Our final lab asked of us to do two things to test our abilities on algorithms. The first asked of us to ‘detect’ trig identities through a randomization algorithm, and to use backtrack to solve the partition problem, which is to see if one set of integers can be divided into two subsets that add up to the same value when split, say the set {1,2,3,7} would result in the sets {1,2,3}, {7}.

The first task was a bit tricky at first. I began by creating a list of strings that went over all trig function given to us. (sin, cos, tan, sec, -sin, -cos, -tan, sin(-t), cos(-t), tan(-t), sin/cos, 2sin(t/2)cos(t/2), sin^2, 1-coss^2, (1-cos(2t)) / 2, 1 / cos). Out of the 16 trigonometric functions only 9 of them are identities. To verify the identities, I check to see if the functions had a 1 \*10^-12 margin of error between the functions, a difference that is so marginally small between the function that they must then be equal if the difference between them is less than that tolerance value, then I would repeat this 1000 times and had the values be between -pi and pi. If the function deemed them to be equal, I would print the functions stating they were found equal. I would compare the trig function to all others in the list and remove it after finding the equalities of that one function.

The next part asked us to solve the partition problem via backtracking. To solve this, I modified the subset sum code provided to us from the class website to take 2 lists and the last integer. I would make the original set be s1 and the set to move new items into be s2. If the sum of s2 is less than the sum of all in s1 then take the last value in s1 and add it to s2. If it was greater then don’t take the last value taken and move on. That’s the basic idea, later on while testing I found more issues with the code that I will be going over in the testing portion. If the sum of both sets is equal then you would stop and print the resulting sets, if the program goes through the entire list and no such set exists then you stop and tell the user their set cannot be partitioned.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated When testing the first part. I initially ran into the issue of not getting enough of the equalities only getting about 6 or so of the equalities. Initially I would remove both functions in the equalities from the list which would give me both runtime errors and errors of the identities such as giving me incorrect identities. Once I found my issue, I did change the program to only remove one function at a time to actually go through the entire list and compare all possible functions to see what is a proper identity which resulted in the image on the right.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA picture containing object

Description automatically generatedWhile testing the second I got many runtime errors. Most of these testing errors were in testing cases when I first believe I finished. The Main error I would receive was an error stating index out of bounds, which took a while to trace down the main issue but I found it was in returning to previous calls the last would be a higher number than the indices with the current list leading to a program crash. This would only occur in few sets such as the set {2, 4, 5, 6, 8}, how ever after adding an if statement that checked if the length was valid then the program would function properly in all cases. (error is left most image, after adding if statement is middle, current test cases output bottom most). That was my main issue with the partition problem of the algorithms, which in the end gave me little trouble.

A screenshot of a cell phone

Description automatically generatedWhile checking runtimes the first part had an average runtime of .2 seconds. The second part through all runtimes tested of increasing set sizes gave me an output of 0.0 sec using the set {1, 2, 3} as a minimum set and continuing up to sets of length 20.

Appendix--------------

"""

Course 2302(Data Structures)

Instructor: Olac Fuentes

Teaching Assistant: Anithdita Nath, Mali Zargaran

Lab 8

Last Edited on 5/9/2019

@author: Seth

"""

import random

import numpy as np

import time

import math

import mpmath

from math import \*

def equal(f1, f2,tries=1000,tolerance=0.000000000001):

for i in range(tries):

x = random.uniform(-math.pi, math.pi)#range -pi to pi

if(f2 == 'sec(x)'):

f2 = 'mpmath.sec(x)'

if(f1 == 'sec(x)'):

f1 = 'mpmath.sec(x)'

y1 = eval(f1)

y2 = eval(f2)

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

return False

return True

def sumSet(S):

if S == None or len(S) == 0:

return 0

c = 0

for i in S:

c+=i

return c

def subsetsum(s1, s2,last):

# print('set 1',s1)

# print('set 2',s2)

if sumSet(s1) == sumSet(s2):#found equal subsets

return True, s1, s2

if sumSet(s1)<sumSet(s2):

if last < 0:#has gone through whole set with no possible subsets

return False, s1, s2

s1.append(s2[-1])#if there is more set to go through continue

s2.remove(s2[-1])

return False, s1, s2

if last > len(s1) - 1:#in case no sub sets exist and returns with sets that no longer work with this portion of the code

return False, s1, s2

s2.append(s1[last])#adds s1's next element to s2 and removes it from s1

s1.remove(s1[last])

result, s1,s2 = subsetsum(s1,s2,last-1)

if result:#found sub set

return True, s1,s2

else:

return subsetsum(s1,s2,last-1) # Don't take S1[last]

FUNctions = ['sin(x)', 'cos(x)', 'tan(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)\*sin(x)', '1 - (cos(x)\*cos(x))', '(1-cos(2\*x)) / 2','sec(x)', '1/cos(x)']

Found = 0

print(len(FUNctions), ' functions')

start = time.time()

while len(FUNctions) != 0:#runs until all identities are found

f1 = FUNctions[0]#first function to test

FUNctions.remove(f1)#removes f1 from list to choose a non f1 function to compare

for i in range(len(FUNctions)):

f2 = FUNctions[i]

if equal(f1, f2):

print(f1, ' = ', f2)

Found += 1

end = time.time()

print(Found, ' identities found')

print(end-start)

Set1 = [1,2,3]

Set2 = [1, 3,5,2,8, 12,14,32,46, 20, 10, 16, 18, 19, 21, 34, 30, 42, 22, 26]

start = time.time()

P, s1, s2 = subsetsum(Set1, [], len(Set1)-1)

end = time.time()

if P:

print('Set ', sorted(s1+s2), 'has a partition\n', s1, '\n', s2)

else:

print('No partition for', sorted(s1+s2))

##test set 1 then set 2

print(end - start)

start = time.time()

P, s1, s2 = subsetsum(Set2, [], len(Set2)-1)

end = time.time()

if P:

print('Set ', sorted(s1+s2), 'has a partition\n', s1, '\n', s2)

else:

print('No partition for', sorted(s1+s2))

print(end - start)

“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.”

- Seth Abel Flores