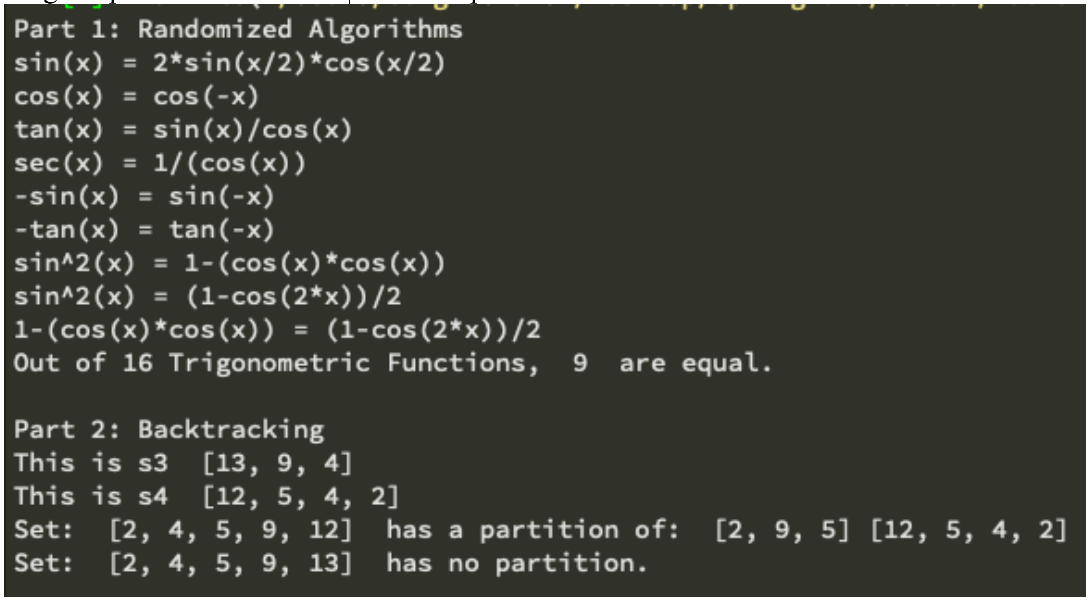
Sergio Gramer – 88521512
Lab #8: Randomization & Backtracking

In lab 8 we are tasked with creating a method that will randomly check if certain functions are equal. There are a total of 16 trigonometric functions that we compare and by default or by trigonometry rules some are equal. This method was to take these functions and compare each one to another to see which were equal and finally printing the equalities. Then, we are to take the backtracking method learned in class to receive a list of numbers. We are to split the function into 2 parts; equal or unequal; add the numbers inside and see if they are equal. We had 2 functions to use according to the lab PDF; one that was able to be partitioned and receive an equality after summing it and another that was not able to successfully be partitioned. This method was taught to us in class and given examples of.

For the first portion of this lab assignment (randomization) I used one method to send a list of the functions to as strings. I would then remove slot 0 of the list and assign it to a variable. Then inside of another for loop I would traverse the remainder of the list sending the original function first popped out of the list and the rest of the elements in order into a method provided to us by the instructor called "equal". I modified this method for it to try a certain amount of times and compare if the results were the same. After they came out "True" I would return this value (true) and back to the original method either if true print out the two functions or skip to the next one. The math operations could not be detected at times so implemented some if statements to adjust the string variable accordingly so it could be evaluated using the "eval" python function. I also set up a counter that would keep track of how many times two functions were equal so that I could print in the end how many of those were the same.
NOTE
The instructor had set up a tolerance level which I kept because removing it was giving me some problems. It seems as though some functions are the equal but during the evaluation process inside of the computer the final numbers would change just a bit so the threshold implemented by Dr. Fuentes solved that problem.

Original problem: 1000 tries | same list provided in lab:

```
Part 1: Randomized Algorithms
sin(x) = 2*sin(x/2)*cos(x/2)
cos(x) = cos(-x)
tan(x) = sin(x)/cos(x)
sec(x) = 1/(cos(x))
-sin(x) = sin(-x)
-tan(x) = tan(-x)
sin^2(x) = 1-(cos(x)*cos(x))
sin^2(x) = (1-cos(2*x))/2
1-(cos(x)*cos(x)) = (1-cos(2*x))/2
Out of 16 Trigonometric Functions,  9  are equal.

Part 2: Backtracking
This is s3  [13, 9, 4]
This is s4  [12, 5, 4, 2]
Set:  [2, 4, 5, 9, 12]  has a partition of:  [2, 9, 5] [12, 5, 4, 2]
Set:  [2, 4, 5, 9, 13]  has no partition.
```

Appendix: Source Code #Comments removed for readability

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May  8 21:44:35 2019

@author: SergioGramer
Instructor: Dr. Fuentes
TA: Anindita Nath
Purpose: The purpose of this lab is to use our randomizing skills and create
        an algorithm that takes the functions provided by Dr. Fuentes and checks
        (randomly) which are equal and which are not.
        Also, we are to take a set of numbers, and use our backtracking knowledgee
        to split the list and see if any combination of numbers in the set; when summed
        is equal.
"""

import random
import numpy as np
from math import *
import math

def randomized_alg(trig_func) :
    total = 0
    while trig_func != [] :
        comp1 = trig_func.pop(0)
        for j in range(len(trig_func)) :
            comp2 = trig_func[j]
            eq = equal(comp1, comp2)
            if eq :
                print(comp1, '=', comp2)
                total += 1
    print('Out of 16 Trigonometric Functions, ', total, ' are equal.')


def equal(f1, f2,tries=1000,tolerance=0.0001):
#    print('This is F1: ', f1)
#    print('This is F2: ', f2)
    for i in range(tries):
        x = random.uniform(-(math.pi), math.pi)
        if f1 == 'sec(x)' :
            f1 = '1/math.cos(x)'
        if f2 == 'sec(x)' :
            f2 = '1/math.cos(x)'
        if f1 == 'sin^2(x)' : #power function not defined so used math.pow
            f1 = 'math.pow(math.sin(x), 2)'
        if f2 == 'sin^2(x)' :
            f2 = 'math.pow(math.sin(x), 2)'
        y1 = eval(f1)
        y2 = eval(f2)
        if np.abs(y1-y2)>tolerance:
            return False
    return True


def sets(s1, s2, last) :
    if last < 0 :
        return False, s1, s2, s1+s2

    if sum(s1) == sum(s2) :
        return True, s1, s2, s1+s2

    if sum(s1) < sum(s2) :
        s1.append(s2[-1])
        s2.remove(s2[-1])
    if sum(s1) > sum(s2) :
        s2.append(s1[last])
        s1.remove(s1[last])
    return sets(s1, s2, last-1)

print('Part 1: Randomized Algorithms')
trig_func = ['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^2(x)', '1-(cos(x)*cos(x))', '(1-cos(2*x))/2', '1/(cos(x))']
randomized_alg(trig_func)

S0 = [2, 4, 5, 9, 12]
S2 = [2, 4, 5, 9, 13]
F2 = [2, 4, 5, 9, 13]
S3 = []
B, s1, s2, S0 = sets(S0, S3, len(S0)-1)
B2, s3, s4, S2 = sets(S2, S3, len(S2)-1)
print()
print('Part 2: Backtracking')
S0.sort()
if B :
    print('Set: ', S0, ' has a partition of: ', s1, s2)
if not B :
    print('Set: ', S0, ' has no partition.')

if B2 :
    print('Set: ', S2, ' has a partition of: ', s3, s4)
if not B2 :
    print('Set: ', F2, ' has no partition.')
```

ACADEMIC CERTIFICATION

I "Sergio Gramer" 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.

Signed: 05/13/2019

Sergio Gramer

[illegible]