Ricardo Godoy

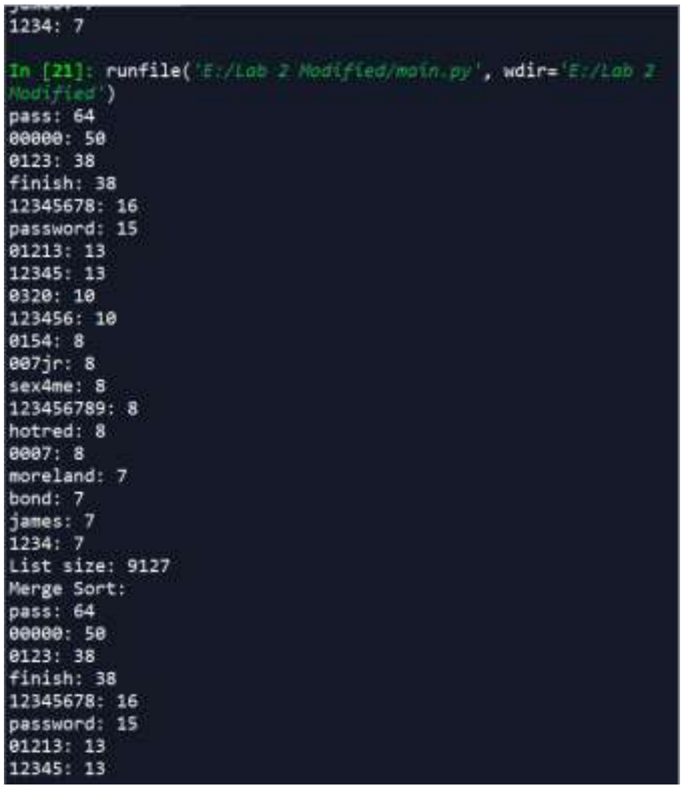
Lab Report

The purpose of this program is to read a text file that contains the usernames and passwords of Internet users, and return the twenty most repeated passwords inside that file. The program implements a Linked list, adding the passwords to the list. As the program reads the file and obtains all the passwords, it traverses the linked list created to check if the password is already on the linked list. In case the password is already in the list, the program adds one to the object's count property. If not, then the program creates a new node with the corresponding properties (password, count, next) of that node, and adds it to the list. This is accomplished in two ways: traversing a linked list to search if the password is already on the list, and using a dictionary to avoid traversing the whole list. After creating the list, the program will sort the list in descending order using both bubble sort and merge sort, respectively, and then print the 20 most used passwords along with the number of times they appear in the file.

I attempted to solve the program by first opening the file that contained all the passwords and then read it line by line, assigning the actual password to a variable. From there, I traversed the linked list each time I read a password in order to search for it in the linked list and to meet one of the cases. After traversing the list and depending on whether the password was already on the linked list or not, the program will perform the determined operation: increment the count or add a new node to the list, respectively. After the list was created, the program performs bubble sort on the list, sorting the 20 most repeated passwords in descending order. Bubble sort first compares the two first nodes of the list and swap them if the first node's count is greater than the second one. This operation is applied throughout the entire list until all elements are sorted correctly.

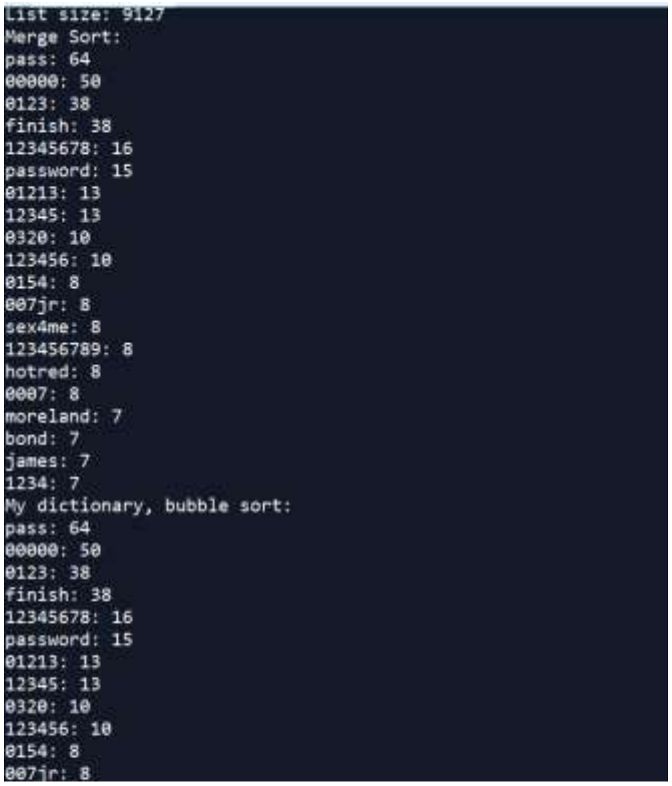
Along with bubble sort, merge sort is also implemented in the program. The merge sort algorithm consists on splitting the list, and sort the resulting lists recursively, until it reaches its smaller size. Once all the lists are sorted individually, the program merges the resulting lists into one sorted linked list.

The first test was with a list of 100,000 users and passwords. The text file contains 100,000 username and passwords. The program will still work for users that have no password. In order for the program to work, the text file must contain first the username then the psssword, separated by a blank space in the same line I decided to use this test in order to run the program faster, but still without losing focus of the algorithm. The first picture displays the results of the list created by the method of traversing the loop and inserting the password or incrementing the count of the node sorted with bubble sort. The second image displays the same list sorted by merge sort. The third image displays the list created with the dictionary algorithm, sorted with bubble sort and merge sort respectively. Here are the results:

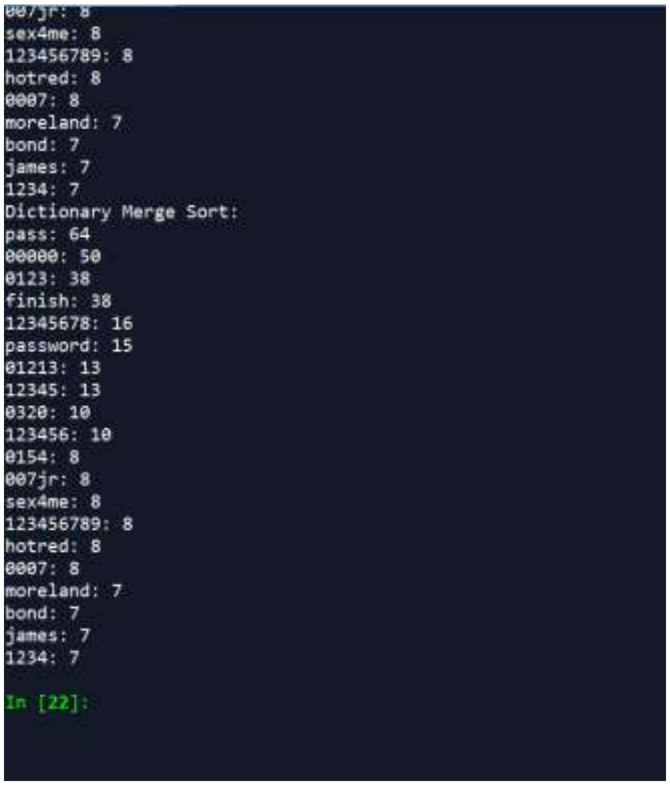


```
1234: 7

In [21]: runfile('E:/Lab 2 Modified/main.py', wdir='E:/Lab 2
Modified')
pass: 64
00000: 50
0123: 38
finish: 38
12345678: 16
password: 15
01213: 13
12345: 13
0320: 10
123456: 10
0154: 8
007jr: 8
sex4me: 8
123456789: 8
hotred: 8
0007: 8
moreland: 7
bond: 7
james: 7
1234: 7
List size: 9127
Merge Sort:
pass: 64
00000: 50
0123: 38
finish: 38
12345678: 16
password: 15
01213: 13
12345: 13
```



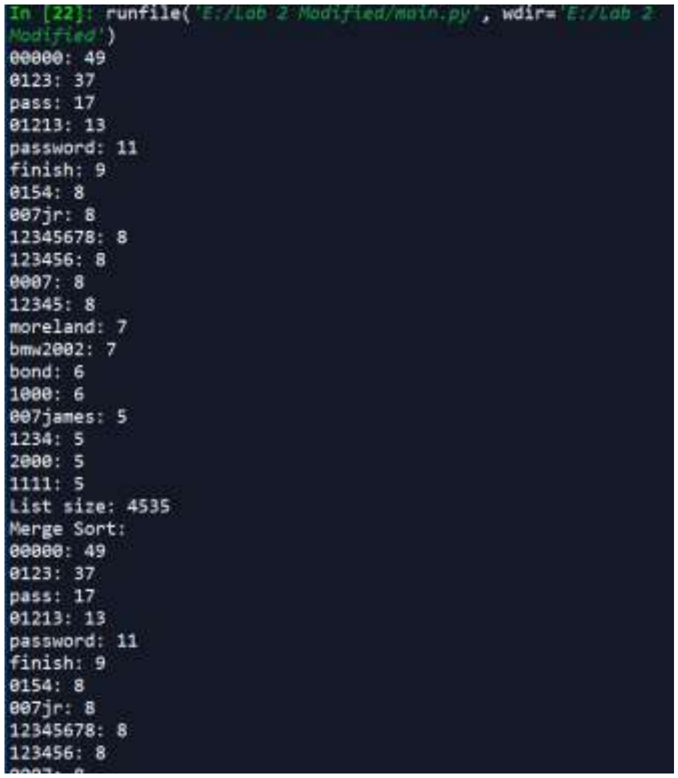
```
List size: 9127
Merge Sort:
pass: 64
00000: 50
0123: 38
finish: 38
12345678: 16
password: 15
01213: 13
12345: 13
0320: 10
123456: 10
0154: 8
007jr: 8
sex4me: 8
123456789: 8
hotred: 8
0007: 8
moreland: 7
bond: 7
james: 7
1234: 7
My dictionary, bubble sort:
pass: 64
00000: 50
0123: 38
finish: 38
12345678: 16
password: 15
01213: 13
12345: 13
0320: 10
123456: 10
0154: 8
007jr: 8
```



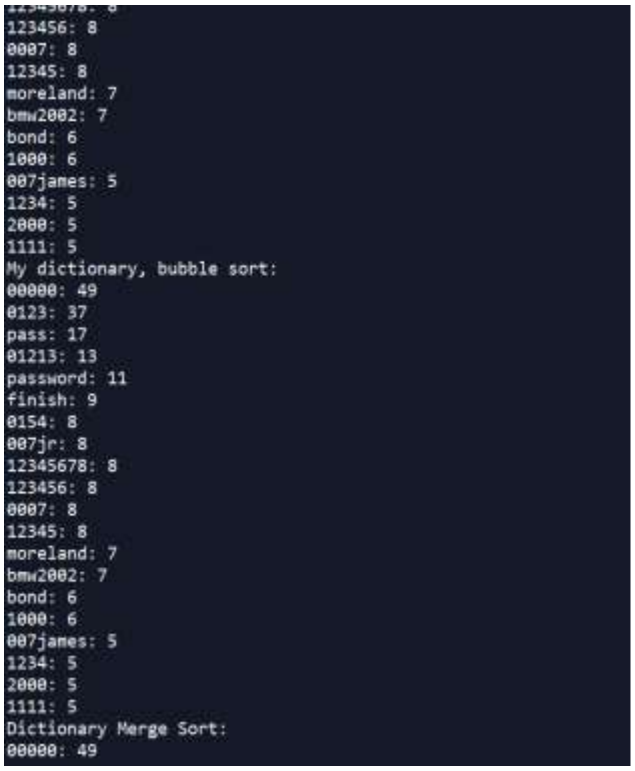
```
007jr: 8
sex4me: 8
123456789: 8
hotred: 8
0007: 8
moreland: 7
bond: 7
james: 7
1234: 7
Dictionary Merge Sort:
pass: 64
00000: 50
0123: 38
finish: 38
12345678: 16
password: 15
01213: 13
12345: 13
0320: 10
123456: 10
0154: 8
007jr: 8
sex4me: 8
123456789: 8
hotred: 8
0007: 8
moreland: 7
bond: 7
james: 7
1234: 7

In [22]:
```

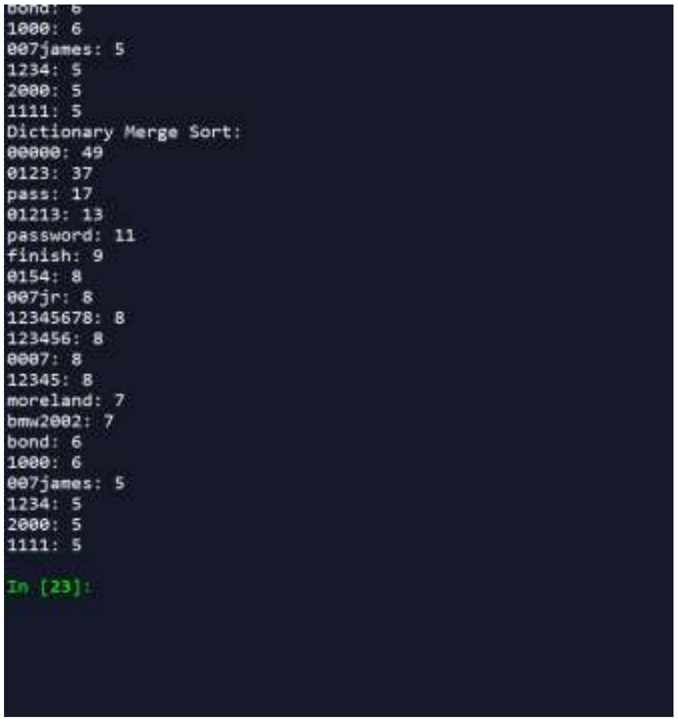
My second test was to decrease the size of the text file to only 50,000 users and passwords, or 50,000 lines. Bubble sort, merge sort for both solutions, respectively. Here are my results:



```
In [22]: runfile('E:/Lab 2 Modified/main.py', wdir='E:/Lab 2
Modified')
00000: 49
0123: 37
pass: 17
01213: 13
password: 11
finish: 9
0154: 8
007jr: 8
12345678: 8
123456: 8
0007: 8
12345: 8
moreland: 7
bmw2002: 7
bond: 6
1000: 6
007james: 5
1234: 5
2000: 5
1111: 5
List size: 4535
Merge Sort:
00000: 49
0123: 37
pass: 17
01213: 13
password: 11
finish: 9
0154: 8
007jr: 8
12345678: 8
123456: 8
```



```
123456: 8
0007: 8
12345: 8
moreland: 7
bmw2002: 7
bond: 6
1000: 6
007james: 5
1234: 5
2000: 5
1111: 5
My dictionary, bubble sort:
00000: 49
0123: 37
pass: 17
01213: 13
password: 11
finish: 9
0154: 8
007jr: 8
12345678: 8
123456: 8
0007: 8
12345: 8
moreland: 7
bmw2002: 7
bond: 6
1000: 6
007james: 5
1234: 5
2000: 5
1111: 5
Dictionary Merge Sort:
00000: 49
```



```
1000: 6
007james: 5
1234: 5
2000: 5
1111: 5
Dictionary Merge Sort:
00000: 49
0123: 37
pass: 17
01213: 13
password: 11
finish: 9
0154: 8
007jr: 8
12345678: 8
123456: 8
0007: 8
12345: 8
moreland: 7
bmw2002: 7
bond: 6
1000: 6
007james: 5
1234: 5
2000: 5
1111: 5

In [23]:
```

In this project, I learned how to create linked list and improve my understanding of sorting algorithms. In addition, I learned how pointers work in linked lists and how to implement different solutions to this lab's problem.

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.

Appendix

```
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 14 17:34:00 2018

@author: Ricardo
"""
# Course: CS2302
# Author: Ricardo Godoy
# Assignment: Lab 2 B
# T.A: Saha, Manoj
# Instructor: Diego Aguirre
# Date of last modification: 10/22/18


# This program reads a file that contains username and passwords and stores the password in a
linked
# list. If a password is repeated, the program increments the count in the respective node. If not,
# a new node is created with the respective properties and added to the linked list. The list is
# sorted using bubble sort and merge sort. Along with that solution, a dictionary is also
# created to store the 20 most repeated passwords and identify them in the linked list. Finally,
# the program return the 20 most repeated passwords in descending order.

# Node class
class Node(object):
    password = ""
    count = -1
    next = None
    def __init__(self, password, count, next):
        self.password = password
        self.count = count
        self.next = next


# Linked list class
class LinkedList:
    def __init__(self):
        self.head = None


# Reads the file, split and stores all passwords
def read(password_list):
    with open('10-million-combos.txt', 'r') as file:
        for line in file:
            line = line.split()
            if len(line) > 1:
                add_Password(password_list, line[1])


# Adds all passwords to a linked list
def add_Password(password_List, password):
    node = password_List.head
    while node != None:
        if node.password == password:
            node.count += 1
            return
        node = node.next
    password_List.head = Node(password, 1, password_List.head)


# Computes the size of the linked list
def get_List_Size(password_List):
    node = password_List.head
    list_Size = 0
    while node != None:
        list_Size += 1
        node = node.next
    return list_Size


# Prints the list of the passwords most repeated
def print_List(password_List):
    node = password_List.head
    count = 0
    while node != None and count < 20:
        print(node.password + ":", node.count)
        node = node.next
        count += 1


# Sorts the list in descending order, with bubble sort algorithm
def bubble_Sort(password_List, list_Size):
    for i in range(list_Size):
        curr_Node = password_List.head
        next_Node = curr_Node.next

        for j in range(list_Size - 1):
            if curr_Node.count < next_Node.count:
                temp_Count = curr_Node.count
                curr_Node.count = next_Node.count
                next_Node.count = temp_Count
                temp_Password = curr_Node.password
                curr_Node.password = next_Node.password
                next_Node.password = temp_Password
            curr_Node = next_Node
            next_Node = next_Node.next



# The following three methods form part of the merge sort algorithm
# Time complexity: O(log n)
def merge_Sort(password_List):

    if password_List == None or password_List.next == None:
        return password_List

    left_Half, right_Half = split(password_List)
    return merge_Lists(left_Half, right_Half)


# Splits the list in two
def split(password_List):
    if password_List == None or password_List.next == None:
        left_Half = password_List
        right_Half = None
        return left_Half, right_Half
    else:
        mid = password_List
        front = password_List.next
        while front != None:
            front = front.next
            if front != None:
                front = front.next
                mid = mid.next
    left_Half = password_List
    right_Half = mid.next
    mid.next = None
    return left_Half, right_Half



# Merge the two lists created above
def merge_Lists(left_Half, right_Half):
    new = Node("", -1, None)
    curr = new
    while left_Half and right_Half:
        if left_Half.count < right_Half.count:
            curr.next = right_Half
            right_Half = right_Half.next
        else:
            curr.next = left_Half
            left_Half = left_Half.next
        curr = curr.next
    if left_Half == None:
        curr.next = right_Half
    elif right_Half == None:
        curr.next = left_Half
    return new.next

# Adds all passwords to dictionary
def add_Password_to_Dictionary(password, Dict):
    if password in Dict:
        Dict[password] += 1
    else:
        Dict[password] = 1



# Inserts all passwords into the dictionary
def insert(password_List, Dict):
    for password in Dict:
        node = Node(password, Dict[password], password_List.head)
        password_List.head = node



def read_dict(password_List, Dictionary):
    with open('10-million-combos.txt', 'r') as infile:
        for line in infile:
            read = line.split()
            if len(read) > 1:
```

```
            add_Password_to_Dictionary(read[1], Dictionary)



def main():
    #Create list
    password_List = LinkedList()

    #Read file
    read(password_List)
    list_Size = get_List_Size(password_List)

    #Sort list with bubble sort and print
    bubble_Sort(password_List, list_Size)
    print_List(password_List)
    print("List size:", list_Size)

    #Sort list with merge sort and print
    new_Head = merge_Sort(password_List.head)
    new_List = LinkedList()
    new_List.head = new_Head
    print("Merge Sort:")
    print_List(new_List)

    #Create dictionary and print list sorted with bubble sort.
    list_Size = 0
    Dictionary = {}
    password_List = LinkedList()
    read_dict(password_List, Dictionary)
    insert(password_List, Dictionary)
    list_Size = len(Dictionary)
    bubble_Sort(password_List, list_Size)
    print("My dictionary, bubble sort: " )
    print_List(password_List)
    print("List size: " + str(list_Size))

    #Merge sort list created with dictionary
    new_head = merge_Sort(password_List.head)
    new_list = LinkedList()
    new_list.head = new_head
    print("Dictionary Merge Sort:")
    print_List(new_list)
    print("List size: " + str(list_Size))

main()
```