**Introduction**

This final assignment focuses on the use backtracking and randomization on producing answers algorithmically to problems that are done usually in O(2^n) time. The first objective of the assignment asks that a function be made to check whether two trigonometric identities form the given list are equal or not. The next objective requires a function to be made that checks if a partition exists from a subset, then if it does exist, print the existing partition, otherwise printing that there is not partition.

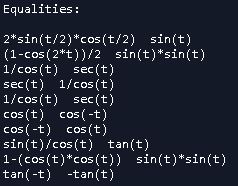
**Proposed Solution Design and Implementation**

The part of the assignment dealing with equalities uses two functions: one function named “DiscoverIdentities” and another named “Equal”. The function “DiscoverIdentities” takes a list of strings named “TrigID” (The given list of trigonometric identities) as its arguments, and creates a new list called “equal”. This new “equal” list will store two identities in one row if those two identities are equal to one another. The next step of this function selects two random trigonometric identities from the list “TrigID”. There is an if statement placed that checks if the two random identities selected are not the same identitiy chosen from the same inded in the list. Inside this if statement’s parameters follows another function named “Equal”. This second function takes the two chosen identities and determines if they are equal or not. The “Equal” function takes four arguments: The two trigonometric identities, the number of tries, and a tolerance. Within this funciton, a random range from -pi to pi range is given as “t”, since “t” is the variable given within each identity. The function then proceeds to evaluate each function with the given “t” value, and compares the difference between both functions to “tolerance”. If at any point the difference between both identities is greater than the tolerance, it is safe to assume that these two identities are not equal and the function returns False, however, if the for-loop exceeds the number of “tries”, and the difference of the identities never exceeded the tolerance, these two identities can be safely regarded as equalities, thus the function returns True. When this “Equal” function returns True, the “DiscoverIdentities” function appends the two identities two the “equal” list, else if they are not equal, the function repeats until the number of tries is equal to zero. When the loop is done, the function returns the list “equal”which is later printed for the user to see all the trigonometric identities.

The final part of the assignment regarding the backtracking algorithm is done using two global variables, these two variables are lists S1 and S2 which make up the partition of S. To check if a partition exists, simply take the total sum of the set S and divide by 2, if two subsets can be created to equal the quotient (since their sum totals will be the same) and all the numbers of set S are stored in either subsets S1 or S2, then a partition exists, otherwise no partition exists. The function “SubsetSum” always stores the values of set S into either S1 or S2, regardless of a partition existing. This is done since if the function returns False, there is no partition and a print statement prints “No partition exists” regardless of what is stored in the lists S1 and S2. However, if a partition does exist, then simply printing the lists from the global variable executes perfectly. This can be done with an if statement that holds the function “SubsetSum” and the sum of the subset lenghts equal to the original set, as its parameters. This ensures that if “SubsetSum” returns False (The sums are not equal) or if the lengths are not equal, then the print statement “No partition exists” executes, otherwise the subsets are printed as a partition exists.

**Experimental Results**

**Equalities (Randomization):**



**Partition (Backtracking):**

1. S = [2,4,5,9,12]:



1. S = [2,4,5,9,13]:



**Conclusions**

In conclusion, this particular assignment felt to be the easiest. There was ample time to complete it, as there was ample time to explain each part of the source code and its functions. Having completed this assignment, the idea of randomization and backtracking feel almost second nature, as it is understood how they work and how to manipulate them. The only thing to change and perfect if second chances were possible, would be to change the function “SubsetSum” to not have global variables, and instead use two lists entered as its arguments.

**Appendix**

"""

Course: CS 2302

Author: Jacob Montenegro

Lab: 8

Instructor: Dr. Olac Fuentes

T.A.: Anindita Nath, Maliheh Zargaran

Date of Last Modification: 5/8/2019

Program's Purpose: Certain algorithms attribute to feasible solutions through

Randomization and Backtracking, this assignemnt ensures that

different algorithms are understood. Produce a list of

trigonometric equalities to practice with

Randomizatiion. Find a partition (if possible) using

Backtracking and show whether a partition exists or not.

"""

import random

from math import \*

import numpy as np

from mpmath import \*

#This function is used to append equal identities to a list

def DiscoverIdentities(TrigID):

equal = []

tries = 100

while tries > 0:

F1 = random.randint(0,len(TrigID)-1)

F2 = random.randint(0,len(TrigID)-1)

if TrigID[F1] != TrigID[F2] and Equal(TrigID[F1],TrigID[F2]):

equal.append([TrigID[F1],TrigID[F2]])

tries -= 1

return equal

#This function is used to determine if two functions are equal

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

for i in range(tries):

#selects a random number from the pi to -pi range

t = random.uniform(-pi, pi)

y1 = eval(f1)

y2 = eval(f2)

#This function is boolean function that returns True if the functions are equal, False otherwise.

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

return False

return True

#This list contains all the trigonometric identities

trigID = ['sin(t)','cos(t)','tan(t)','sec(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(t)\*cos(t))','(1-cos(2\*t))/2','1/cos(t)']

#Returns the list of equal identities

e = DiscoverIdentities(trigID)

#prints the list of equal identities

print('\n\nEqualities:\n')

for i in range(len(e)):

for j in range(len(e[i])):

print(e[i][j],end=' ')

print()

#returns True if the sums of the two lists are equal, False otherwise

def Partition(S1,S2):

return sum(S1) == sum(S2)

#This function traverses across the Set S (its numbers) and returns True and saves 2 sets that add up to goal

#or returns False if goal or last are negative

#All insert functions are set to 0 (First item in list) to ensure the lists are sorted

def SubsetSum(S,last,goal):

#These global variables store the values of the set S, they are the partition

global s1

global s2

#Here, we are adding the first element of the list if we've found that last is equal to 0

if last == 0:

s1.insert(0,S[last])

#We return true if we've met our goal

if goal == 0:

return True

#We return False if both goal and last are negative numbers

if last < 0 or goal < 0:

return False

#Checks if the current element in the list is of use to the partition and saves it to the second list

#Not appending the integer otherwise

if SubsetSum(S,last-1,goal-S[last]):

s2.append(S[last])

return True

#inserts the current item into the first list

s1.insert(0,S[last])

#returns the function and moves to next element in list

return SubsetSum(S,last-1,goal)

print('----------------------------------------------')

#initializes the global lists

global s1

global s2

s1 = []

s2 = []

#Initializes the Set

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

#Checks if a partition exists: checks that 2 sets of numbers from the list

#add up to the length of the set divided by 2, then checks that all elements are used

#(ensuring that this is a true partition)

if SubsetSum(S,len(S)-1,sum(S)/2) and (len(s1) + len(s2)) == len(S):

#prints the 2 subsets if a partition exists

print('Partition:\n')

print(s1)

print(s2)

else:

#prints that no partition exists

print('No partition exists')

**I certify that this project is entirely my own work. I wrote, debugged, and tested the code 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. Signed, Jacob M. Montenegro.**