Armando Ortiz

May 13, 2019

CS 2302 Data Structures

Lab Report 8

UTEP

Instructor: Olac Fuentes

**Introduction**

The purpose of this program is to use the Algorithm design techniques of randomization and backtracking to compare equalities. The randomization function compares trigonometric identities and prints the equalities of these identities. The backtracking algorithm partitions a list into smaller sublists with equal sums. These two algorithms were created using functions from the class website.

**Proposed Solution Design and Implementation**

For the comparison of trigonometric identities, a list is created filled with the identities as strings. This list is then sent to the equal function provided in the class website. The random algorithm function uses the equal function to compare the identities. The value of t is what makes this random, as t is a random number between pi and negative pi. The equality of these two functions are checked through a loop to confirm the equality.

The backtracking algorithm uses the subset sum function provided by the professor to create a sublist that is equal to an integer that is half the total of the larger list. This uses backtracking to create the sublist. This is only done if the sum of larger list is even, as two sublists can’t have an equal, odd sum. Another list is created only containing the items not in the other sublist. If these two lists are equal then they are printed.

**Results**

**Output**

**Trigonometric Identities**

**A close up of text on a black background

Description automatically generated**

**Partitions**

**Size: 10 Size:20**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**Size: 30**

**A screenshot of a cell phone

Description automatically generated**

**Size: 40**

**A screenshot of a cell phone

Description automatically generated**

**Runtimes**

**Trigonometric Identities**

|  |  |
| --- | --- |
| **Tries** | **Runtime** |
| **10** | ﻿0.06081 |
| **50** | ﻿0.23487 |
| **100** | ﻿0.31568 |
| **150** | ﻿0.55976 |
| **200** | ﻿0.70708 |
| **250** | ﻿0.87722 |
| **300** | ﻿1.11229 |
| **350** | ﻿1.13912 |
| **400** | ﻿1.28030 |
| **450** | ﻿1.53189 |
| **500** | ﻿1.74027 |

**Partitions**

|  |  |
| --- | --- |
| **List Size** | **Runtime** |
| **10** | ﻿0.00343 |
| **20** | ﻿0.00677 |
| **30** | ﻿0.01440 |
| **40** | ﻿0.01435 |
| **50** | ﻿0.02127 |
| **60** | ﻿0.01990 |
| **70** | ﻿0.02185 |
| **80** | ﻿0.02742 |
| **90** | ﻿0.03055 |
| **100** | ﻿0.03475 |

**Conclusion**

In this case the backtracking algorithm has a smaller runtime than the random algorithm. This could be due to the fact that the main method of testing runtime was in number of changes compared to list size, and the partition algorithm doesn’t always have to run completely due to restrictions in the way a partition can be made. These algorithms show the different ways these techniques can be used to solve problems of all types.

**Appendix**

﻿"""

Author: Armando Ortiz

Course: CS 2302 Data Structures

Assignment: Lab 8

Instructor: Olac Fuentes

T.A.: Anindita Nath, Maliheh Zargaran

Last Modification: May 13, 2019

Purpose: The purpose of this program is to use algorithm design techniques to perform two functions

- Use Random algorithms to determine the equalities of Trigonometric identities within a list

- Use Backtracking to partition two sublists with equal sums from the same list.

"""

import random

import math

from mpmath import \*

import numpy as np

import time

#Determines equality of both identities

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

for i in range(tries):

t = random.uniform(-math.pi,math.pi)#Picks a random number between -pi and pi

y1 = eval(f1)

y2 = eval(f2)

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

return False

return True

#Goes through list of Identities to test their quality

def comparisons(eq):

for i in range(len(eq)):

for j in range(len(eq)):

if i != j: #Avoid Duplicates

if equal(eq[i],eq[j]):

print(eq[i],"=",eq[j])

#Creates subset list that adds up to goal number

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]

#Takes list and creates two partitions with equal sums

def Partition(S):

total = sum(S)

S2 = S.copy()

if total % 2 == 0:#Sublist with two equal partitions not possible if total is odd

a,s = subsetsum(S,len(S)-1,total//2)

if a:

for j in range(len(s)):

S2.remove(s[j])#Remove items from subset list

if sum(S2) == sum(s):#Compare totals of list

print("Partition Exists")

print('S1:',s, '= S2:',S2)

else:

print('No Partition Exists')

else:

print('No Partition Exists')

print("Trigonometric Equalities")

eq = ['sin(t)','cos(t)','tan(t)','-sin(t)','-cos(t)','-tan(t)','sec(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()

comparisons(eq)

end = time.time()

comparisonRT = end - start

print("")

print("Comparison Runtime:",comparisonRT)

print("")

S = []

print("Equal Subsets")

start = time.time()

for i in range(0,10):

S.append(i)

print('S:',S)

Partition(S)

print("")

end = time.time()

partitionRT = end - start

print("Partition Runtime:",partitionRT)

A screenshot of a cell phone

Description automatically generated