**Introduction**

There are two objectives for this lab which is to make a program to verify whether two equations are equivalent by using randomization and partition a set if from a set by using backtracking. Throughout, this lab, I would like to learn how to use both algorithm and take advantage of their implementation idea.

**Part-1:** write a program to discover trigonometric identities

In order to discover the equivalent trigonometric function, randomization algorithm is the best way to detect the equalities. Randomization is a algorithm which pick a random number and runs the code for certain times to check if the condition breaks or not. First, since there are several trigonometric functions, I make an array which contains every function. On randomization, we check for certain times if the difference between the two functions is less than the tolerance. If it is less, return false since they are difference, otherwise do some more tests until for loop ends. When comparing two functions, I used a “eval” which converts a string into an mathematical expression and plugs in a number for x.

**Part-2**: partition of a submit into two equivalent subsets

The idea to implements this part is to use backtracking algorithm. Inside of a “findsubset” which search for two subsets that have, we have base-cases to catch and return the appropriate value. First base case is when the goal value is 0 which means the subset is found and returns True. If g is less than 0 or the current index is less than 0, return false because there is no further need to search and returns False. Recursive calls are made two time depending on the value. The first call is to go to the next index, and subtract the current value from goal number. If the base case enters into goal==0, returns true and if return value is True, append the current value of the array into “subset” and remove that value from “S2”. If the return from the basecase is False, call a recursive call with current index-1 and do not subtract current value of array from g.

Output:

D:\UTEP\Sophmore\spring2019\CS2302\labs\Lab8>lab8.py

sin(x) & 2\*sin(x/2)\*cos(x/2) are the same.

cos(x) & cos(-x) are the same.

tan(x) & tan(x) are the same.

tan(x) & sin(x)/cos(x) are the same.

sec(x) & 1/cos(x) are the same.

-sin(x) & -sin(x) are the same.

tan(x) & sin(x)/cos(x) are the same.

sin(x)\*sin(x) & 1-cos(x)\*cos(x) are the same.

sin(x)\*sin(x) & ((1-cos(2\*x))/2) are the same.

1-cos(x)\*cos(x) & ((1-cos(2\*x))/2) are the same.

Time for randamization: 4702.795654296875

Main set: [1, 2, 4, 6, 9, 12]

Solution: [1, 4, 12] & [2, 6, 9]

Time for backtracking: 15.62451171875

Output:

D:\UTEP\Sophmore\spring2019\CS2302\labs\Lab8>lab8.py

sin(x) & 2\*sin(x/2)\*cos(x/2) are the same.

cos(x) & cos(-x) are the same.

tan(x) & tan(x) are the same.

tan(x) & sin(x)/cos(x) are the same.

sec(x) & 1/cos(x) are the same.

-sin(x) & -sin(x) are the same.

tan(x) & sin(x)/cos(x) are the same.

sin(x)\*sin(x) & 1-cos(x)\*cos(x) are the same.

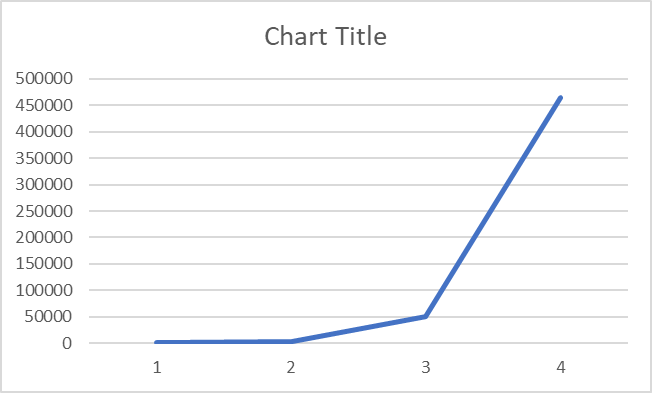
sin(x)\*sin(x) & ((1-cos(2\*x))/2) are the same.

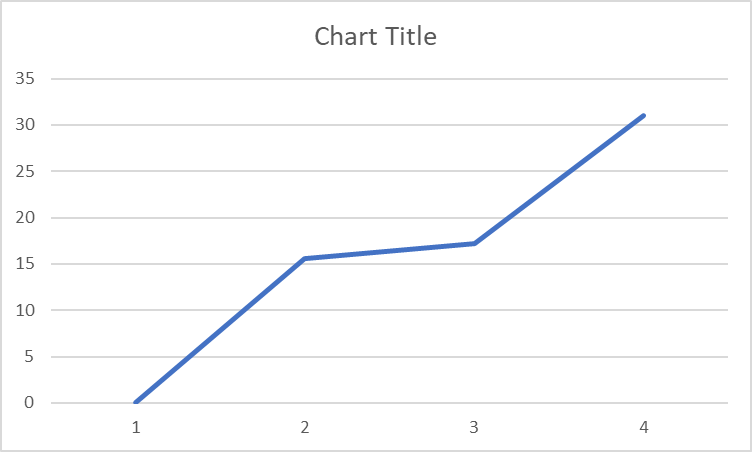
1-cos(x)\*cos(x) & ((1-cos(2\*x))/2) are the same.

Time for randamization: 3593.52001953125

There is no solution

Time for backtracking: 15.624755859375

Randomization 

Backtracking 

"""""""""""""""""""""""""""""""""""""""

Lab 8 - Algorithm Design Techniques

05/09/2019 - Ken M. Amamori

CS2302 MW 10:30 - 11:50

Professor: Olac Fuentes

TA: Anindita Nath, Maliheh Zargaran

IA: Eduardo Lara PL: Erick Macik

"""""""""""""""""""""""""""""""""""""""

import random

from mpmath import \*

import numpy as np

import time

#randamization part, tries 1000 times and verify if they are same

def equal(f1, f2,tries=1,tolerance=0.0001):

for i in range(tries):

x = random.random() #get random number

y1 = eval(f1)

y2 = eval(f2)

if np.abs(y1-y2)>tolerance: #compare if there is any difference

return False

return True

#make every possible pair and send to "equal" to verify it

def checkeq(eq):

for i in range(len(eq)):

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

if equal(eq[i], eq[j]): #test with randomization

print(eq[i], "&", eq[j], "are the same.")

#find if there is any subset that meet the criterial by backtracking

def findsubset(S, S2, last, g):

if g==0: #found

return True, []

if g<0 or last<0: #not found

return False, []

res, subset = findsubset(S, S2, last-1, g-S[last])

if res:

subset.append(S[last]) #add because it is part of the first subset

S2.remove(S[last]) #remove because it is part of the other subset

return True, subset

else:

return findsubset(S, S2, last-1, g) #do not add the last element

def partition(S):

t = sum(S)

if t%2==0:

S2=[i for i in S] #copy S

a, s = findsubset(S, S2, len(S)-1, t//2) #check if there is a subsets that are equal

if a: #found

print("\nMain set:", S)

print("Solution:", s, " & ", S2)

else: #not found

print("There is no solution")

else: #no solution

print("There is no solution")

#part 1

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

s1 = time.time()\*1000

checkeq(eq)

s2 = time.time()\*1000

print("Time for randamization:", round((s2-s1),4))

#part 2

S = [1, 2, 4, 6, 9, 12, 15, 18, 20, 30, 31, 32, 40, 45, 50, 56, 57, 90, 100, 101, 102, 103, 105, 140, 200, 230, 250, 260, 283, 302] #parent set

s3 = time.time()\*1000

partition(S)

s4 = time.time()\*1000

print("Time for backtracking:", round((s4-s3),4))

**Conclusion:**

Throughout this lab, I learned how to manage well algorithms to use backtracking and randomization. I realized that it is not difficult the implementation once I implemented by myself. Also, I felt that they are pros and cons of using them and I need to identify when to use them appropriately.

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.