Laila Romero

CS 2302 Data Structures

May 13, 2019

Lab 8

* Introduction

For this lab we were trying to discover trig functions and test their equality using random integers ranging from -pi to pi. For the second part of the lab we were trying to partition sets into two equal subsets by find the sum of the elements in the larger set and trying to equally divide it into two sets.

• Proposed Solution Design and Implementation

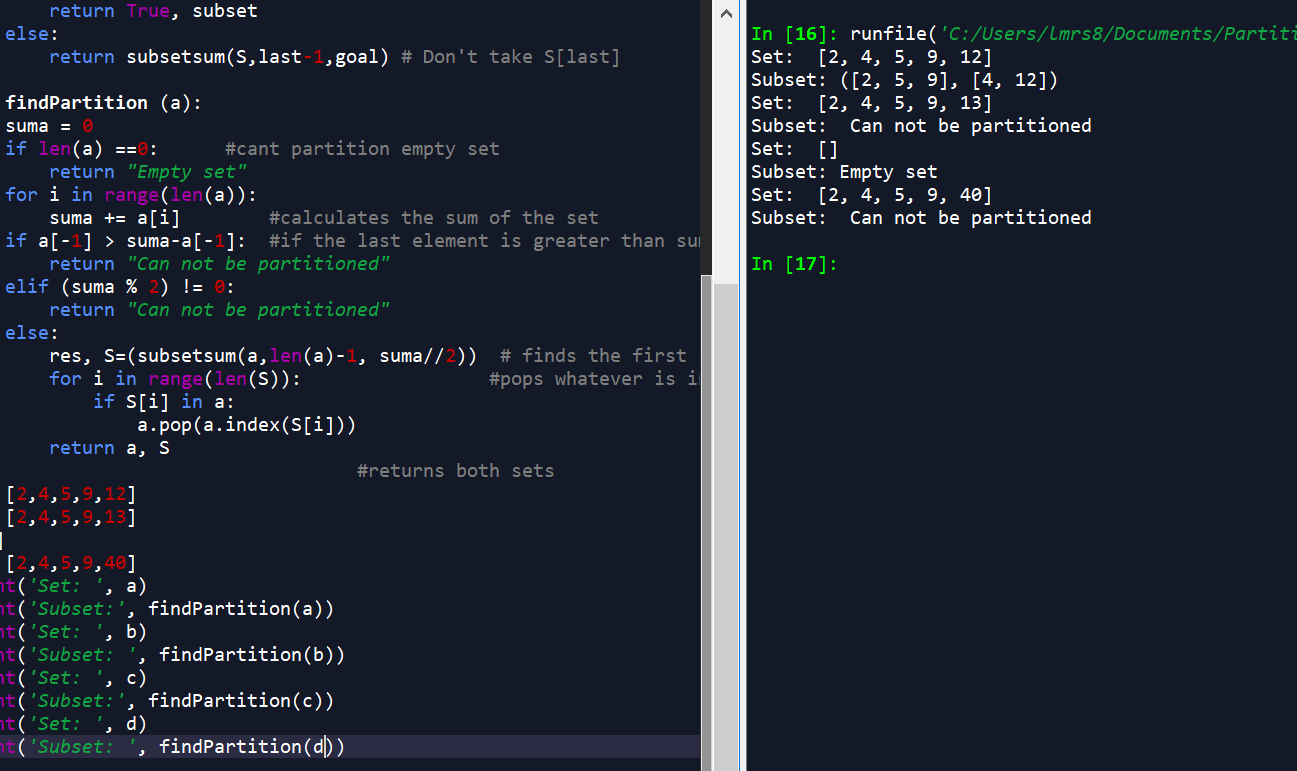
For the trig function section of the lab I used the equal method and a method I created called equaltrig which took the list of trig functions as a parameter. First, I modified the equal method by making the range of the random generated number to be from -pi to pi. Then I created the equaltrig function which consisted of multiple nested for loops which traverse through the list of trig functions and then calls for the equal method to compare each function to the other and state if their values are equal or not equal.

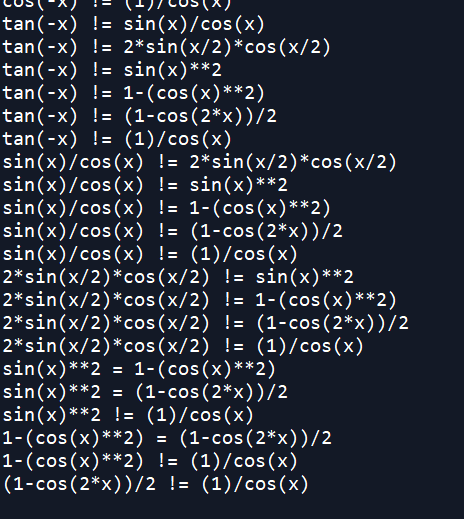
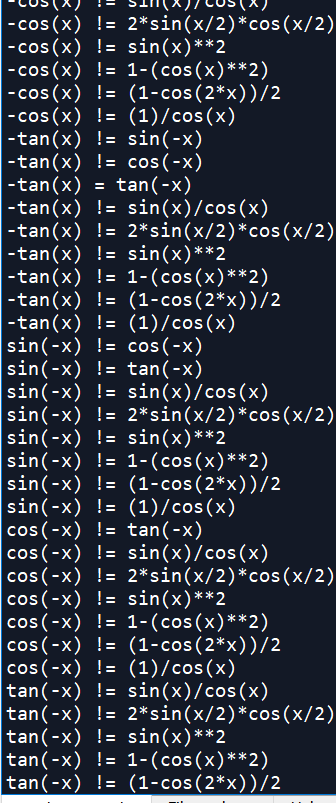
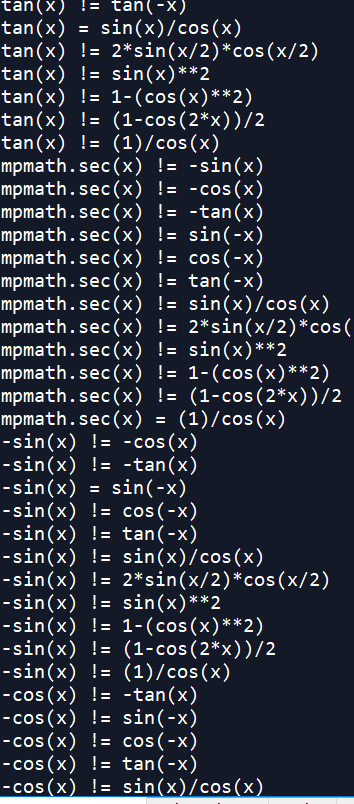
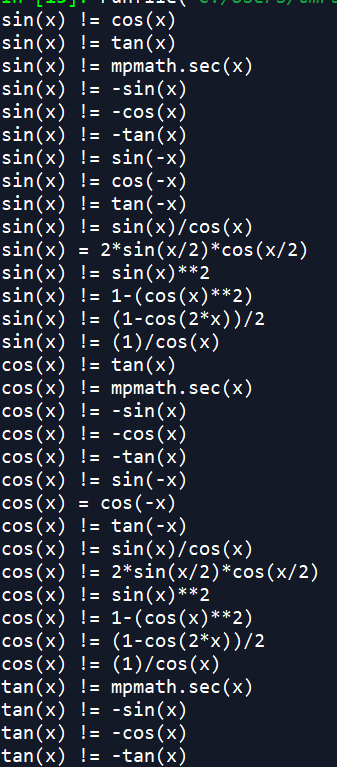
For the partitioning section of the lab I used the subsetsum method and created a method called findPartition which took the set as a parameter. For the find partition function, I created a counter that added up the values of the elements in the set and calculated the total sum. It then test to see if the set is empty and if the last element is greater than the sum of the other elements. In either of these cases the set can not be partitioned. This also goes for if the sum of the set is not even. If the set can be partitioned, then it calls for the subsetsum method of half of total sum of the set. This then returns one subset of the original set whose elements will add together to be half of the total sum. To get the other subset, I used a for loop to traverse through the subset and check which elements in the original set were in the subset. If the element was in both, it would be popped from the original set. This would then leave only the elements that were not in the subset and subsequently give us the second subset.

• Experimental results

In order to test my methods, I first tried multiple runs for the trig functions and made sure that the outputs were correct no matter the random integer created. To test the partition method I tried multiple sets that would produce every possible output, so I tested empty sets, sets with elements whose last element was larger than the sum of the other elements, sets whose sums could not be divided evenly into two and sets that could be divided evenly into two. Below I have provided the outputs of the methods.

|  |  |
| --- | --- |
| **Method** | **Big O** |
| equal | O(n) |
| equaltrig | O(n2) |
| subsetsum | O(n) |
| findPartition | O(n) |





• Conclusion

In conclusion, I learned how to test the equality of trig functions and randomly generate a number between -pi and pi. I also learned that python does not recognize the secant function so you have to import mpmath in order the read it. I learned how to partition sets into equal subsets and how to traverse through the sets to find the subsets.

• Appendix – Source codes

"""

Laila Romero

CS 2302

Professor Olac Fuentes

Lab 5

05/09/19

This code discovers the trig functions and compares them to other functions with

random integers ranging from -pi to pi.

"""

import random

import numpy as np

import mpmath

from math import \*

from math import pi

def equal(f1, f2,tries=1000,tolerance=0.0001):

for i in range(tries):

x = (2\*pi\*random.random())-pi

y1 = eval(f1)

y2 = eval(f2)

if np.abs(y1-y2)>tolerance:

return False

return True

def equaltrig(List): #Traverses through the list of trig functions

for i in range(len(List)):

for j in range(i+1, len(List)):

if equal(List[i], List[j]):

print(List[i], '=', List[j])

else:

print(List[i], '!=', List[j])

S = [2,4,5,9,12]

List = ['sin(x)','cos(x)','tan(x)','mpmath.sec(x)','-sin(x)', '-cos(x)', '-tan(x)',

'sin(-x)','cos(-x)','tan(-x)','sin(x)/cos(x)', '2\*sin(x/2)\*cos(x/2)', 'sin(x)\*\*2',

'1-(cos(x)\*\*2)', '(1-cos(2\*x))/2', '(1)/cos(x)']

equaltrig(List)

"""

Laila Romero

CS 2302

Professor Olac Fuentes

Lab 5

05/09/19

This code takes a set of integers and determines if it can be divided into two

equal subsets. The sums of the subsets must be equal.

"""

def subsetsum(S,last,goal):

if goal ==0:

return True, []

if goal<0 or last<0:

return False, []

res, subset = subsetsum(S,last-1,goal-S[last]) # Take S[last]

if res:

subset.append(S[last])

return True, subset

else:

return subsetsum(S,last-1,goal) # Don't take S[last]

def findPartition (a):

suma = 0

if len(a) ==0: #cant partition empty set

return "Empty set"

for i in range(len(a)):

suma += a[i] #calculates the sum of the set

if a[-1] > suma-a[-1]: #if the last element is greater than sum of other elements it cant be partitioned

return "Can not be partitioned"

elif (suma % 2) != 0:

return "Can not be partitioned"

else:

res, S=(subsetsum(a,len(a)-1, suma//2)) # finds the first set through subset sum

for i in range(len(S)): #pops whatever is in the subset out original set

if S[i] in a:

a.pop(a.index(S[i]))

return a, S

#returns both sets

a = [2,4,5,9,12]

b = [2,4,5,9,13]

c=[]

d = [2,4,5,9,40]

print('Set: ', a)

print('Subset:', findPartition(a))

print('Set: ', b)

print('Subset: ', findPartition(b))

print('Set: ', c)

print('Subset:', findPartition(c))

print('Set: ', d)

print('Subset: ', findPartition(d))

“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.”

