Lab 2 Report

Problem:

In this lab we were tasked to create 4 sorting algorithms for a linked list. Once ordered we had to find the middle element.

Proposed solution:

We needed to use a few separate methods to find the middle element. Such as a Copy method that creates a copy of the list. element\_at method which through a counter will find a node and print its content. I also created a random number list which got a number and created a list with those many elements with random values from 1 to 101. Bubble sort was the simplest algorithm to do because we needed to compare each element with every other element. We had a Boolean expression until we were completely done looking at the whole list there is a Boolean variable that will turn true when you need to make a swap. Running time is O(n^2) Merge sort needs to split the list until one element is left. It will recursively be built back up which sorts it. Then you compare each element to the other in two separate lists and make a new list. The running time is O(nlogn). Quicksort does something similar the only difference is the pivot we use. The running time is O(nlogn).We look for a pivot and separate by larger and smaller numbers to create the separate lists. The last algorithm is a modification of quicksort that only checks one side of the list by looking at the length.

Conclusion:

I learned that base cases are not simple. For merge sort if the list is none I did not know what to return. I had a recurring error.

Appendix:

import random

class Node(object):

# Constructor

def \_\_init\_\_(self,item,next=None):

self.item = item

self.next = next

class List(object):

# Constructor

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

self.head = None

self.tail = None

def element\_at(L, pos):

count = 0

temp = L.head

while temp is not None:

if count == pos:

return temp.item

count +=1

temp = temp.next

print("item not found")

def IsEmpty(L):

return L.head == None

def Print(L):

# Prints list L's items in order using a loop

temp = L.head

while temp is not None:

print(temp.item, end=' ')

temp = temp.next

print() # New line

def Append(L,x):

# Inserts x at end of list L

if IsEmpty(L):

L.head = Node(x)

L.tail = L.head

else:

L.tail.next = Node(x)

L.tail = L.tail.next

def get\_length(L):

counter = 0

temp = L.head

while temp is not None:

counter += 1

temp = temp.next

return counter

def random\_number\_list(n):

L = List()

for i in range(n):

num = random.randint(1,101)

Append(L,num)

return L

def Copy(L):

temp = L.head

copied = List()

while temp is not None:

Append(copied,temp.item)

temp = temp.next

return copied

def Median(L):

C = Copy(L)

return element\_at(C,get\_length(C)//2)

def bubble\_sort(L):

count = 0

change = True

while change:

t = L.head

change = False

count += 1

while t.next is not None:

if t.item > t.next.item:

x = t.item

t.item = t.next.item

t.next.item = x

change = True

count += 1

t = t.next

Print(L)

def merge\_sort(L):

L1 = List()

L2 = List()

temp = L.head

if get\_length(temp)//2 < 1:

print(temp)

return None

if temp is None:

return 0

if get\_length(temp)//2 == 1:

return temp

else:

return merge\_sort(temp.next)

temp2 = L.head

while temp2 is not None:

if temp2.next is None:

Append(L2, temp2.item)

if get\_length(temp2)//2 > get\_length(temp2):

Append(L1,temp2.item)

else:

Append(L2, temp.item)

if get\_length(L1)>1:

L1 = Append(L1,merge\_sort(L1))

if get\_length(L2)>1:

L2 = Append(L2,merge\_sort(L2))

temp3 = L1.head

temp4 = L2.head

Sorted = List()

if temp3 is None and temp4 is None:

return Sorted

if temp3 is None:

Append(Sorted,temp2.item)

temp4 = temp4.next

if temp4 is None:

Append(Sorted, temp4.item)

temp3 = temp3.next

if temp3.item < temp4.item:

Append(Sorted, temp3.item)

temp3 = temp3.next

if temp4.item < temp3.item:

Append(Sorted,temp4.item)

temp4 = temp4.next

return Sorted

L = List()

L = random\_number\_list(3)

Print(L)

bubble\_sort(L)

print(Median(L))

print(" ")

L2 = List()

L2 = random\_number\_list(4)

Print(L2)

merge\_sort(L2)

Academic Agreement:

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.

Andres Arellanes