CS 2302 Lab 2-Option A Lab Report

Introduction

For this lab, I was tasked with finding the solution of duplicate IDs that happens when two companies merge. The possible IDs combinations ranged from 0 to 6000 inclusive. I had to use linked list to implement solutions to find the duplicates.

Proposed Solution Design and Implementation

To implement the methods described, I read the file and made a linked list with all the IDs from the both files I then applied the methods in the program to find duplicates.

Experimental Results

The Big O running times for the methods are:

bubble\_sort: O(n^2)

compare\_every\_element: O(n^2)

seen\_before: O(n)

Conclusions

I could not figure out how to implement the merge sort method in my program, other than that, the seen before is the one that ran the best in a worst-case scenario.

Appendix

# -\*- coding: utf-8 -\*-

########################

#Course: CS2302

#Author: Cezar Barba

#Assignment: Lab 2 Option A

#Instructor: Diego Aguirre TA: Manoj Saha

#Date of Last Modification: 12/06/2018

#The purpose of this program is to implement a solution to

#find duplicate IDs in the files activision.txt and vivendi.txt

########################

class Node(object):

def \_\_init\_\_(self, item, next):

self.item = item

self.next = next

#reads from two files and creates an unordered list

def employee\_list():

employees=None

f2= open("activision.txt","r")

#f2= open("activision\_test.txt","r")

activision = []

for x in f2:

activision.append(f2.readline())

for i in range(len(activision)):

employees = Node(int(activision[i]), employees)

f1= open("vivendi.txt","r")

#f1= open("vivendi\_test.txt","r")

vivendi = []

for x in f1:

vivendi.append(f1.readline())

for i in range(len(vivendi)):

employees = Node(int(vivendi[i]), employees)

return employees

#uses nestedloops to compare an element in the list to the other elements in the list

#to find duplicates.

def compare\_every\_element(employees):

duplicate\_list = []

temp1=employees

while temp1 != None:

temp2= temp1.next

while temp2 != None:

if temp1.item == temp2.item:

duplicate\_list.append(temp1.item)

temp2= temp2.next

temp1= temp1.next

print("These are the IDs with duplicates:")

for i in duplicate\_list:

print(i)

#performs bubble sort to a linked list and prints the ordered list with duplicates being next to each other.

def bubble\_sort(employees):

temp = employees

swap\_occurrence = "t"

while swap\_occurrence == "t":

temp = employees

swap\_occurrence == "f"

while temp.next.item != -1:

if temp.item > temp.next.item:

swap\_occurrence == "t"

num\_switch = temp.item

temp.item = temp.next.item

temp.next.item = num\_switch

temp = temp.next

temp = employees

while temp.next.item != 1:

print(temp.item)

temp = temp.next

return

#Initiales an array with boolean values to check if an ID is duplicate by iterating through

#the whole unordered list. appends to another array the IDs that were duplicates and prints them.

def seen\_before(employees):

m=6000

seen= [False]\* (m+1)

temp = employees

duplicates=[]

while temp != None:

if seen[temp.item] == True:

duplicates.append(temp.item)

else:

seen[temp.item]=True

temp = temp.next

print("These are the IDs with Duplicates")

for i in range(len(duplicates)):

print(duplicates[i])

#main method for testing the other methods

def main():

employees = employee\_list()

compare\_every\_element(employees)

bubble\_sort(employees)

seen\_before(employees)

main()

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.