**CS 2302 Data Structures**

**Fall 2019**

**Lab Report #1**

Due: September 6th, 2019

Professor: Olac Fuentes

TA: Anindita Nath

Programmer: Miriam Olague

**Introduction**

For this lab we were asked to find the anagrams of a word using recursion. The purpose of this lab is to get familiarized and practice recursion. The main objective of this lab is to learn how to manipulate sets and arrange the words the sets contain to obtain the anagrams of the word the user entered recursively.

**Proposed Solution Design and Implementation**

**Part #1:**

For this operation, I used the code from Chapter 2.6.1 on Zybooks as a reference. With this code I iterated through every word in ‘words\_alpha.txt’ (I created it into a set). I first saved the first character of the word into a temporary variable and the remaining of the word was saved into a different variable. After I iterated through the entire word, I saved the word into a temporary set and then compared it to the original set. I then updated the set by comparing the words (all the anagrams) with the original set and see if the anagrams of the words existed inside of the original set. I also called the method (scrambled is a recursive method) and iterated through all the word set I created.

**Part #2.1:**

For this operation, I approached it in a very similar way as **Part #1**. I iterated through every word in ‘words\_alpha.txt’ (I created it into a set). Inside of an if statement, I compared the first character of the word with the rest of the characters inside that word to prevent it from making recursive calls if the character repeats itself. If the conditions are true for this word, I save the first character inside of a temporary variable. Then, I saved the rest of the word inside of another temporary variable. After iterating through the entire word, I then saved the word into a temporary set and compared it to the original set. I continuously updated the set each time if the new anagram existed inside of the original word set.

**Part #2.2:**

For this operation, I approached this problem in a similar way as Part #2.1. I iterated through all the words that are inside ‘words\_alpha.txt’ with a set. I created an if statement, in which I allowed it to make recursive calls if and only if the word the user inputs is inside of the set I created, in which it contains all the prefixes of the original word set (words\_alpha.txt). I then saved the first character of the word inside of a temporary variable. Then saved the rest of the characters of the word into a temporary variable. After iterating through all the word, I saved the word into a temporary set and compared it to the original set (words\_alpha.txt). I repeated this step for all anagrams that were created so I could compare it and then update it if the anagram existed inside of the original word set.

**Experimental Results**

**Part #1:**

For this operation, I created an empty set, in which all the existing anagrams of the user’s input were added into. I first thought that I had to start the time inside of the recursion method but then realized that the time would reset. I attempted to print all the existing anagrams inside of the recursion method, but then decided it would be better to put them in a set and then print them outside of the recursion method. I tried printing the anagrams inside of the recursion call, but I found it cleaner to print them outside.

A screenshot of a cell phone

Description automatically generated User’s input = poster

A screen shot of a computer

Description automatically generated User’s input = by

A screenshot of a cell phone

Description automatically generated User’s input = table

**Part #2.1:**

For this operation, I had a hard time understanding how I could compare a character with the rest of the characters. All the experiments I made were the same ones as **Part #1**. The only different experiment I had to do was how to compare the characters with the rest.

A screenshot of a cell phone

Description automatically generated User’s input = poster

A screen shot of a computer

Description automatically generatedUser’s input = by

A screenshot of a cell phone

Description automatically generated User’s input = table

**Part #2.2:**

For this operation, I approached this problem similarly as **Part #2.1**. The way I created a set with all the prefixes of the original set (words\_alpha.txt) was with a list. I made a for loop in which I added the prefixes of the set and then converted it to a set in order to compare it to the original set (words\_alpha.txt), however, all of the anagrams of the word were not found inside of the word set. Since many of the words that were in the list were added incorrectly into the set, it did not find all the existing anagrams inside of the original set (words\_alpha.txt).

A screenshot of a cell phone

Description automatically generatedUser’s input = poster

A screen shot of a computer

Description automatically generatedUser’s input = by

A screenshot of a cell phone

Description automatically generatedUser’s input = table

A screenshot of a cell phone

Description automatically generated

As the results show, finding the anagrams of a word takes more time iterating through every word in the word set rather than finding the anagrams of a word by only calling the recursive method if no characters repeat. Since I separated **Part #2.1** (no duplicates) and **Part #2.2** (prefixes), if the word is found, it takes more time to iterate through the prefix set and iterating through the original word set (words\_alpha.txt).

**Conclusion**

This lab helped me reinforce my knowledge of recursion. It helped me understand and practice the way lists and sets can be used. This lab was challenging but managed to learn a lot from it the more I spent time on it. I enjoyed it since I am barely learning Python and I find it very interesting. I learned how to be a little bit more organized with my code and learned to give meaningful names to my variables so that way I don’t get lost. I learned how to use recursion with python and how to scramble words in every possible way (anagrams).

**Appendix**

A screenshot of a computer screen

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a computer

Description automatically generatedA screen shot of a computer

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.