**Lab 8 Report**

This lab was easier and harder than lab 7. First, I started this lab by writing down the methods and what I needed for them. After I figured a rough draft of how to implement what I needed, I moved over methods given by Dr.Fuentes to help further my code. **A screenshot of a computer

Description automatically generated**

I first focused on the first method, mine named, “RandCompare”. I tried a couple of drafts I wrote out on paper. Trying to use combinations tool to randomized the calling of any function I chose to output different functions to compare in ‘equal’ method, and even using one for loop. I would change between these two constantly and change each line of code to see which would output correctly, but neither of these two ideas would work. I would either get an error for ‘pi’, an error for “not subscripted”, etc. It wasn’t until I viewed the loop, that I got an idea. I changed my one for loop into two, viewing the list of functions I created as a 2D array, with ‘I’ and ‘j’ values to use for my ‘f1’, first function, and my ‘f2’, second function.A screenshot of a computer

Description automatically generated

When switching this around and changing the values in my list, it would only then output the correct comparisons I was creating. Some of the functions given, I wasn’t able to work out, but most, I was. I also added what was the value of the random number ‘t’ to show that this number was constantly changing and if it was the same for each function. Then, I added to print the function itself to see if the function was also changing with the ‘I’ and ‘j’ values I was feeding it with.

This **running time** is: 0.6230792999267578 seconds

Next, I focused on the second method, mine named, “solvPartition\_Back”. This method, I wasn’t able to finish. I started with creating two separate subsets from the main ‘Set’ I created. Once I did, I then set a while loop to check if the length of the ‘Set’ given did not equal to 0 or 1, since then, it couldn’t create a partition.A screenshot of a computer

Description automatically generated

I tried drawing out a draft of maybe getting a generalized value of each of the elements in the list and adding it to another and checking if either sum would equal to each other, hence my ‘Sum’ method. This try didn’t work. Next, I figured that maybe I could divide the length, make the first part my first subset, the second half my second subset, and try those if those would be a partition. I ran into more errors and eventually ran out of time, as shown now.

Thank you, TA’s, for your hard work this semester.

**Honesty Statement:**

Academic dishonesty includes but is not limited to cheating, plagiarism and collusion. Cheating may involve

copying from or providing information to another student, possessing unauthorized materials during a test, or

falsifying data (for example program outputs) in laboratory reports. Plagiarism occurs when someone

represents the work or ideas of another person as his/her own. Collusion involves collaborating with another

person to commit an academically dishonest act. Professors are required to - and will - report academic

dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

I hereby state hereby this code is mine and mine alone.

**Appendix**:

import random

import numpy as np

from math import \*

import time

def equal(f1, f2,tries=1000,tolerance=0.0001): #random algorithm to find how similar

for i in range(tries):

x = random.random()

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]

def edit\_distance(s1,s2): #gives distance between both, such as words

d = np.zeros((len(s1)+1,len(s2)+1),dtype=int)

d[0,:] = np.arange(len(s2)+1)

d[:,0] = np.arange(len(s1)+1)

for i in range(1,len(s1)+1):

for j in range(1,len(s2)+1):

if s1[i-1] ==s2[j-1]:

d[i,j] =d[i-1,j-1]

else:

n = [d[i,j-1],d[i-1,j-1],d[i-1,j]]

d[i,j] = min(n)+1

#print(d)

return d[-1,-1]

#####################

a= 'sin(t)'

b= 'cos(t)'

c= 'tan(t)'

"""

d= 'sec(t)'

"""

e= '-sin(t)'

f= '-cos(t)'

g= '-tan(t)'

h= 'sin(-t)'

i= 'cos(-t)'

j= 'tan(-t)'

k= 'sin(t)/cos(t)'

l= '(2\*sin(t/2))\*cos(t/2)'

"""

m= 'sin^2\*t'

n= '1-(cos^2)\*t'

"""

o= '1-(cos(2\*t))/2'

p= '1/cos(t)'

List=[a,b,c,e,f,g,h,i,j,k,l,o,p]

t=random.uniform(-math.pi,math.pi)

#Part 1

def RandCompare(L): #will create all combinations of variables to test each is equal

#for loop to go through characters A thru P

for i in range(len(L)):

for j in range(len(L)):#another loop for different variable

f1=L[i]#setting variable w/ 'i' value

print(f1,' t is: ',t)

f2=L[j]

print(f2,' t is: ',t)#setting variable w/ 'j' value

print('Are they equal? ',equal(f1,f2))#check if equal

start=time.time()

print(RandCompare(List))

end=time.time()

print('Running time in seconds: ',end-start)

#Part 2

Set=[1,2,3,4,5]

def Sum(S):#getting the sum

sum=0

for n in range(0,len(S)):

sum+=S[n]

return sum

def solvPartition\_Back(S):

one=set()

two=set()

#creating two subsets of S

while len(S)!=0 and len(S)!=1:#checking to only use actual partitions

i=len(S)//2

if sum[0:i]==sum[i:]:#testing if first half of set is equal to second half

one.append(range(S[0:i]))#appends values if statement is true

two.append(range(S[i:]))

return one,two

print("No partition")

"""

solvPartition\_Back(Set)

"""