CS 2302

Lab 2.B Report

Author: David Ayala

ID: 80617838

Instructor: Diego Aguirre

TA: Gerardo Barraza

**Introduction**

The purpose of this lab was to take a file that will have a list of passwords (10 million). Once you read the file you are supposed to find the 20 most used password. Part A is doing this by using a loop and putting them into a list, while part B is about using a dictionary to accomplish this.

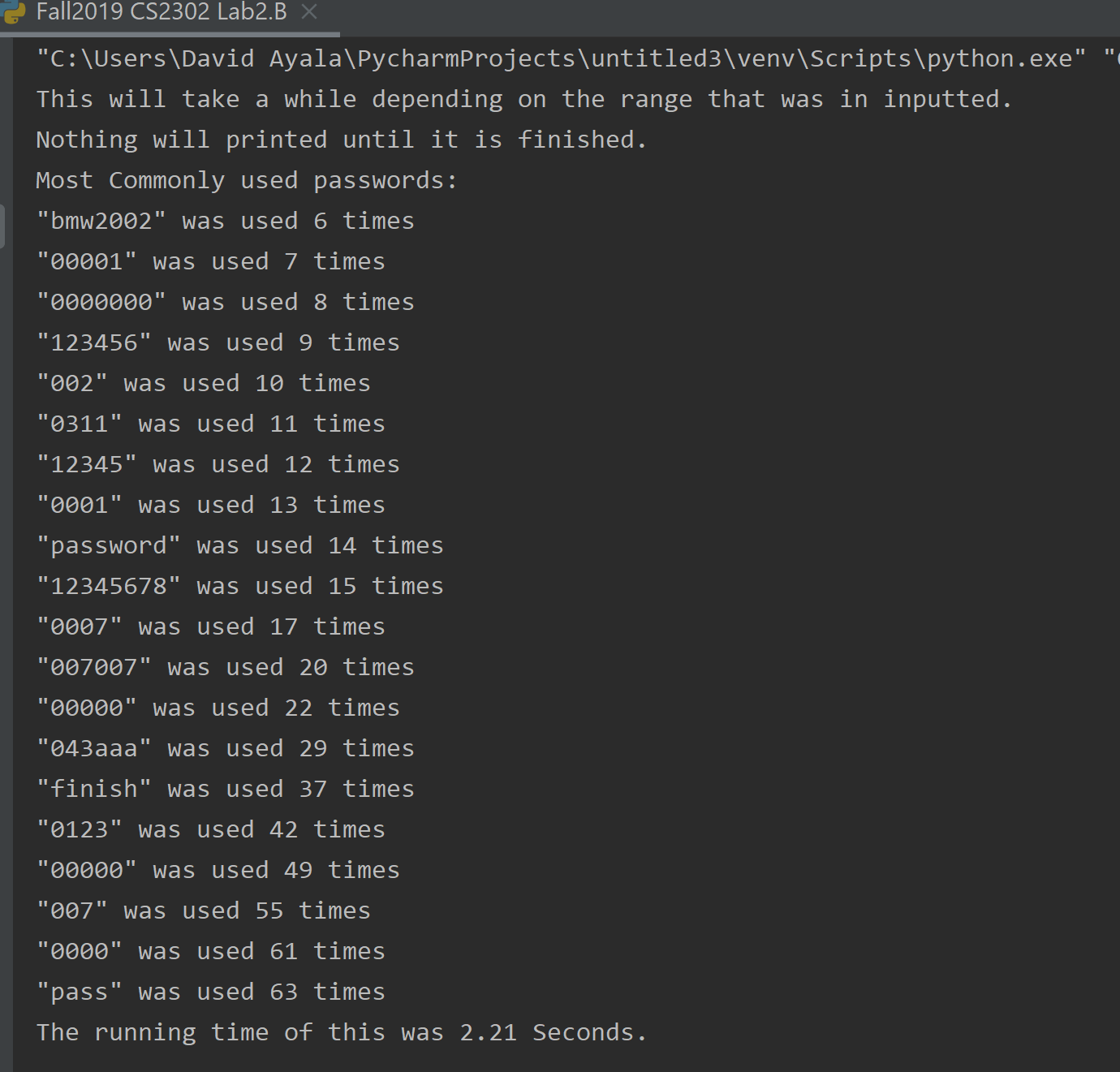
**Proposed Solution & Design Implementation**

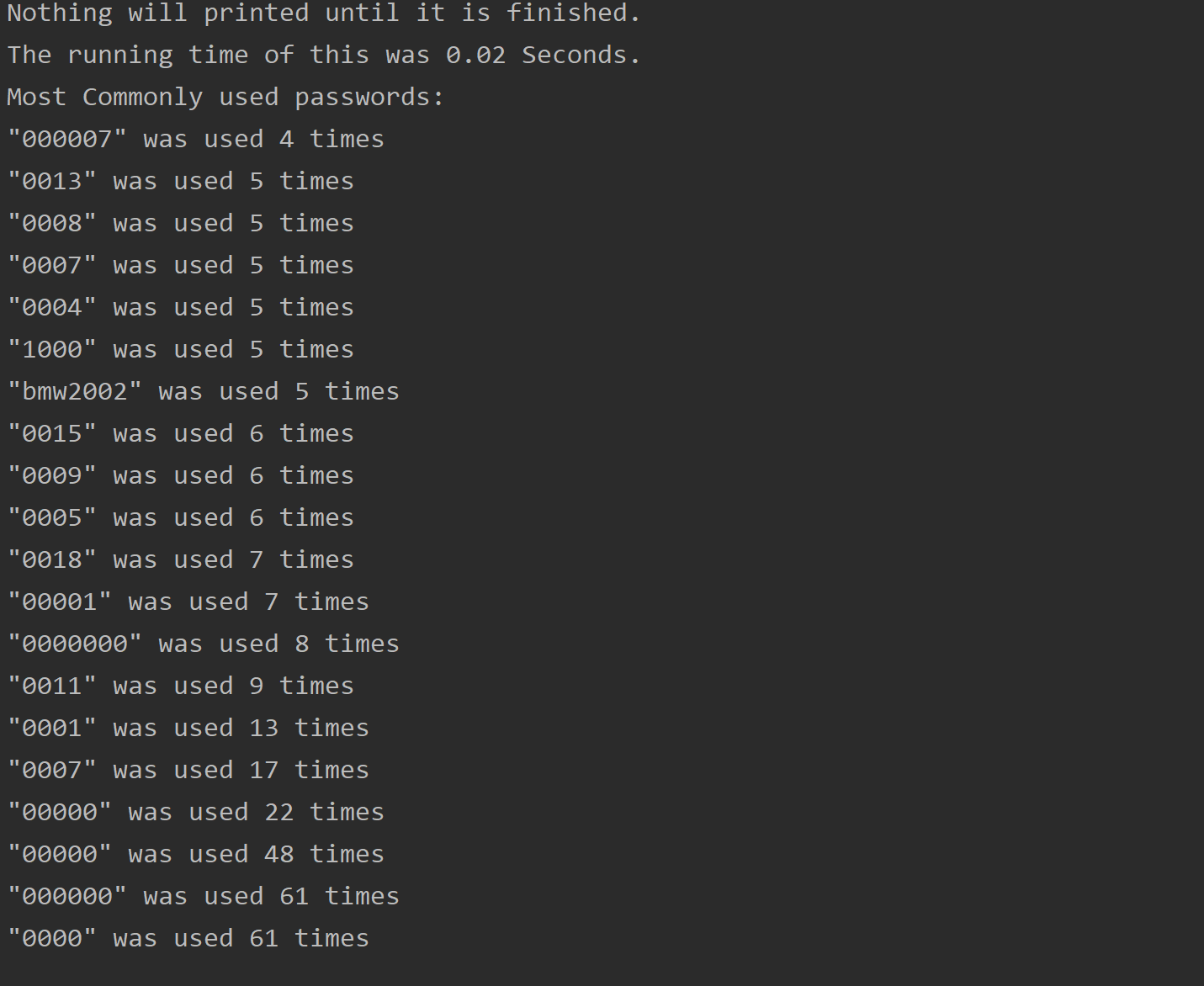
So I knew that I would need to read the file put into a list and transverse the list using a loop to find duplicate. Then implement bubble sort and honestly that was it for Part A. while part b was more difficult because I had to use a dictionary to find the duplicates. It was hard because I didn’t really understand dictionary so it took a lot of googling and youtubing to understand. Once I understood it I then had to sort using merge sort and then finally had to look for the duplicates.

**Experimental Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1000** | **10000** | **100000** | **1000000** |
| **Part A** | **0.02 Seconds** | **2.35 Seconds** | **N/A** | **N/A** |
| **Part B(dictionary)** | **0.02 Seconds** | **2.08 Seconds** | **N/A** | **N/A** |

Part A and B took too long to complete with lengths of 100000 and 1000000, putting a lot of stress on my laptop. But part B seems to be faster than part A.





**Conclusion**

This lab allowed me to learn how to implement merge sort. I also learned how to use a dictionary for the most part. I also saw that bubble sort takes longer than merge sort. And that people aren’t really great at making passwords so yeah.

**Appendix**

# Course:CS 2302 MW 1:30-2:50, Author:David Ayala

# Assignment:Lab #2B, Instructor: Diego Aguirre

# Teaching Assistant: ????, Date of last Modification: 9/27/2019

# Purpose of program:write a Python 3 program that finds the 20 most used password.

#Solution A

import time

import collections

# I put some German in this lab, since I'm currently taking it and

# I'm trying to get better at it.

# i is pronounced ee in German.

# j is pronounced yott in German.

# k is pronounced kah in German.

# a is pronounced ah in German.

# b is pronounced beh in German.

# c is pronounced tseh in German.

# password is Passwort in German.

# duplicates is Duplikate in German.

# duplicate is Duplikat in German.

# yield is Ergebnis in German.

# commonly used is häufig verwendet in German.

# number is Nummer in German.

# part is Teil in German.

# identification is Identifizierung in German.

# cleanser is Reiniger in German.

# part is teil in German.

class Node(object):

def \_\_init\_\_(self):

self.DuplikatMenge = 0

self.Passworts = []

self.Duplikate = []

def Reiniger(self):

tseh = collections.Counter(self.Duplikate)

print('Most Commonly used passwords:')

ee = 0

häufigVerwendet = []

for Identifizierung, Ergebnis in tseh.most\_common(20):

ee += 1

häufigVerwendet.append("\"{}\" was used {} times".format(Identifizierung.rstrip(), Ergebnis))

for ee in range(len(häufigVerwendet)):

print(häufigVerwendet[len(häufigVerwendet) - (ee + 1)])

# will find number of duplicates using a loop and if no duplicates then add to link list.

def TeilAh(self, Passwort):

for ee in range(len(Passwort)):

if Passwort[ee] in self.Passworts:

self.DuplikatMenge += 1

self.Duplikate.append(Passwort[ee])

else:

self.Passworts.append(Passwort[ee])

with open('10-million-combos.txt', 'r', encoding='utf-8', errors='ignore') as text:

motor = Node()

startTime = time.time()

print('This will take a while depending on the range that was in inputted.')

print('Nothing will printed until it is finished.')

for ee in range(1000): # Change range to any int that is that is not 10 million or more

motor.TeilAh(text.readline().split('\t'))

print('The running time of this was {0:.2f} Seconds.'.format(time.time() - startTime))

motor.Reiniger()

# Course:CS 2302 MW 1:30-2:50, Author:David Ayala

# Assignment:Lab #2B, Instructor: Diego Aguirre

# Teaching Assistant: ????, Date of last Modification: 9/27/2019

# Purpose of program: write a Python 3 program that finds the 20 most used password.

# Solution B

import time

from collections import OrderedDict

# I put some German in this lab, since I'm currently taking it and

# I'm trying to get better at it.

# i is pronounced ee in German.

# j is pronounced yott in German.

# k is pronounced kah in German.

# a is pronounced ah in German.

# b is pronounced beh in German.

# c is pronounced tseh in German.

# password is Passwort in German.

# duplicates is Duplikate in German.

# duplicate amount is Duplikat Mengein German.

# yield is Ergebnis in German.

# commonly used is häufig verwendet in German.

# number is Nummer in German.

# part is Teil in German.

# identification is Identifizierung in German.

# cleanser is Reiniger in German.

# dictionary is wörterbuch in German.

# part is teil in German.

# left is links in German.

# right is richtig in German.

# middle is Mitte in German.

class Node(object):

def \_\_init\_\_(self):

self.duplikatMenge = 0

self.wörterbuchDuplikate= {}

self.Passworts = []

self.Duplikate = []

def behErgebnisse(self, Ergebnisse):

array = []

ee = 0

for Ergebnis in Ergebnisse:

ee += 1

if ee > 20:

break

for Identifizierung, Nummer in self.wörterbuchDuplikate.items():

if Ergebnis == Nummer:

array.append('\"{}\" was used {} times'.format(Identifizierung.rstrip(), Nummer))

break

for ee in range(len(array)):

print(array[len(array) - (ee + 1)])

# will find the number of duplicates using a dictionary

def teilBeh(self, Passwort):

for ee in range(len(Passwort)):

if Passwort[ee] in self.Passworts:

self.duplikatMenge += 1

self.Duplikate.append(Passwort[ee])

else:

self.Passworts.append(Passwort[ee])

def Reiniger(self):

if self.Duplikate is None:

return None

else:

for Artikel in self.Duplikate:

if Artikel in self.wörterbuchDuplikate:

self.wörterbuchDuplikate[Artikel] = self.wörterbuchDuplikate[Artikel] + 1

else:

self.wörterbuchDuplikate[Artikel] = 1

array = []

for Artikel in self.wörterbuchDuplikate:

array.append(self.wörterbuchDuplikate[Artikel])

return array

def mergeSort(list):

if len(list) > 1:

Mitte = len(list) // 2

links = list[:Mitte]

richtig = list[Mitte:]

mergeSort(links)

mergeSort(richtig)

ee = 0

yott = 0

kah = 0

while ee < len(links) and yott < len(richtig):

if links[ee] < richtig[yott]:

list[kah] = links[ee]

ee = ee + 1

else:

list[kah] = richtig[yott]

yott = yott + 1

kah = kah + 1

while ee < len(links):

list[kah] = links[ee]

ee = ee + 1

kah = kah + 1

while yott < len(richtig):

list[kah] = richtig[yott]

yott = yott + 1

kah = kah + 1

return list

with open('10-million-combos.txt', 'r', encoding='utf-8', errors='ignore') as txt:

motor = Node()

print('This will take a while depending on the range that was in inputted.')

print('Nothing will printed until it is finished.')

startTime = time.time()

for ee in range(10000): # Change range to any int that is that is not 10 million or more

motor.teilBeh(txt.readline().split('\t'))

results\_to\_merge = motor.Reiniger()

Ergebnisse = list(OrderedDict.fromkeys(mergeSort(results\_to\_merge)))

correct\_results = []

for ee in range(len(Ergebnisse)):

correct\_results.append(Ergebnisse[len(Ergebnisse) - (ee + 1)])

print('Most Commonly used passwords:')

motor.behErgebnisse(correct\_results)

print('The running time of this was {0:.2f} Seconds.'.format(time.time() - startTime))