Assignment 2 Report

Benchmark Analysis:

I don’t understand what programs this part of the assignment wants. It say the run times for “the program” but I made 3 separate programs.

Time Complexity Analysis:

Check\_html.py has a time complexity of O(n)

Test\_lists.py has a time complexity of O(n log n)

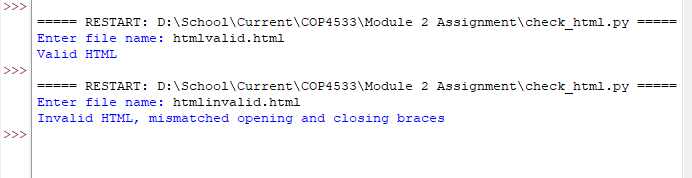
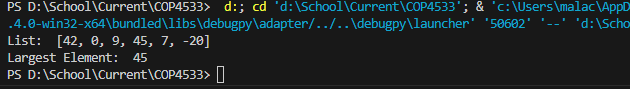
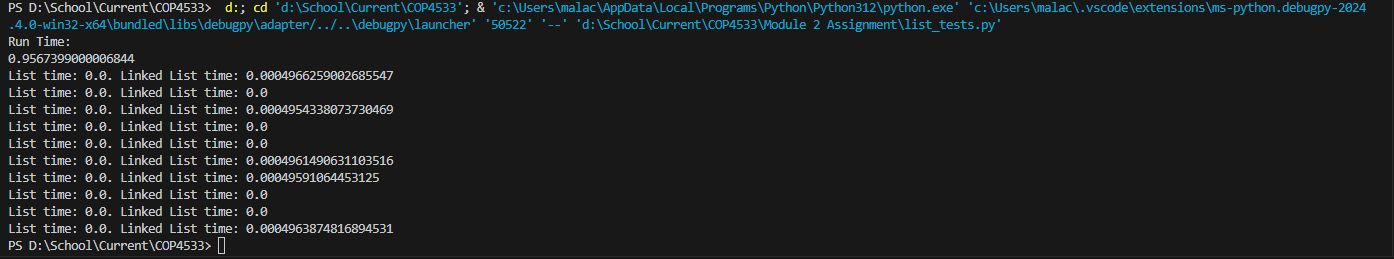
Recursion\_max.py has a time complexity of O(log n)

Discussion:

Check\_html.py has a time complexity of O(n) because there are no nested loops, just a single for loop that will iterate for the number of tags found in the file. It also does not recursively call itself in the loop.

Test\_lists.py has a time complexity of O(n log n) because it is complex containing multiple nested loops that run an uncertain number of times. There is half of the code that creates the tests and formatting, and the other half iterates through a loop hundreds of times.

Recursion\_max.py has a time complexity of O(log n) because the number of iterations is variable and will depend on the input provided. If the input is very large, the number of iterations will change, however, not at a constant rate. I figure it will either be O(n log n) or O(log n) because the number of iterations will not evenly “square” the more elements are added to the array, and I know it is not O(n log n) because there is no outer loop that runs “n” times, so it must be O(log n)



Check\_html.py

import re

#function checks html file and returns true or false, true if valid, false if invalid

def checkhtml(file):

arr = []

with open(file, 'r') as f:

#formats the html file

data = f.read().replace('\n', '')

#gets each tag using re library

tags = re.findall(r'(</?[a-z]\*>)', data)

#iterates through each tag found, searching for completeness, removing each complete pair

for tag in tags:

if tag.startswith("</"):

if len(arr) == 0:

return False

else:

arr.pop()

elif tag.startswith("<"):

arr.append(tag)

#if each pairs match has been found , len(arr) = 0, and true is returned, valid html

return len(arr) == 0

#runner code

def main():

filename = input('Enter file name: ')

if checkhtml(filename):

print('Valid HTML')

else:

print('Invalid HTML, mismatched opening and closing braces')

main()

list\_tests.py

import random

import time

import timeit

##class for regular python lists tests

class List(object):

def \_\_init\_\_(var, data):

var.data = data

var.next = None

def getdata(var):

return var.data

def getnext(var):

return var.next

def setdata(var, new\_data):

var.data = new\_data

def setnext(var, new\_next):

var.next = new\_next

##class for unordered lists tests

class UnorderedList(object):

def \_\_init\_\_(var):

var.N = 0

var.head = None

def size(var):

return var.N

def isempty(var):

return var.N == 0

def add(var, data):

var.N += 1

temp = List(data)

temp.setnext(var.head)

var.head = temp

def search(var, data):

curr = var.head

found = False

while curr and not found:

if curr.getdata() == data:

found = True

curr = curr.getnext()

return found

def remove(var, item):

curr = var.head

prev = None

while curr.getdata() != item:

prev = curr

curr = curr.getnext()

if not prev:

var.head = curr.getnext()

else:

prev.setnext(curr.getnext())

var.N -= 1

## code for outputting the program run time

def myProgram():

y = 3.1415

for x in range(100):

y = y \*\* 0.7

return y

print('Run Time: ')

print(timeit.timeit(myProgram, number=100000))

## this outputs the python list and linked list run times

for i in range(10, 10001, 1000):

list1 = list(range(i))

list2 = UnorderedList()

for j in range(i):

list2.add(j)

j = random.randrange(0, i)

start\_time1 = time.time()

list1.remove(j)

end\_time1 = time.time()

start\_time2 = time.time()

list2.remove(j)

end\_time2 = time.time()

print("List time: {0}. Linked List time: {1}".format(end\_time1-start\_time1, end\_time2-start\_time2))

recursion\_max.py

def recursion\_max(arr, i):

# if the current element is the last

if i >= len(arr) - 1:

# return the current/last

return arr[i]

else:

# get the maximum of the next the current array except the current

max\_in\_sub = recursion\_max(arr, i + 1)

# if the maximum in the is greater than the current

if max\_in\_sub > arr[i]:

return max\_in\_sub

# if the maximum in the sub array is smaller than the current element, return the non sub array

else:

return arr[i]

#runner code

arr = [42, 0, 9, 45, 7, -20]

print('List: ', arr)

print('Largest Element: ', recursion\_max(arr, 0))