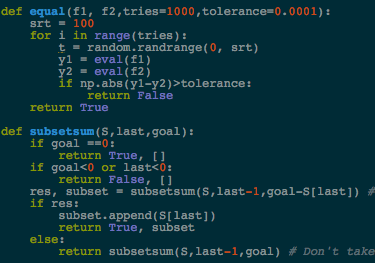
**Lab # 8 Report: Randomization and backtracking**

**Introduction:**

﻿For this lab assignment we were asked to do 2 things. First to see if the two trigonometric functions are equivalent and the second task was to see if there is an existing partition between 2 subset of numbers. The main purpose of this lab assignment is to learn how different algorithms work. For this lab assignment we worked with randomization.

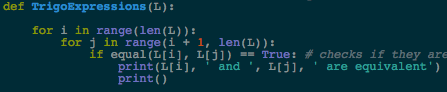
**Pre-methods:**

For pre-methods I only used few of them, the necessary ones. To do the equivalency of the trigonometric expressions I used the method *equal()* which checks if to expressions are equivalent. To get the partition of the subsets I used *subsetsum()* to get the sum of the numbers inside the subset.



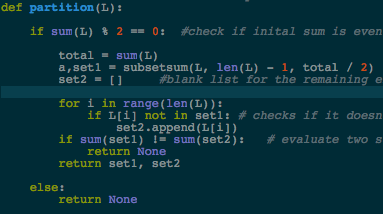
**Trigonometric Expressions:**

To know if two expressions were equivalent I created a method *TrigoExpressions()* which receives a list with all the trigonometric identities. I did two for loops to check in each identity and in each iteration it the index in the first loop is equal to the index of the second for loop. If this is true then it only prints that they are equivalent.



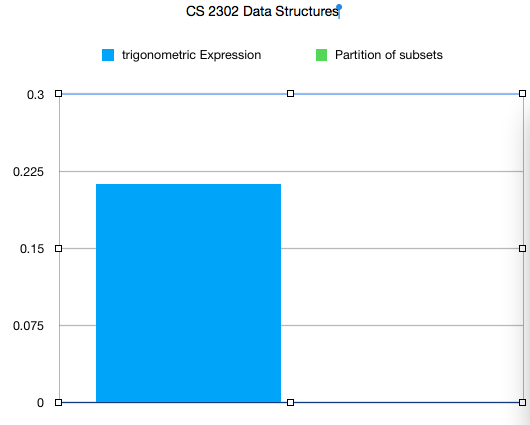
**Partitions in subsets:**

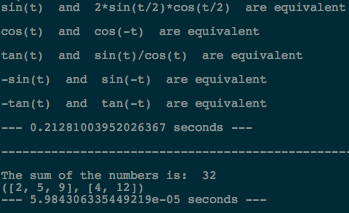
For this task I created a method partition() which receives a list with the subset and it checks if the sum of the elements list is even. If this is true it initializes two variables with the sum of the subset and it initializes another variable in where we will append the values of the other subset. In a for loop if the item in the index of the list is not in the variable created with *subsetnum()* then it is appended to the variable with the empty list. If the sum of the first set of numbers is not equal to the sum of the second set then it returns None, otherwise it returns both sets. If sum of the items in the list is not even it returns None.

****

**Conclusion:**

This lab assignment was really helpful for me because I did not understand backtracking at all but now it is more understandable. I think this lab was really good for us because it was not that difficult and also because now we know better how different algorithms work.



****

**﻿**

import random

import numpy as np

from math import \*

import time

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

srt = 100

for i in range(tries):

t = random.randrange(0, srt)

y1 = eval(f1)

y2 = eval(f2)

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

return False

return True

def subsetsum(S,last,goal):

if goal ==0:

return True, []

if goal<0 or last<0:

return False, []

res, subset = subsetsum(S,last-1,goal-S[last]) # Take S[last]

if res:

subset.append(S[last])

return True, subset

else:

return subsetsum(S,last-1,goal) # Don't take S[last]

# This method sees if the identities are equivalent to each other

def TrigoExpressions(L):

for i in range(len(L)):

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

if equal(L[i], L[j]) == True: # checks if they are equal

print(L[i], ' and ', L[j], ' are equivalent')

print()

# this method checks if there's a partition in the subsets

def partition(L):

if sum(L) % 2 == 0: #check if inital sum is even

total = sum(L)

a,set1 = subsetsum(L, len(L) - 1, total / 2) #get an inital subset

set2 = [] #blank list for the remaining elements in the original list

for i in range(len(L)):

if L[i] not in set1: # checks if it doesn't repeat

set2.append(L[i])

if sum(set1) != sum(set2): # evaluate two sums

return None

return set1, set2

else:

return None

identities = ['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))']

start1 = time.time()

TrigoExpressions(identities)

print("--- %s seconds ---" % (time.time() - start1))

print()

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

print()

# FOR SUBSET SUM

#S = [1,2,3]

S = [2,4,12,5,9] # this one shows the sets

#S = [2,6,8,12,20,30] # this shows None

g = sum(S)

print('The sum of the numbers is: ',g)

start2 = time.time()

print(partition(S))

print("--- %s seconds ---" % (time.time() - start2))

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.

* David A. Davis