Lab 8

**Introduction:**

In this lab we were given a several trigonometric functions, and we were given the task to come up with a function that would determine which trigonometric functions are equal to each other. In order to do this, we were instructed to use a randomization. The second question required us to partition a list of numbers into two sets that add up to the sum of the original list. The sum of each of the set must be equal to each other as well. For question two we were instructed to use backtracking to partition two sets from the original.

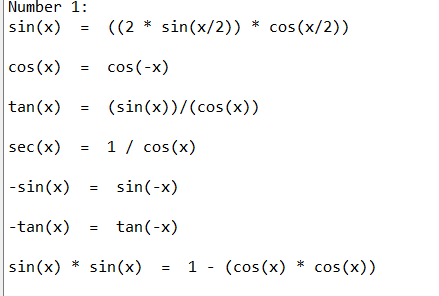
**Proposed Solutions:**

For question one, I made a list of strings, where each string in the list represents a trigonometric function. I then used the equals function that was provided in class to compare all the elements in the list with each other. The equals function provided to us in class uses randomization to plug random numbers into a function.

In question two the first thing I check for was if the sum of the original list was even or not, only even sums will have been able to divided into two sets with the same sums. If the sum was even, I used the subsetsum function provided to us in class and I set the goal to half the sum of the original list. This gave me my first set, so then all I did after that was make a set with all the remaining numbers in the original list that weren’t in the first set I made. I then compared the sums of both of the subsets, and if they’re the same I return the two sets, and if they’re not I return none.

**Experimental Results:**

For question one, I only used the functions that were required. I went and actually checked the results to make sure that they were all equal. Below is a picture of the results that I got.



For the run-times I used the same list, I just doubled, tripled, and quadrupled the length of the list to get a better idea of the run-time. Below is a graph representing the run-times and the lengths of the lists.

Based off of this graph the function I made has a O(n) run-time, linear run-time.

For question two I tested my method by plugging in lists that I knew would have subsets and lists that I knew wouldn’t have any subsets. Below is a screenshot of the results I got using the list provided to us in the lab.

A picture containing object

Description automatically generated

For the run-times I changed the length of the list each time. Below is a graph that represents the run-times in relation to the length of the list.

By looking at this graph, I’d have to say that the run-time is O(logn) because the time is slowly increasing as the length of the list increases greatly.

**Conclusion:**

This lab has shown me how to implement randomization and backtracking. The second part of this lab was initially pretty challenging to me, but then I ended up using the subsetsum method that was provided to us, and that really helped.

**Source Code / Appendix:**

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing indoor

Description automatically generated

A screenshot of a cell phone

Description automatically generated

“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.”

Joey Roe