**Lab 8**

The code makes use of the algorithm design techniques Randomization and Backtracking. The first method, discover(), takes a list of functions written as strings and then uses nested for loops to read the list, test and compare each to one another a thousand times with a tolerance of 0.00001 to check if they are equivalent to one another and append them to a new list that is returned as output. The second method, subsets() that first checks if the is odd, if so, then the set cannot be partitioned, if it is even, then a call is made to the subsetsum() method that returns one of the partitioned sets S1, if it is possible otherwise the method ends, if it is “True” then a list S2 is created with the remaining elements of S, if the sum of Se is equal to the sum of S1, if so then the sets are equal and there are neither intersections between them nor left over elements of S. Thus it will return the partitioned sets S1 and S2.

**Experimental Results**

|  |  |  |
| --- | --- | --- |
| Methods | Inputs | Output |
| Discover | **['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))']** |  |
| Time = 31.055963277816772 |
| Subsets | **S = [2,4,5,9,12]** |  |
| Time = 0 |
| **S = [1,2,4,5,9,12]** |  |
| Time = 0 |
| **S = [1,2,4,5,9,12,5,11,8,9,7,3,15,17]** |  |
| Time = 0 |
| **S = [2,4,5,9,12,5,11,8,9,7,3,15,17]** |  |
| Time = 0 |
| **S = [1,2,4,5,9,12,5,11,8,9,7,3,15,17,30,24,15,13,27,6,23,22,268]** |  |
| Time =0.000997304916381836 |
| **S = [1,2,4,5,9,12,5,11,8,9,7,3,15,17,30,24,15,13,27,6,23,22,268,100,138,155,47,88,97,56,64,123,220,225,174,176,168,145]** |  |
| Time = 0.000997781753540039 |

**Time Complexity**

Discover() = O(n2)

Subsetsum() = O(2n)

Subsets() = O(2n)

**Conclusion**

Randomization is the most appropriate way to check equations and any procedures they might have, like equals or results. To produce the required sets, backtracking is the most simple way to implement the operations required to make the partitioned sets.

**Source Code**

**"""**

**Created on Wed May 8 13:40:45 2019**

**CS 2302 - Data Structures**

**Instructor:Olac Fuentes**

**Lab 8, Algorithm Design Techniques**

**Makes use of the Backtracking and Randomized Algorithm techniques to find equal expressions and**

**to find if a set S can be partitioned in to two different sets that add up to the same number.**

**TA:Anindita Nath**

**@author: Hugo Chavez**

**"""**

**#from math import \***

**from mpmath import \***

**import math**

**import random**

**import numpy as np**

**import time**

**#------------------------------------------------------------------**

**def discover(exp):**

**tolerance = 0.0001**

**equalities = []**

**for i in range(len(exp)):**

**for j in range(len(exp)):**

**equal = True**

**for e in range(1000):**

**t = random.uniform(-1\*math.pi,1\*math.pi)**

**y1 = eval(exp[i])**

**y2 = eval(exp[j])**

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

**equal = False**

**if equal:**

**temp = []**

**temp.append(exp[i])**

**temp.append(exp[j])**

**equalities.append(temp)**

**return equalities**

**#----------------------------------------**

**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 subsets(S,pos,count,S1,S2):**

**if sum(S)%2 ==1:**

**return False, S1,S2**

**else:**

**a,S1 = subsetsum(S,len(S)-1,sum(S)/2)**

**if a:**

**for i in range(len(S)):**

**S2.append(S[i])**

**for j in range(len(S1)):**

**S2.remove(S1[j])**

**if sum(S2) == sum(S)/2:#b and(len(S1) + len(S2) == len(S)):**

**return True, S1, S2**

**S1 = []**

**S2 = []**

**return False,S1,S2**

**#-----------------------------------------------------------------------------**

**print('\n------------- Trigonometric Identities -------------')**

**expressions = ['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))']**

**start = time.time()**

**print('\n')**

**equals = discover(expressions)**

**print('Equal expressions:')**

**for i in range(len(equals)):**

**print('Equation ', i+1 ,':',equals[i])**

**end = time.time()**

**print(end - start)**

**print()**

**print('\n---------------- Partition Problem ----------------')**

**S = [1,2,4,5,9,12,5,11,8,9,7,3,15,17,30,24,15,13,27,6,23,22,268,100,138,155,47,88,97,56,64,123,220,225,174,176,168,145]**

**#print(sum(S))**

**S1=[]**

**S2 = []**

**start = time.time()**

**a,Sn,St = subsets(S,0,len(S)-1,S1,S2)**

**if a:**

**print('\nS', S)**

**print('\nS1 =', Sn)**

**print('\nS2 =', St)**

**else:**

**print('\nS has no partition')**

**end = time.time()**

**print(end - start)**

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.

\_\_\_\_\_\_\_\_\_\_\_\_Hugo Chavez\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_