Algorithm Assignment 2 Report.

Contents:

Introduction:

The given assignment was to create a program which uses algorithms to search and sort through a handful of text files/ data.

Application Description:

The program starts of by creating a list to hold record's, it will then fetch the data needed to fill said record and convert them to doubles if needed, this is done to make integers easier to sort.

Once it fetches the data the program creates the objects for the record by adding the respective line form each text file (for example, record one will take the first line from each text file and the second record would take the second line and so on).

The application at this point begins the main program, meaning what the user will see, I use a large switch statement to take user inputs and create a menu, it begins by asking the user what they would like to do (sort an array in ascending and descending order, search for a specific value in an array or too search for all records connected to a chosen month) they then follow the given rules if choosing to sort an array, it will first ask the user which array they would like to sort and then ask them if they would like to sort it in ascending and descending order, once choosing the array the program will sort said array with its respective quick-sort algorithm (there is are 4 algorithms, 2 for sorting integers and strings and then another 2 for sorting (either integer or strings) in descending and ascending order.

If the user selects the option to search through a specific array, it will first ask them which array (or text file) they would like to search through and then ask them to type in the value they are looking for.

for example the user may choose to search through the record of years, and then choose to look for any existence 2017, if the record holds any earthquakes in 2017 let's say 5 times, the program will print out 2017, 5 times and state there were 5 earthquakes during 2017, this is done with a linear search.

Finally if the user chooses the last option it will prompt them to enter which month they would like to search through and then give back to them every full record with said month, for example if you was to enter July, the program will find every earthquake that happened in July, and print each record (meaning it will print each full record, so the earthquake on the 1st of July and its magnitude, latitude, longitude and so on, it will print each day and its respective information that is in the month July) this again uses a linear search.

Description of the algorithmic choices:

In my program I use a quicksort algorithm to sort the data in each array, I use a collection of 4 different types of quicksort algorithms, 2 algorithms to sort integers and 2 algorithms to sort strings each type has one algorithm to sort in ascending order and one to sort in descending order.

I used a quick-sort algorithm for sorting data as it's time complexity while sorting is rather good, running at a O-notation of (n log (n)) consistently meaning its best, average and worst time complexity will all be fairly fast (n log (n)), although for example a bubble sort and insertion sorts best time is (n) its average time complexity (meaning the most likely to occur) is (n^2) which is rather bad/ slow, meaning Quicksort is overall more reliable, almost guaranteeing the time complexity to be rather fast, the only time this may not stand is in its worst case scenario where a merge sort would but better however a worst case scenario is not likely to happen, not only this but a quicksort has a more efficient space complexity over merge sort running at (log(n)) where as a merge sort only reaches a (n) O-notation meaning the trade-off seems better when using a quick-sort.

I use a linear searching algorithm in my program for searching through the arrays of data, there are 2 linear searches, one for integers and one for strings. I use a linear algorithm as the amount of data we are searching through is not that large, therefore it would proof pointless to create a say binary search for such a small amount of data. The downside to using a linear search (average, O(n)) is compared to say a binary search (average, log n) it is not as efficient recording its time complexity, however a linear search would use less memory, which when I would say searching through a small amount of data would be helpful, on terms of space complexity they both share a worst case O-notation of O(1). Finally linear algorithms only require sequential access.

Video:

https://youtu.be/y0f5eF3QJ0U