CS2302 - Data Structures

Spring 2019

Lab No.8 Report

Introduction

This lab will cover the use of different algorithm design techniques in order to solve given problems. Making principal use of backtracking and randomization algorithms.

Design & Implementation

Trigonometric Identities:

This problem gives us different trigonometric functions and we must find a way in order to compare them and find an identity between them. For this problem we will perform a randomization algorithm, were a random value between -pi and pi is chosen and then used to compare the different functions.

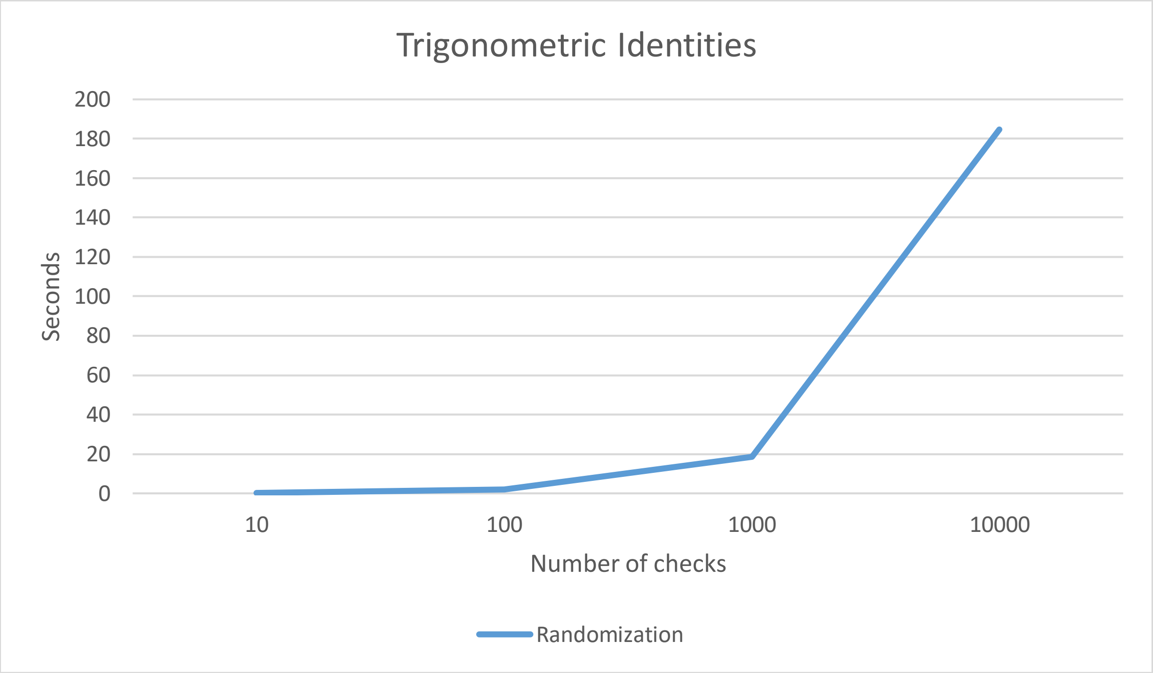
Using the mpmath library to perform trigonometric functions. Our method will first create an array of all trigonometric functions that will be checked. Then this array will be used in comparisons, taking trigonometric functions from the array in order and making certain number of comparisons with each other function. When one function passes all comparisons then an identity has been found.

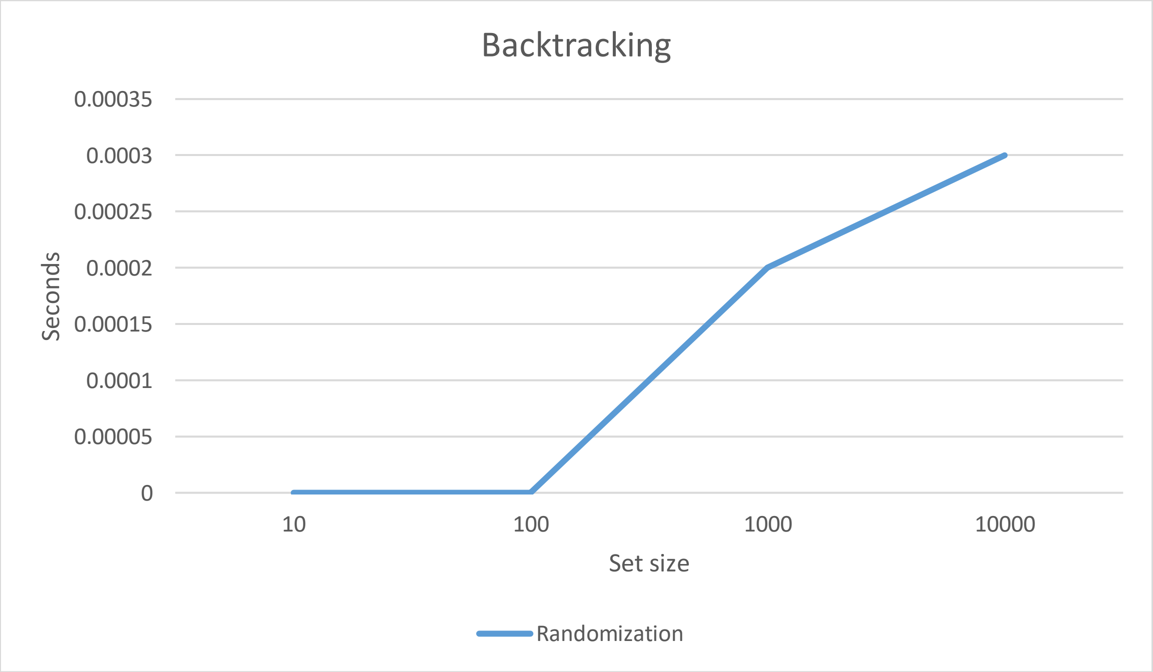
Partition Problem:

The problem tells us to divide a given set into two subsets that have the same sum of their elements, providing us the subsets or an output indicating that no such partition exists. For this problem we will perform a backtracking algorithm.

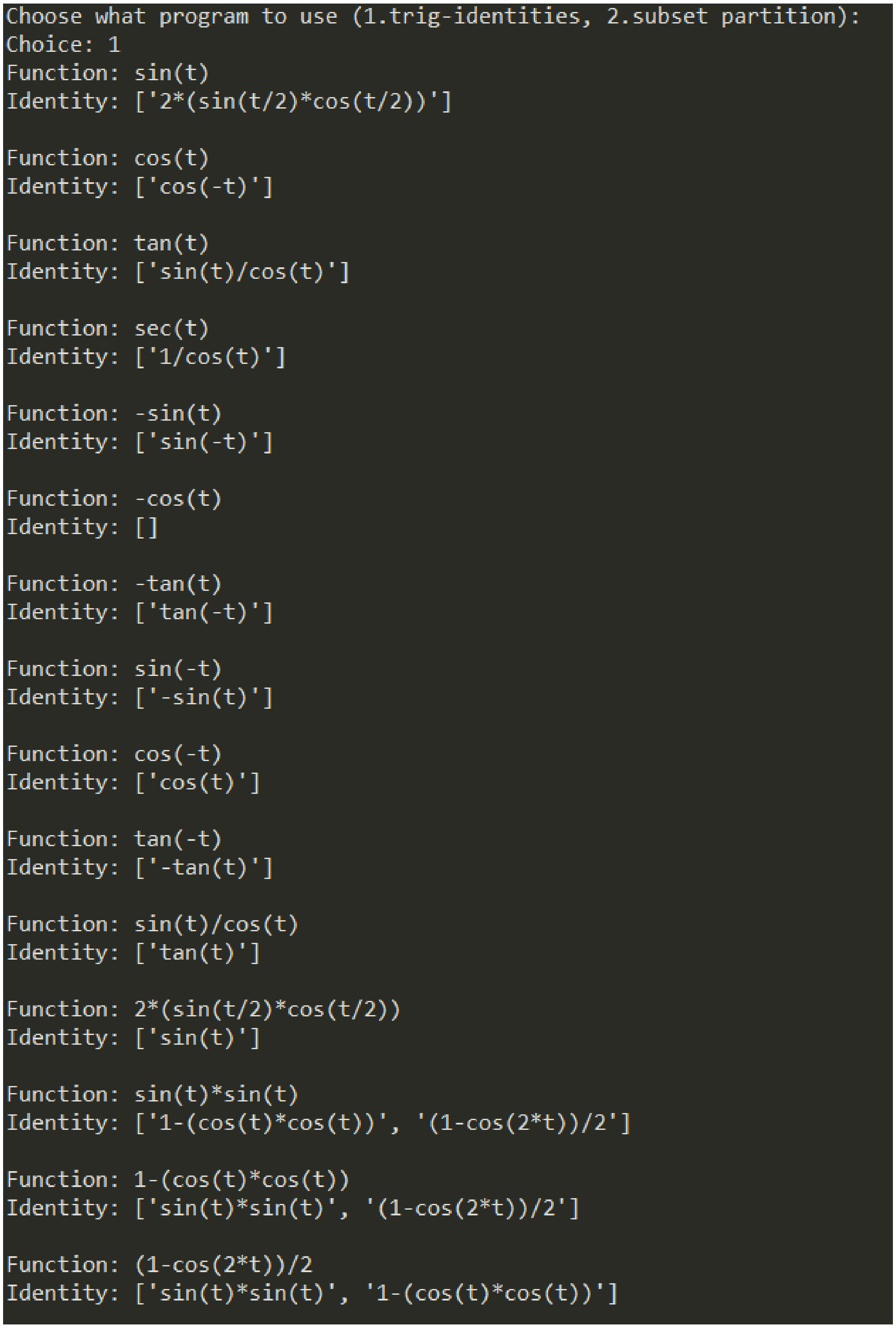
We will first determine if the set can be partitioned by summing up the values in it and determining if the sum can be divided in 2, returning the value of the half of this sum. Another method will then be called, receiving the set to be partitioned, the partition goal and the last element in the set. The method’s first base case is when the goal value is 0 which means the subset is found and another base case for when goal is lees than 0 meaning there is no further search need returning False. Recursive calls are then made, adding in a way the numbers in the array, subtracting from the goal the current index number.

Running Times

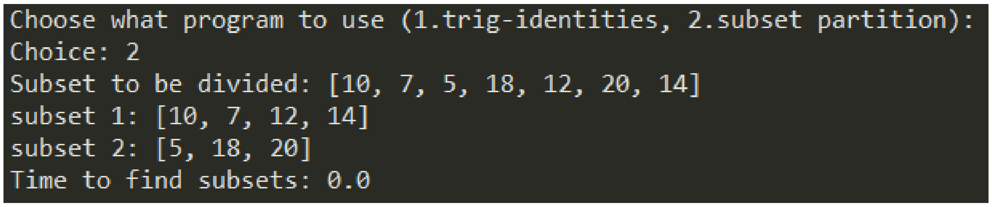




Outputs



```
Choose what program to use (1.trig-identities, 2.subset partition):
Choice: 1
Function: sin(t)
Identity: ['2*(sin(t/2)*cos(t/2))']

Function: cos(t)
Identity: ['cos(-t)']

Function: tan(t)
Identity: ['sin(t)/cos(t)']

Function: sec(t)
Identity: ['1/cos(t)']

Function: -sin(t)
Identity: ['sin(-t)']

Function: -cos(t)
Identity: []

Function: -tan(t)
Identity: ['tan(-t)']

Function: sin(-t)
Identity: ['-sin(t)']

Function: cos(-t)
Identity: ['cos(t)']

Function: tan(-t)
Identity: ['-tan(t)']

Function: sin(t)/cos(t)
Identity: ['tan(t)']

Function: 2*(sin(t/2)*cos(t/2))
Identity: ['sin(t)']

Function: sin(t)*sin(t)
Identity: ['1-(cos(t)*cos(t))', '(1-cos(2*t))/2']

Function: 1-(cos(t)*cos(t))
Identity: ['sin(t)*sin(t)', '(1-cos(2*t))/2']

Function: (1-cos(2*t))/2
Identity: ['sin(t)*sin(t)', '1-(cos(t)*cos(t))']
```



```
Choose what program to use (1.trig-identities, 2.subset partition):
Choice: 2
Subset to be divided: [10, 7, 5, 18, 12, 20, 14]
subset 1: [10, 7, 12, 14]
subset 2: [5, 18, 20]
Time to find subsets: 0.0
```

Appendix

```
"""
CS2303 - Data Structures
Jesus A. Hernandez - 80629917
Lab#8 - Algorithm Design Techniques
Instructor - Dr. Olac Fuentes
TA - Anindita Nath & Maliheh Zargaran

Analyze different problems and find a solution to it implementing one of the
different algorithm design techniques such as backtracking and randomization.
Last Modified on May 9, 2019
"""
import numpy as np
import random
import time
import math
from mpmath import *

"""
Trigonometric Identities Methods ---------------------------------------------
"""
#Create a list that holds identities and stores the number of times it has matched
def trig_functions():
    trig = [[] for i in range(16)]
    trig[0].append("sin(t)")
    trig[1].append("cos(t)")
    trig[2].append("tan(t)")
    trig[3].append("sec(t)")
    trig[4].append("-sin(t)")
    trig[5].append("-cos(t)")
    trig[6].append("-tan(t)")
    trig[7].append("sin(-t)")
    trig[8].append("cos(-t)")
    trig[9].append("tan(-t)")
    trig[10].append("sin(t)/cos(t)")
    trig[11].append("2*(sin(t/2)*cos(t/2))")
    trig[12].append("sin(t)*sin(t)")
    trig[13].append("1-(cos(t)*cos(t))")
    trig[14].append("(1-cos(2*t))/2")
    trig[15].append("1/cos(t)")
    return trig

"""
Compare a function with the different trigometric identities,if one
matches a certain amount of times add it to a list then return this
list of found identities.
"""
def Eval(func,tries=100,tolerance=0.0001):
    trig = trig_functions()
    identities = []
    for i in range(len(trig)):
        if trig[i][0] != func:
            counter = 0
            for j in range(tries):
                t = random.uniform(-math.pi,math.pi)
                val1 = eval(func)
                val2 = eval(trig[i][0])
                if np.abs(val2-val1) < tolerance:
                    counter += 1
            if counter == tries:
                temp = trig[i][0]
                identities.append(temp)
    return identities


"""
Partition Subset Methods ----------------------------------------------------
"""
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])
    if res:
        subset.append(S[last])
        return True, subset
    else:
        return subsetsum(S,last-1,goal)

"""
Find the total sum of the elements in the set then determine
if the set can be partitioned into 2, return the half of the
total sum.
"""
def half(S):
    sum = 0
    for i in S:
        sum += i
    if sum%2 != 0:
        print("No partition exists.")
        return
    goal = sum//2
    return goal

"""
Main ------------------------------------------------------------------------
"""
print("Choose what program to use (1.trig-identities, 2.subset partition):",end='')
choice = 0
while 1: # Continue until 1 or 2 is input
    try:
        choice = int(input("Choice: "))
        if choice == 1 or choice == 2:
            break
        else:
            print("Choose 1 or 2")
    except:
        print("Choose 1 or 2")

if choice == 1:
    start = time.time()
    trig = trig_functions()
    for i in range(len(trig)):
        print("Function:",trig[i][0])
        print("Identity:",Eval(trig[i][0]),"\n")
    end = time.time()
    print("Time to find trigonometric identities:",end-start)
else:
    S = [10,7,5,18,12,20,14]
```

```
print("Subset to be divided:",S)

start = time.time()

goal = half(S)

if goal != None:

    t,sub1 = subsetsum(S,len(S)-1,goal)

    sub2 = []

    for i in S:

        if i not in sub1:

            sub2.append(i)

    print("subset 1:",sub1)

    print("subset 2:",sub2)

end = time.time()

print("Time to find subsets:",end-start)
```

I certify that this essay is original work prepared by me, Jesus A Hernandez.