Hardik Anand

Professor Sharon Gumina

ITEC 590-J56

18th June 2020

**Final Project**

**This application implements the 3 famous sorting algorithms**, which are, 1. Selection Sort, 2. Bubble Sort, and 3. Insertion Sort. The application takes input from the user, the input being the name of the text file which needs to be sorted (ascendingly), full of integers. The text file is read into a string list, as the numbers in the file are stored in a new line, so to accommodate the “\n” or the new-line character, we use a string list. The string list is then converted into a new string list, with using the “strip()” functionality of the string, which gets rid of the new line character. The list is then checked for anything other than an integer value, if the list contains only integers, everything is appended to a new list by casting each item to an int value, otherwise an error message is displayed at the screen asking the user to use a different file or modify this file.

Then, the user is provided with a menu with several options: 1. Selection Sort, 2. Bubble Sort, 3. Insertion Sort, 4. Print out the average time for all the algorithms, 5. Print out all the steps for each algorithm, 6. Change the file (Current file: filename), 0. Exit. Options 1, 2, and 3 perform the algorithm and display the sorted and unsorted lists, along with the time taken by the algorithm to sort the list. Option 4 is used to print out the average time for each algorithm, and this only gets updated when an algorithm has run at least 3 times. Option 5 prints out the steps for each iteration in the sorting algorithm. Option 6 is used to change the current file to another file, and lastly, option 0, is used to exit the program.

The resources that I used for this were the libraries: copy, os, sys and time. The copy library was used to copy a list into another list as it is. The os library was used for error checking, basically to check if the file being input by the user exists in the working directory or not. The sys library is used to exit the program, and is also used in the typewriter function, to write to the screen and flush the buffer. Finally, the time library is used to record the internal clock’s time at any given moment, and also is used in the typewriter function to make the process sleep for 0.016 s.

Some of the web resources I used were, how to use the time library’s “time.time()” functionality, I used the web to brush up my memory about insertion sort. I also found it helpful to look up how to use the sys library to use the typewriter function.

**Moving on to the output of the program. As described above, the program takes input from the user as shown below in the image.**



After the input is given, the user is displayed with a list of commands he/she can enter into the console.

A picture containing clock, black

Description automatically generated

When 1 is selected, the console displays the sorted list using selection sort, unsorted list and the time taken by the algorithm to get executed. Note: the time taken for this list would be 0.0 ms because, the list is very small (10 elements) and since most of the modern laptops have very high processing power, along with SSD’s it takes very little to no time when sorting a list this small. You will be able to see that the time thing works in the video that I will upload with this.

A picture containing holding, clock

Description automatically generated

When 2 is selected, the console displays the sorted list using bubble sort, unsorted list and the time taken by the algorithm to get executed. The same thing will happen for the time as above, i.e., 0.0 ms.

A close up of a screen

Description automatically generated

When 3 is selected, the console displays the sorted list using insertion sort, unsorted list and the time taken by the algorithm to get executed. The same thing will happen for the time as above, i.e., 0.0 ms.

A picture containing holding, clock, phone, room

Description automatically generated

When option 4 is selected, the program display the average time for each algorithm to run, and this only works if an algorithm has run at least 3 times. Note: in this case as we are using a very small list, the average time will remain 0.0 ms, as the total time taken on every run would be 0.0 ms. I will prove that this function works in the video I will upload along with this.

A screen shot of a computer

Description automatically generated

Option 5 gives a mini menu with 3 options as shown below.

A clock on each of it s sides

Description automatically generated

When any of these options is selected, the console displays the steps for the algorithm to sort the list. When 1 is selected:

A clock mounted to the side

Description automatically generated

It will look something like this for each of the algorithm. Note: I’m not going to post all the pictures of the steps, as it will hog all the space up and throw me way over the page limit. Everything will be demonstrated in the video.

After opting for option 6, we can change the file currently being used in the program.

A picture containing clock, meter

Description automatically generated

As you can see from the above picture that the filename has been changed to test.txt from sort.txt, in all the other pictures. Option 0 is to exit the program.

**Let me talk about the code and implementation.** Firstly, copy, os, sys, and time libraries are used for some functionality, as described on the first page. There is a function by the name of typewriter(), which takes in a parameter message, and a for loop is run that iterates through every character in the string, to print out each character to the console, and making the program sleep for 0.016 seconds. The function read\_file() takes in input from the user and then the input variable is appended with “.txt” so that the program knows, we are trying to read from a text file. Then we check to see if the file exists in the working directory. If it does, the file is opened for reading and where the contents of the file are transferred to a string list, which is then transferred to another string list without the “\n” character, and finally, the contents are converted to an integer after some error checking (if line.isDigit()), and transferred to an integer list. The list, exists (a Boolean condition for error checking later on), and the name of the file is returned from the function.

The avg\_time\_taken() function computes the average of a given list, and this is used to display the average times by clicking option 4 as described above.

The function update\_dict() is used to update the dictionary’s value created in the main function, which stores all the average time for each of the algorithm. This function can only be called if the algorithm has been run at least 3 times. This function takes in a dictionary, a key and a value as its parameters, and returns the dictionary after execeution.

The selection\_sort() sorts the list ascendingly, by creating a for loop and replacing the current element with the element that has the lowest value towards the right of the current element. The current element switches to the current + 1’th element after the swap.

The step\_print\_ss() function is the same as above, with just printing out what is being swapped right before each swap is made.

The bubble\_sort() sorts the list ascendingly, by swapping the adjacent elements if they are in wrong order. This keeps on happening until the whole list is sorted.

The step\_print\_bs() function is the same as above, with just printing out what is being swapped right before each swap is made

The insertion\_sort() sorts the list ascendingly, by checking if the 0th element is smaller than the 1’th element, if not, the 1th element is put towards the beginning of the list and thus a sorted part is created. The list is now checked for elements if the elements towards the left are bigger, if true, the element is sent to the sorted part and is sorted over there while comparing the values of the elements.

The step\_print\_is() function is the same as above, with just printing out what is being swapped right before each swap is made

The list\_printer() function prints out the sorted and unsorted list. It prints it out using the typewriter function if the length of the list is less than 30, otherwise it is printed using the print statement.

The print\_new\_list() function prints out only the sorted list. The rest is exactly the same as above.

The main function is designed in a way to get variables returned from the read\_file function and run the program after error checking as explained in the video. Several variables are created for this program, and a while loop is used to keep printing out the command menu until Option 0 is pressed.

**The learning outcomes** from this program was that I truly believe that I have become a better python coder because of it’s user-friendliness, when compared to other languages like C++ and Java. I learned how to compute the total time an algorithm takes, and also learned about how to give your console output a typewriter effect. I learned that working with File I/O is actually very useful and not at all hard when You understand it properly. I learned about various exceptions and also learned how to use some of python’s in-built functionality like, the os.path.exists(name), which checks if the a file exists in the working directory.

I have become a better and more experienced coder after completing this course, and specially the final project. I got a better grasp of using loops and returning solving problems by returning multiple variables, which I had never done before using python. I also learned that selection sort and insertion sort have similar time frames in which they get executed, while bubble sort takes more time than both of these to run, so in other words, I learnt more about the algorithm’s time constraints and which one is efficient for what type of data.

I learnt more about random numbers, and the random.sample functionality which is actually very useful for programs like this. I confirmed that Selection sort is actually better than bubble sort, like I always thought, but never had any proof. I learnt a lot of new things from this project, and everything can not even be specified in this word document. Things like getting motivated, feeling the sense of accomplishment and much more was achieved through this project and this course. This led me to believe that the 4 years of college that I have spent learning to code, have not at all been a waste, and I was easily able to switch to python, and honestly I would like to keep it that way from now on. I am very thankful for this course, and my professor, who always helped with anything I sent her.