CS2302 - Data Structures

Fall 2019

Lab 1 - Recursion

Due Friday, September 6, 2019

An anagram is a permutation of the letters of a word that produces another word. Your task for this lab consists of writing a program that asks a user to input a word and then prints all the anagrams of that word.

Part 1

Download the file *https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt*, which contains over 466000 words in the English language. Your program should read the words in the file and, as the words are read, they should be stored in a set (see section 13.24 in the online textbook). Your program should then use a recursive function, similar to the ones seen in class and described in the textbook, to find the anagrams. The anagrams should be displayed in alphabetical order and contain no duplicates. Your program should also display the time it took to find the anagrams.

This a possible run of your program:

```
Enter a word or empty string to finish: poster
The word poster has the following 6 anagrams:
presto
repost
respot
stoper
topers
tropes
It took 0.000000 seconds to find the anagrams

Enter a word or empty string to finish: university
The word university has the following 0 anagrams:
It took 12.506083 seconds to find the anagrams

Enter a word or empty string to finish: permutation
The word permutation has the following 1 anagrams:
importunate
It took 138.054811 seconds to find the anagrams

Enter a word or empty string to finish:
Bye, thanks for using this program!
```

Part 2

As you can notice, finding the anagrams of long words takes a very long time, since all permutations of the characters in the word need to be generated, and there are $n!$ permutations, where n is the length of the word. For part 2, you will implement the following two optimizations:

- If the string has duplicate characters, only make recursive calls the first time the character appear. This will avoid generating the same anagram multiple times
- Stop recursion if the partial word you have is not a prefix of any word in the word set. To do this, you'll need to create a set containing the prefixes of all the words in the word set. For example, if your word

set is {'data', 'science'}, your prefix set should be {'', 'd', 'da', 'dat', 's', 'sc', 'sci', 'scie', 'scien', 'scienc'}

After applying the optimizations, your results should look like this:

```
Enter a word or empty string to finish: poster
The word poster has the following 6 anagrams:
presto
repost
respot
stoper
topers
tropes
It took 0.000000 seconds to find the anagrams

Enter a word or empty string to finish: university
The word university has the following 0 anagrams:
It took 0.015600 seconds to find the anagrams

Enter a word or empty string to finish: permutation
The word permutation has the following 1 anagrams:
importunate
It took 0.046800 seconds to find the anagrams

Enter a word or empty string to finish:
Bye, thanks for using this program!
```

Write a report describing your programs and experiments, as described in the syllabus.