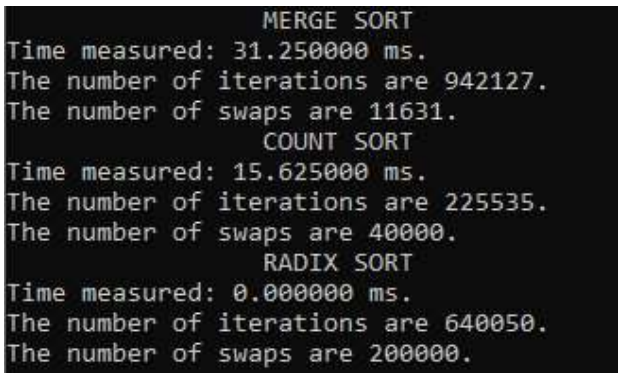




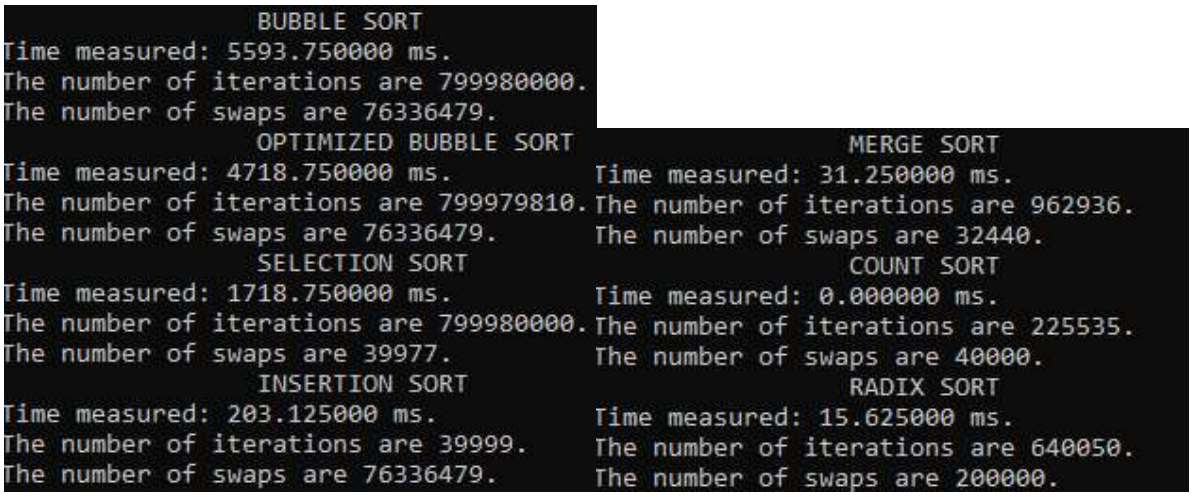
This test was carried out on a reverse sorted array. So, the placement of the elements was completely wrong. This was done by first correctly sorting the array and then putting them in another array but in the opposite direction. Then that array was used in the sorting algorithms. This is the case of High N and High Range since that gives distinct time and number of swaps and iterations differences. It could be noticed that there was no difference for the radix and count sort. For all the other algorithms, everything increased as expected since this is the worst case possible. The only exception is the number of swaps for the selection sort, which decreased due to how tat algorithm works.

Case 11: Almost Sorted



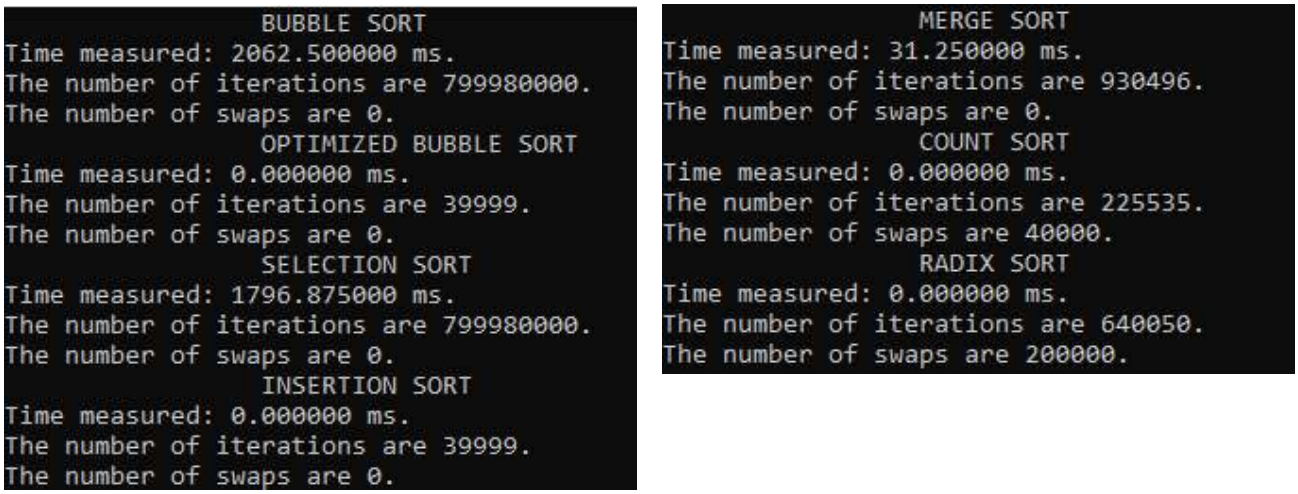


This case is one of the types of almost sorted array. This was made by sorting the algorithm but then swapping each element with its adjacent element, so it becomes almost sorted. The first and foremost thing that could be noticed is that how the number of swaps for all the algorithms is the same in this case and much lower than the random array as well. The number of iterations for optimized bubble sort are lower too, which makes it much faster as seen. Count and Radix Sort remain constant since the case used is the same (High N and High Range) and they are dependent on that.



This is another case of almost sorted array. The random array (High N and High Range) was sorted for the starting 90% of the elements and the remaining last 10% were not sorted. Since most of it is sorted, both the bubble sorts too much lesser time but still used a great deal of iterations and swaps. Merge, Count and Radix Sort remain constant since of the above stated reason. **Quick Sort has to be eliminated** for these cases, since it was giving an error for these cases, and it couldn't be figured out what the problem is.

Case 12: Already Sorted (Best Case)



This is the last case that was tested on the algorithm. This is the best case, since the array (High N and High Range) was already sorted, but it was put in the algorithms anyways to test their response to the sorted algorithm. It could be noticed that there were no swaps for the initial algorithms. The number of iterations is minimum for the optimized bubble sort. There is again no difference for count and radix sort since they are based on size and range. It could also be seen that some of the algorithms still took measurable amount of time because of their number of iterations.

It could be seen how the bubble sort is the worst algorithm in all cases 😂. Also that the radix and count sort are the reigning kings (except the best cases).