**Maxwell Butler – Algorithms & Complexity Assessment 1 Report**

**Contents**

* Basic Design for the Application
  + Written description
  + Implementation of Tasks 1-7
* A description of the algorithmic choices I made for the application
  + Table providing the number of steps each algorithm took
* Reference list of items I used for learning

**Basic Design for the Application**

* Written description
  + Line 4
    - Imports the necessary systems.
  + Line 7
    - Program class is defined.
  + Line 9 and lines 40-50
    - Each text file is loaded into its own named array using the LoadData() method. The arrays are then defined in an integer array.
  + Lines 13-21
    - The LinearSearch() method is used later to search for a user-defined value. The search works by iterating from element 0 to the end of the array, until the value is found. If the value is not found, an error message is printed.
  + Lines 24-36
    - The CopyFileToArray() method copies each line from the chosen text file and converts them into an array. These arrays are now able to be sorted and searched through using the methods.
  + Line 39
    - An integer value, Counter1 is set 0. Throughout the program, before each algorithm that runs, the Counter adds 1 to itself. The total is printed at the end of the program.
  + Lines 54-59
    - A menu is printed out, laying out to the user their options. To analyse the chosen text file, the user must enter a correct option. If not, an error and help message is printed out.
  + Lines 63-732
    - An if statement is implemented. This couples with the user’s input (defined in the string variable ‘option’). Whatever choice the user inputted, decides on which option is chosen for the rest of the code.
    - For each option, the majority of Tasks laid out in the Assessment Brief are met. However, Task 4 has not been successfully met. Nevertheless, the rest of the Tasks have been successfully implemented. Moreover, I have printed out messages to break down to the user what sort/search method has been used on the array, as well as including the Console.ReadKey() method, to wait for any user input (this helps the program to feel more user-friendly).
* Implementation of tasks 2 to 7
  + I started by creating a Bubble Sort algorithm to sort the array into ascending order. I started with a for loop that iterated from element 0 over the length of the array, iterating every element (+1). I then created a following for loop using the iterator j that iterates through so long as iterator j is greater than iterator i. The for loops ends once both iterators are equivalent to each other. A foreach statement is then used to print out the sorted ‘items’ in the array.
  + When sorting an array into descending order, the algorithm is essentially the same as the previous one (Bubble Sort). However, if iterator j is greater than iterator I, the loop will end.
  + To find every nth element (either 10th or 50th), I created a for loop that iterates over the length of the array and every 10th element (by multiplying the iterator by 10). Within this for loop, for every 10th element, a comma is placed after it.
  + The LinearSearch() method works by iterating across the array until the user inputted ‘target’ integer value is reached. Once reached, the algorithm ends, printing out its location (indicated by which element it stops at).

**A description of the algorithmic choices I made for the application**

* The ascending and descending sort algorithms are both Bubble Sort algorithms. I knew that once I worked out how to create one, I could do essentially the same for the other (minus a few tweaks). Hence, why they are very similar algorithms. In terms of time efficiency, Bubble Sort is a very inefficient, as it has a worst-case efficiency of O(n^2).
* Finding every nth value was fairly simple. For every nth element in the array, print the element. It therefore only required a simple for loop to get working.
* The LinearSearch() has a less time efficiency (O(n) to a Binary Search. Therefore, I aimed to implement a Binary Search instead, as it is more time efficient (as it takes O(log n) complexity. However, I could not work out the code.
* My program as a whole is messy. It is very space inefficient. I struggled to organise my code in a more efficient manner. I wanted each algorithm to have its own method. I failed to implement this idea. This would mean that I could define each method and thus, call them on a single line. As opposed to copy and pasting the code for each algorithm for each if statement branch.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms  no. of steps | Linear Search | Bubblesort | nth value |  |
| 256 | 256 | 65,356 | 2.56 (every 10th) |  |
| 2,048 | 2,048 | 4,194,304 | 40.96 (every 50th) |  |
| 2,304 (merged) | 2,304 | 5,308,416 | 46.08 (every 50th) |  |
|  |  |  |  |  |

**Reference list of items I used for learning**

# References

Butler, M. (n.d.). Lecture Notes.

GeeksForGeeks. (n.d.). Retrieved from GeeksForGeeks.org: https://www.geeksforgeeks.org/

W3Schools. (n.d.). Retrieved from W3Schools.com: https://www.w3schools.com/