**Programming Project Report**

Name: Blake Fasse

Date: 6/22/2017

**Academic Integrity Statement**: I pledge that I have neither given nor received unauthorized help on this programming assignment.

**Problem Statement:**

The goal of the assignment was to create faster sorting algorithms that are based on the quick sorting algorithm. The program takes in the name of the text file that the user wants to use and then a number corresponding to which type of sorting algorithm would like to be used. Once the program has those two inputs from the user it will output the time it took the program to complete the sorting algorithm and it will tell the user that the sorting algorithm has been sorted as a way to tell the user that the sorting algorithm used worked. Error checking was used to make sure that what the user typed in matches with the name of the text file and that the user typed in a correct number when asked which algorithm they would like to use.

**Design:**

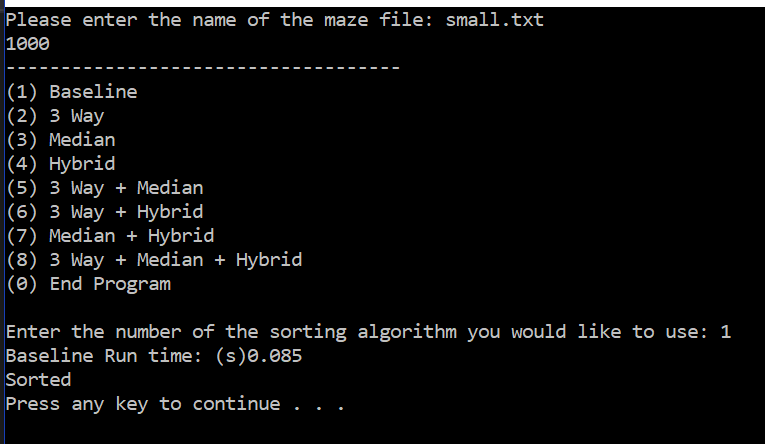
There are a total of 19 functions in this program. Each algorithm has its own partition and quick sorting algorithms and each set of algorithms is separated but comment lines so that the different algorithms functions don’t get mixed up together. With there being eight different sorting algorithms, the baseline quick sorting, the 3 way quick sort, the median partition, the hybrid quick sort, a combination of 3 way and median, 3 way and hybrid, median and hybrid, and a combination of 3 way, median and hybrid algorithms. There are no pros to the way this was designed and the only con is that since this is the last day there was no time to make sure that any of the functions are not being repeated.

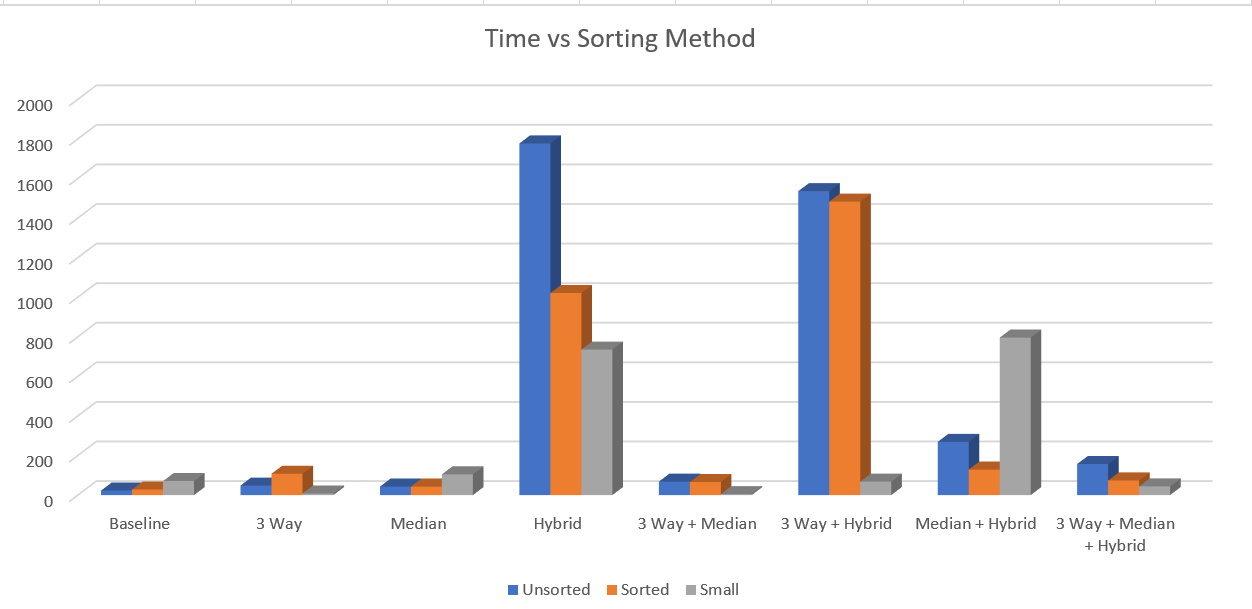
**Implementation:**

This assignment was not an easy task. Instead of making separate versions of all the algorithms and then putting them together at the end the final combination of all three algorithms was designed. Sample code was given for the baseline quick sort and partition algorithms. This code was then expanded on by adding conditions within the quick sorting algorithm that if the size of the input is too small then a better sorting algorithm is called. The partition function was expanded on to make it partition the array into three parts instead of two and also allowed to partition better by finding the correct median of the array. Development time took the full two weeks.

**Testing:**

Testing was done by taking all the times it took each algorithm to sort three different text files and putting them in a chart and comparing them to a table of times that was provided. The three text files contained arrays that the program would read in of sorted, unsorted, and have a few unique values. Each algorithm was used for each of the arrays. Special cases that were tested were mainly to show that the program rejects names for text files that are not provided and that the program would not take anything but the numbers provided when asking for which algorithm is to be used. The program does rejects names of files that are not in its directory and the program also rejects *numbers* that do not correspond with the algorithms but if the user types in a character or a string then the program goes into an infinite loop. A picture of the graph of times for each algorithm on each array and a sample of how to output screen looks on the console is provided below.





**Conclusion:**

All in all this program turned out to be a success. If this assignment were to be done over again then I would make sure that none of the functions are repeated. This assignment took the full two weeks to complete.