**Author:** Julian Gonzalez

**Assignment:** Lab 8Report

**Course:** CS 2302 - Data Structures 10:30-11:50

**Instructor:** Fuentes, Olac

**T.A.:** Nath, Anindita

***Introduction***

In this assignment we used where to used randomization and backtracking to complete two given problems. For randomization we had to discover different trigonometric identities with a given list of trig functions. The second question was to solve the partition problem with backtracking

***Solutions***

Discovering identities was done by two functions. The first function would take a list of the given trig list. Inside this “discover” function a for loop was used to iterate x number of times. Each iteration selected two random numbers from the list the two numbers where compared in an if statement and if they where not the same. Inside the if statement another if statement was used which would call the “testIden” function which had the parameters of two random items of the trig list. The testIden function was essentially the same code provided for us on the class website the only change that was needed for this assignment was changing the range of the random number generator to be from -pi to pi.

The partition problem was also done with two functions. The first of which was “subPartition”. subPartition has the parameter S which was the list that would be used for the partition problem. First the sum of S was found then an if statement would check if the sum of the problem was odd it would return false as an odd sum wont allow for the subset sum to work. After that a return statement was used that called the “subsetsum” function with the parameters of S (the original list), the length of s and finally the sum of s divided by two. The subsetsum function was essentially the same code provided for us on the class website with two minor changes to allow it to work for our problem. The two changes were just changing the range of “S[last]” to be “S[last]-1”

***Experimental***

For the randomization problem the way I experimented was changing the number of times that two random numbers (trig identities) where selected to be compared. Randomization the only variable that could change was the number of items that where in the list for the partition problem.

A screenshot of a cell phone

Description automatically generated

***Conclusion***

I learned how to implement backtracking and randomized algorithms and be able to understand their practical uses.

Appendix:

1. # -\*- coding: utf-8 -\*-
2. """
3. Course: CS-2302 Data-Stuctures
4. Author: Julian Gonzalez
5. Assignment: Lab 8
6. Intstuctor: Olac Fuentes
7. T.A"s: Anindita Nath, Maliheh Zargaran
9. """
10. **import** random
11. **import** math
12. **import** numpy as np
13. **import** time
14. **from** mpmath **import** \* #not needed
15. #import mpmath #not needed
17. #used to find which trig identities are equal
18. **def** discover(iden):
19. x = 200
20. **for** i **in** range(x):
21. y = random.randint(0,len(iden)-1)
22. z = random.randint(0,len(iden)-1)
23. **if** y != z:
24. **if** testIden(iden[y],iden[z]):
25. **print**(f"Try {i}. {iden[y]} and {iden[z]} are equal")
27. **def** testIden(f1,f2,tries=1000,tolerance=0.0001):#used code provided in class modified to work for trig identities
28. **for** i **in** range(tries):
29. t = random.uniform(-math.pi,math.pi)#throws never used error ignore#random num from -pi to pi to be used by the identities
30. y1 = eval(f1)
31. y2 = eval(f2)
32. **if** np.abs(y1-y2)>tolerance:
33. **return** False
34. **return** True
36. **def** subsetsum(S,last,goal):#modified version of given class code (only changed the range of last-1 )
37. **if** goal ==0:
38. **return** True, []
39. **if** goal<0 **or** last<0:
40. **return** False, []
41. res, subset = subsetsum(S,last-1,goal-S[last-1]) # Take S[last] #changed to last-1
42. **if** res:
43. subset.append(S[last-1])#changed to last-1
44. **return** True, subset
45. **else**:
46. **return** subsetsum(S,last-1,goal) # Don't take S[last]
48. **def** subPartition(S):
49. sumS = sum(S) #first get the sum of the entire list
50. **if** sumS % 2 != 0:# check if sum of s is odd if yes then return false cannout have two equal subsets with odd sum
51. **return** False
53. **return** subsetsum (S,len(S), sumS // 2) #call subsetsum function given to us in class (subset with half of the total sum)

56. trigIden = ["sin(t)","cos(t)","tan(t)","sec(t)","-sin(t)","-cos(t)","-tan(t)",
57. "sin(-t)","cos(-t)","tan(-t)","sin(t)/cos(t)","2\*sin(t/2)\*cos(t/2)",
58. "sin(t)\*sin(t)","1-(cos(t)\*cos(t))","(1-cos(2\*t))/2","1/cos(t)"
59. ]
60. start = time.time()
61. discover(trigIden)
62. end = time.time()
63. **print**('Time:',end-start)
65. **print**()
67. S = [3,2,1,3,3]#[2, 4, 5, 9, 12]#[3, 1, 1, 2, 2, 1]#[1, 3, 5, 9, 12]
69. start1 = time.time()
70. tup= subPartition(S)#have to use tuple
71. end1 = time.time()
72. **print**('Time:',end1-start1)

75. s1= tup[1]#take the array out of the tuple
76. s2 = S.copy() #copy of old array used for finding other partion
78. **if** tup[0]:
79. **for** i **in** s1:
80. s2.remove(i)#removes the first occurance of a number from subset one
81. **print**(f"{tup[0]} there exist a partition {s1} and {s2} in {S}")
83. **else**:
84. **print**(f"{tup[0]} there exist no partition")

I Julian Gonzalez 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.

* Julian Gonzalez