CS 2302 Data Structures

Spring 2019

MW 10:30-11:50 in CCSB 1.0202

LAB # 8

Claudio Angel Garcia

Instructor: Olac Fuentes

TA: Anindita Nath, Maliheh Zargaran

IA: Eduardo Lara

Peer leader: Erick Macik

Introduction

For this lab we are tasked to design two different programs that will have two different functionalities using randomization algorithm and backtracking algorithm. For the first problem, we need to do a program that test and detects equalities between different trigonometric expressions, this to find any trigonometric identities. For the second problem we need to do a program that will 􏰀receive a set of numbers S and the program should detect if there is any two sub sets sums that are equal 􏰀S1 = 􏰀S2 solving the partition problem using backtracking.

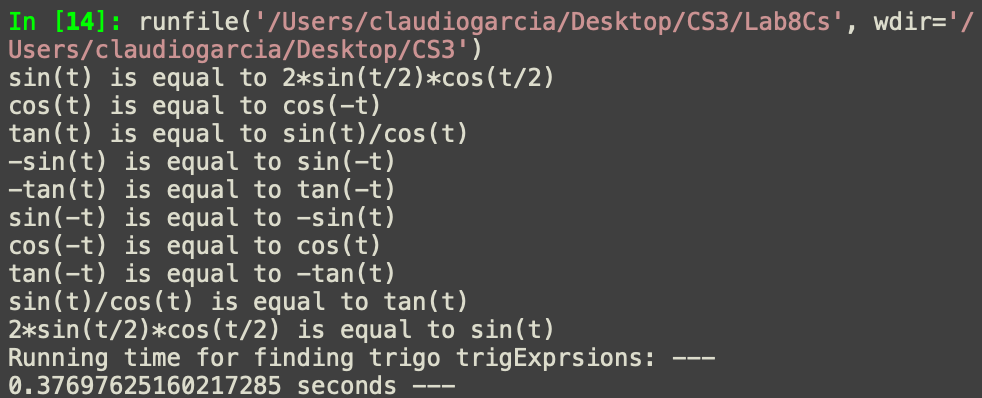
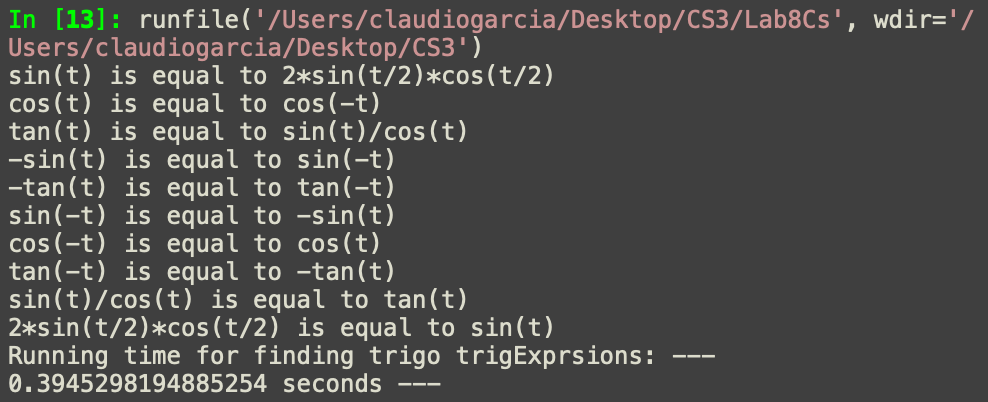
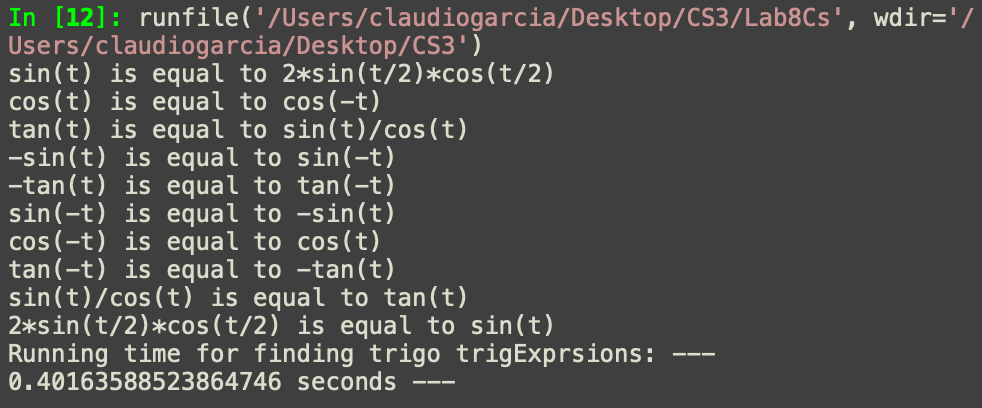
Solution

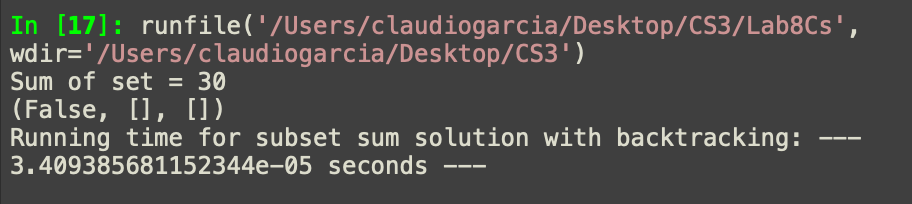
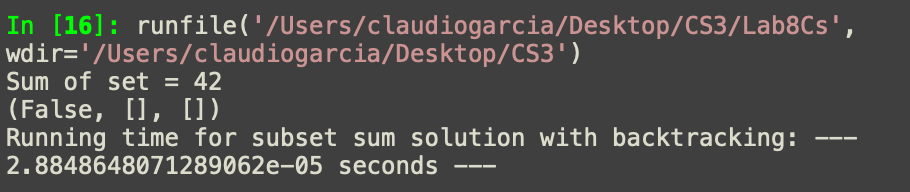
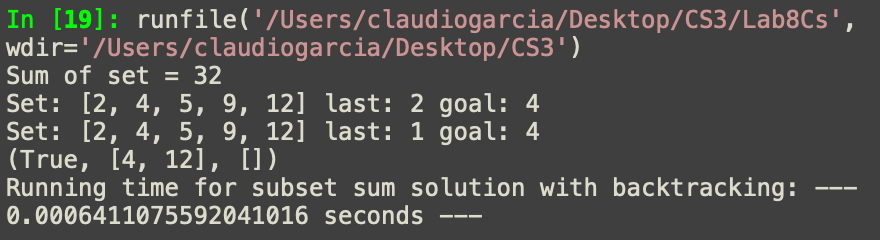
For the first method which is the randomization algorithm, first I make an array of all the given formulas that will be tested. Then I make an integer ’t’ that will contain a random number from -pi to pi. Then I created a method called isEqual to check for the equality of two trigonometric expressions. In the method I stored the expressions in two values and I used the function eval to evaluate these two formulas and see of they have the same result, if they do return True. In the method trigonometric\_expressions I just traverse through the array with two for loops to check each one with all the other ones.

For the subset sum with backtracking I just did one method that receives a set S, the last and the goal. If the goal is 0 then return True and two empty sets which means there is no subsets that have the same sum. if goal%2 != 0 or last<0 then return False and two empty sets with the recursive call subsetBacktracking(S,last-1,goal-S[last]) storing that into subset 1 and subset 2 and r then if r append the last of the set to subset 1 and return True, subset 1 and and an empty set else print the set the last and the goal and return the a recursive call with last -1.

Experimentation

Randomization



Backtracking

Conclusion

With this lab I learned more about randomization algorithms and how to implement them into finding equalities between different formulas. Also I learned about backtracking and how to use it in the implantation of the subset problem.

Appendix

import random

import numpy as np

import math

import time

t = random.uniform(-math.pi,math.pi)

def trigonometric\_exprsions(E):

for i in range(len(E)):

for j in range(len(E)):

if E[i] != E[j]: #Will skip if the same exprsions are compared

if ifEqual(E[i], E[j]): #comparing exprsions

print(E[i],'is equal to', E[j]) #if expressions give the same number

def ifEqual(f1, f2,tests=1000,tolerance=0.0001):

for i in range(tests):

x1 = eval(f1) #evaluating the exprsions in the array and storing them in x1 and x2

x2 = eval(f2)

if np.abs(x1-x2)>tolerance:

return False

return True

trigExprsions = ['sin(t)', 'cos(t)', 'tan(t)',

'-sin(t)','-cos(t)', '-tan(t)', 'sin(-t)',

'cos(-t)', 'tan(-t)', 'sin(t)/cos(t)', '2\*sin(t/2)\*cos(t/2)',

'sin(t) \* sin(t)', '1-cos(2\*t) \* 1-cos(2\*t)', '(1-cos(2\*t)\*1-cos(2\*t))/2', '1/(cos(t))']

def subsetBacktracking(S,last,goal):

if goal == 0:

return True, [], []

if goal%2 != 0 or last<0:

return False, [], []

r, s1, s2 = subsetBacktracking(S,last-1,goal-S[last]) # Take the last of S

if r:

s1.append(S[last])

return True, s1, []

else:

print('Set:', S, 'last:', last-1, 'goal:', goal)

return subsetBacktracking(S,last-1,goal) # Don't take the last of S

#start\_time = time.time()

#trigonometric\_exprsions(trigExprsions)

#print('Running time for finding trigo trigExprsions: '"--- %s seconds ---" % (time.time() - start\_time))

#

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

g = sum(S)

print('Sum of set =', g)

start\_time = time.time()

if g%2 == 0:

print(subsetBacktracking(S,len(S)-1,g//2))

print('Running time for subset sum solution with backtracking: '"--- %s seconds ---" % (time.time() - start\_time))

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.

* Claudio Garcia