KEVIN RODRIGUEZ

LAB 2 REPORT

***Introduction***

For this lab, I was given a file of 10 million passwords. The problem is to create a python program that would read the 10 million passwordsand create a linked list with each node containing two values, one being the password (string) and the other to hold the number of times that password was found in the file (int). After creating the linked list, we had to create an algorithm that would sort the list from most used to less used passwords using the bubble sort method. The second problem was to create a dictionary of the password file. After populating the dictionary, we have to sort it using the merge sort method.

***Solution/Design Implementation***

Solution A:

Created a program that would read the file which uses the partition method in order to skip the usernames and the blank spaces. After the partition, since it returns a tuple I only add the value in the second (2nd) index which is the password. In my ‘add’ method, I created an unsorted list that adds a new node whenever it doesn’t have an existing node with the same password. If there is an existing node with that password, I just add the ‘count’ value of that specific node by 1. After creating the unsorted linked list, I passed it into my bubble sort method. In this method, I have a reference to the previous node, the current node, and the next node in order to do the swaps. This method as well has an improvement in efficiency by having a boolean flag in order to check if there has been a swap. If there hasn’t been a swap, it means the list is sorted and there is no need to keep traversing through the list, therefore it leaves the loop. Since this a bubble sort implementation, the Big O of this Solution is O(n^2).

Solution B:

In order to create a Dictionary, I implemented a method in which it reads the password using the partition as mentioned in Solution A. and checked that specific password index. If their existed an index with that password I simply add one, if not, I create a new index with that password with a value of 1. In order to merge sort the dictionary, I created a method in which it reads every single index, and creates a node with the value of the password (string) and the count that password has (int). After creating the linked list, I created a merge sort function in order to satisfy the requirements for Solution B. The way I created my merge sort is by creating a variable called ‘left’ and assign the value as the head of the list. Then I would find the size of the list and dividing it by two and store it in a variable called middle. I traversed through the list until I am one Node behind the middle, and assigned the next Node as the right side of the list, and in order to unlink the left side from the right side, I pointed middle.next or (left.next after the loop) to None. I then recursively called both lists (left and right) and then insert them in a method that merges both lists and returns a merged sorted list. The Big O of this Solution is O(log n) for the sorting of the list and O(1) when trying to access an index in the dictionary.

***Experimental Results***

In order to test my program, I conducted 5 different tests with different size files. I tested each program with text files holding 55, 150, 1,000 , 20,000 , and 50,000 passwords. Below I inserted a line graph that shows the curve for each sorting method which depends on how long it takes for the program to execute.



As you can see, the line for bubble sort has the curve that a bubble sort method should have O(n^2). The bigger the file, the longer it takes for the program to execute. On the other hand, you see the line for merge sort which holds the same way a O(log n) line should have.

***Conclusion***

In this lab, the main thing that I learned is how to sort linked lists. I’ve had dealt with sorting arrays, but I now developed the technique and skill to sort linked lists. Also, since I am new with python, I learned how to read files and different ways to open a file by either just reading it or writing on it. Another concept I learned is about partition and the tuple value. Lastly, I learned how to use dictionaries in python. I learned that dictionaries are essentially hash tables. Which is great that my professor included this in this lab since it is the next topic in the class. Good introductory to it!

***Signed Academic Honesty Certification***

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class