Nancy Hernandez

Olac Fuentez

Lab 8 Report

**Introduction**

The purpose of this lab is to write a program that “discovers” trigonometric identities and tests all combinations of the trigonometric expressions provided. As well as creating another program that solves the partition problem using backtracking.

**Proposed Solution**

In order to test the equalities between all the trigonometric functions I used the equal method professor Fuentes provided. I did however change the x to generate random floats from negative pi to positive pi. Everything else I kept the same. I then created a list containing all of the expressions as strings. To go about comparing all the strings to each other I created a for loop that goes through each expression, and inside this for loop I created another for loop that also goes through the length of the list. Inside this second for loop I called the equal method twice but in the first call I used ‘i’ as a parameter and the second I use ‘j’ as the parameter, and finally I just printed the output.

When it came to the second part of the lab, I used the functions subsetsum and edit\_distance that were both provided by Professor Fuentes. For the method subsetsum I changed nothing and kept it as it was. No changes were made to the edit\_distance method either. The method edit\_distance computes the distance from string s1 to string s2 using dynamic programing and it also return the edit distance matrix from s1 to s2. When wanting to find whether there is or isn’t a partition to the subset, I created a for loop that repeats 100 times. Inside this for loop I made the ‘goal’ to be equal to ‘i’ and I made ‘a’ and ‘s’ equal to the calling of the method subsetsum with the parameters being the subset itself, the length of the subset minus one, and finally the ‘i’. Followed by this is and if and else statement. If ‘a’ \the I print the solution which is ‘s’ in this case and if it is anything else but ‘a’ then I just print that there is no solution.

**Setup**

To complete this lab I used an HP Pavilion x360 Convertible with a 2.71 GHz Intel® Core(TM) i5 processor.

**Results**

A screenshot of a computer

Description automatically generated**A screenshot of a computer

Description automatically generated**Part 1:

Part 2:

**A screenshot of a computer

Description automatically generated**A screenshot of a computer

Description automatically generated

**Time Analysis**

|  |  |
| --- | --- |
| **Function** | **Time** **Complexity** |
| Edit\_distance | O(mn) |
| subsetsum | O(sum\*n) |

**Conclusion**

I concluded that using dynamic programing is very beneficial when wanting to break a problem down into sub-problems and trying to avoid computing the same results over again. Also, how backtracking works when needing to solve a particular partition problem. I got understand how the edit\_distance and the subset methods worked with different variables too.

**Appendix**

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

for i in range(tries):

x = random.random()

y1 = eval(f1)

y2 = eval(f2)

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

return False

return True

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 edit\_distance(s1, s2):

# Computes edit distance from string s1 to string s2 using dynamic programming

# Returns edit distance matrx from string s1 to string s2

# The actual edit distance is stored in d[-1,-1]

d = np.zeros((len(s1) + 1, len(s2) + 1), dtype = int)

d[0, :] = np.arange(len(s2) + 1)

d[:, 0] = np.arange(len(s1) + 1)

for i in range(1, len(s1) + 1):

for j in range(1, len(s2) + 1):

if s1[i - 1] == s2[j - 1]:

d[i, j] = d[i - 1, j - 1]

else:

n = [d[i, j - 1], d[i - 1, j - 1], d[i - 1, j]]

d[i, j] = min(n) + 1

return d

**Academic Dishonesty Statement**

I, Nancy Hernandez, was not involved in any copying from or providing information to another student, possessing unauthorized materials during a test, or falsifying data in laboratory reports. Neither did I participate in any type of collusion involving collaboration with another person to commit an academically dishonest act.