Ioannis Konstantinou
103043778

Algorithms

Q. How is **Shell Sort** better than **Insertion Sort.**

A. With shell sort being a variation of Insertion sort, they work very similar. Difference being that with Insertion sort elements are only moved one position at a time and require more movement to find its correct position. Shell sort uses Insertion sort but moves the position of the elements of an array according to the gap size. The gap size is then reduced after every iteration. By moving elements with such a gap, they move closer to their correct position sooner requiring less iterations to sort the array.

Q. Time/Analyse the searches and compare against the known Big O notation for the Linear and Binary searches.

A. The Big O notation is used to describe the performance of an algorithm. It specifies the worst-case scenario that an algorithm takes to complete its task. It's extremely handy when comparing algorithms that require large number of steps to execute

For Linear search the Big O notation (worst-case) is O(N). When performing a linear search amongst N elements can take N iterations for worst-case. The linear correlation is between the number of iterations for worst-case and the number of elements of a dataset searched. So worst case for a linear search would be that the array is iterated by its length

For Binary the Big O is O(Log(N)). When performing a binary search, after each iteration half of the data set is discarded. This means that to search a Data set of 500 values it will only iterate one more time than a Dataset of 250 values.

In my Algorithms, I used a count variable to count the number of times each search algorithm ran to find the specified values given to us in this exercise from the Dataset provided in the unsorted_numbers.csv file. I also used the Stopwatch class to time how long each algorithm ran for. Here are my findings.

```
Linear Search
Looped 57282 Times
Elapsed Time is 00:00:00.0004652

Binary Search
Looped 118 Times
Elapsed Time is 00:00:00.0000062
```

As you can see that Binary search looped a significant amount less to find the specified values compared to a Linear search. This is also shown with the elapsed time it took to perform said task.

Ioannis Konstantinou
103043778

Could Merge Sort be run as a multi-threaded application? Would there be likely to be a performance gain in doing so? Why/Why not?

A. Multi-threading is a technique used by a Computers Processor (CPU) to allow multiple parts of a program to run simultaneously. Each part of the program runs in what is called a **Thread.** Multiple threads execute concurrently and share resources amongst each other such as files, data and memory. This technique is used to speed up a program's execution time. For example, by splitting a program into two parts and running them concurrently will half its total execution time.

With Merge Sort being a divide and conquer type algorithm, it works by dividing its data recursively by 2 or more, once each divided section is sorted, they are then merged in a sorted manner. For this reason, Merge sort is well suited for Multithreading. By running each divided section of a Merge sort in its own thread (Multi-Thread) would decrease the total execution time it would take for it to sort its Dataset compared to running each section sequentially in a single thread.