**CSC 382**

**Konstantin Novichenko**

**Lab 1**

Insertion Sort & Merge Sort

1. The program source code is in ‘./Source.cpp’
2. The sample output of the program is located in “./Images/Sample Output.PNG”
3. Original output data from the terminal is located in “./Data/ Results.txt”
4. Output was processed in the form of tables and was saved as MS Excel spreadsheet in “./Data/ Lab 1 Experimental Data.xlsx”
5. Data from the tables was used to build 8 graphs. All graphs are located in “./Images/”

Files to review:

* './Images/Sorted Array - Execution Time.PNG'
* './Images/Sorted Array - Steps.PNG'
* './Images/Reversed Array - Execution Time.PNG'
* './Images/Reversed Array - Steps.PNG'
* './Images/Random Permutation Array - Execution Time.PNG'
* './Images/Random Permutation Array - Steps.PNG'
* './Images/Average of 50 Instances of Random Numbers - Execution Time.PNG'
* './Images/Average of 50 Instances of Random Numbers - Steps.PNG'

1. To calculate the approximate value of the constant, I used Vernier Graphical Analysis software that allows to perform a curve fit by given function.

* Data Set for Vernier Graphical Analysis is located in './Images/Data Set for Graphical Analysis.PNG'
* The Insertion Sort curve fit uses the equation: **C(x^2)**, where **x** represents **N** and **C** is a constant.

Approximate value of **C** from the curve fit: **0.985 +/- 0.1271**

The graph and the results of the fit are located in './Images/Insertion Sort Curve Fit Graphical Analysis.PNG'

* The Merge Sort curve fit uses the equation: **C(x(log(x)))**, where **x** represents **N** and **C** is a constant.

Approximate value of **C** from the curve fit: **34.39 +/- 0.2449**

The graph and the results of the fit are located in './Images/Merge Sort Curve Fit Graphical Analysis.PNG'

Original Vernier Graphical Analysis project is located in ‘./Data/ Lab 1 Data Analysis.ga3’

1. GitHub repository: <https://github.com/konstantinNovichenko/Algorithms-Class---Insertion-Sort-and-Merge-Sort>

Conclusion:

According to my data, Insertion Sort is efficient in cases when N is small and when the array is already sorted or very close to being fully sorted. Merge sort is superior when the numbers are randomly shuffled in the array and when the value of N is big (even though the value of constant C is greater for the Merge Sort algorithm).