Assignment A1: Anagram Finder

Richard Neal

February 05, 2013

Introduction & Summary

For our first COMP 255 assignment, we were tasked with creating a program to find all anagrams for a given word or phrase. Given a plain text file of dictionary words and a word or phrase as input, the program will find all valid permutations of the input that are composed of words from the initial dictionary.
To start out the assignment, we brainstormed in class ways of finding anagrams for a given input word or sentence. My initial idea was to use the Trie data structure, which seemed appropriate for storing the list of dictionary words, and could then be traversed given random permutations of the input string to find anagrams. Other ideas were brought up, including using the Trie to store the permutations of the input string, and given concerns over efficiency, creating a Trie of the whole dictionary fell to the back of my mind.

Idea #1: Trie of Input String

After giving some consideration, I began implementing the Trie in Python, intending to use all permutations (using `itertools.permutations`) as the seed for the Trie, then traversing it while checking the path against valid words. However, concerns over efficiency led me to try a different approach.

Idea #2: Hash Table

My second idea was hash the values of each word to a unique hash for each permutation of the same word, so that I would only have to check one word for each of its own anagrams. However, this wouldn't work well for multiple words, and regardless, I could have just sorted the word.

Idea #3: Trie of Input Dictionary

For my solution, I went back to my original idea, and generated a trie of all the dictionary words, which doesn't actually take long at all. Then, I recursively traverse the trie, taking out letters from the input as I go, until I reach both the end of my letters and the end of a word. To remove duplicates, I only use anagrams whose words are in alphabetical order, for instance, I won't use "finder anagram," I'll only use "anagram finder," as "anagram" ¡ "finder" in python syntax.

Optimization

After arriving on an algorithm that produced proper results with no duplicate rearrangements, the time had come to optimize my code. I was initially dubious that much could be done, but after putting the program through two profilers, I discovered some calls that could be improved upon. The first, and most significant performance increase came from switching the dictionary that contained the letters of the input string and their counts with a defaultdict. Defaultdict is a subclass of Python's built-in dictionaries with one key difference: as its name suggests, keys have a default class value, defined at the time the defaultdict is created. In my case, the default value was an int, giving keys with no set value a default value of zero. This meant that I no longer had to call the dictionaries' get methods, which have a much greater overhead then simply indexing into the dictionary, as evidenced by the performance increase.

Optimization	**statprof**	**cProfile**	**% Decrease** (statprof)	**% Decrease** (cProfile)
Original Algorithm	16.39	42.74	N/A	N/A
defaultdict	13.00	23.48	20%	45%
iteritems	11.84	23.45	9%	0%

Note: All test were conducted with the string "wheaton college," and produced the same number of results, as I only counted actual code speed-ups, not fixes to remove duplicates, etc., as optimizations.

Sample

```
Please enter a word or phrase for which to find anagrams: tom
    armstrong
Would you like to enter advanced options for the anagram finder? Enter
    true or false: true
Please enter a list of comma-separated words that must be in the
    anagram (Hit enter to ignore): storm
Please enter a list of comma-separated words that must NOT be in the
    anagram (Hit enter to ignore): gram
Please enter a maximum number of words for each anagram (Hit enter to
    ignore): 3
    1: mont rag storm
    2: morn storm tag
    3: norm storm tag
    4: rang storm tom
There were 4 unique anagrams made from 'tom armstrong' using the
    dictionary 'english.txt'.
```

COMP 255 Anagrams and Interpretations

- Shawn Wise → Saw she win
  Shawn will bear witness and participate in the rise of women in computing.

- Christopher DeMolles → Code 'em, he's pro, thrills
  Chris wows the crowd as a professional coder with his stunning use of a generator function.

- Andrew Thomas → Saw the random
  Andrew will become a professional card shark, using his skills for calculating the odds to win big in Vegas.

- Malcolm Eaton → Local man, tome
  Malcolm, who hails from the area, will one day be the stuff of legend, with tomes written on his accomplishments

- Matthew Donovan → On than, Tom waved
  Tom waves goodbye to Matt after graduation, telling him to continue on to greater endeavors.

- Caleb Pudvar → Carved lab up
  Caleb emerged as the clear winner in Comp Org lab, stunning Professor Gousie with his assembly skills

- Alexander Dyck → Excel day drank
  Alex's new spreadsheet app became so successful, Excel began to run into some personal problems

- Samuel Kottler → Maker, tell to us
  Sam retires from system administration at 28 to run the world's most popular makerspace, ultimately writing a book on his accomplishments

- Claudia D'Adamo → Ada ad, I am cloud
  Claudia starts a new cloud-based service, named after Countess Lovelace, with advertisements featuring the Dame's image

- Taylor Wright-Sanson → Won arts as Lyon, right?
  Mark remembers back on the time Taylor won the college's most pretigious art award.

- Clayton Rieck → Click, earn toy
  Clayton starts a hugely popular online gaming site, where toys are just a click away

- Kelsey Hichens → Cheese link shy
  Despite Kelsey's love of cheese, she prefers it in block form

- Alexander Taoultsides → Does lead lie? Saturn tax!
  As the planet begins to run out of minerals, Alex begins a giant off-world mining corporation, and fights to keep what is found off-world, taxed off-world.

Appendix

1. Original Performance:

statprof:

```
  %   cumulative      self
 time    seconds   seconds  name
 31.11      5.10      5.10  Anagram.py:60:anagramization
 23.51      3.85      3.85  Anagram.py:58:anagramization
 13.11     15.91      2.15  Anagram.py:64:anagramization
  7.35      1.21      1.21  Anagram.py:61:anagramization
  4.84      0.79      0.79  Anagram.py:65:anagramization
  2.90      0.48      0.48  Anagram.py:63:anagramization
  2.52      0.41      0.41  Anagram.py:45:anagramization
  2.42      0.40      0.40  Anagram.py:49:anagramization
  2.42      0.40      0.40  Anagram.py:43:anagramization
  2.13      0.35      0.35  Anagram.py:47:anagramization
  1.79      0.29      0.29  Anagram.py:53:anagramization
  1.64     15.91      0.27  Anagram.py:56:anagramization
  1.26      0.21      0.21  Anagram.py:51:anagramization
  1.11      0.18      0.18  Anagram.py:35:insert
  0.53      0.09      0.09  Anagram.py:70:__init__
  0.48      0.08      0.08  Anagram.py:68:__init__
  0.15      0.05      0.02  Anagram.py:34:insert
  0.10      0.02      0.02  Anagram.py:77:findChild
  0.10      0.02      0.02  Anagram.py:73:addChild
  0.10      0.04      0.02  Anagram.py:37:insert
  0.05      0.17      0.01  Anagram.py:36:insert
  0.05     16.39      0.01  anagramMainStat.py:22:<module>
  0.05      0.01      0.01  Anagram.py:33:insert
  0.05      0.01      0.01  Anagram.py:38:insert
  0.05      0.01      0.01  Anagram.py:52:anagramization
  0.05      0.01      0.01  Anagram.py:74:addChild
  0.05      0.01      0.01  Anagram.py:30:insert
  0.05      0.01      0.01  Anagram.py:76:findChild
  0.05      0.48      0.01  Anagram.py:14:__init__
  0.00     15.91      0.00  Anagram.py:21:findAnagrams
  0.00      0.48      0.00  anagramMainStat.py:7:main
  0.00     15.91      0.00  anagramMainStat.py:8:main
---
Sample count: 2067
Total time: 16.390000 seconds
```

cProfile:

```
         67483639 function calls (61071190 primitive calls) in 42.744 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 Anagram.py:1(<module>)
        1    0.000    0.000   41.794   41.794 Anagram.py:16(findAnagrams)
        1    0.000    0.000    0.000    0.000 Anagram.py:18(<dictcomp>)
        1    0.000    0.000    0.000    0.000 Anagram.py:25(Trie)
        1    0.000    0.000    0.000    0.000 Anagram.py:26(__init__)
        1    0.000    0.000    0.000    0.000 Anagram.py:3(Anagram)
    45427    0.548    0.000    0.852    0.000 Anagram.py:30(insert)
        1    0.056    0.056    0.926    0.926 Anagram.py:4(__init__)
6412450/1   30.391    0.000   41.793   41.793 Anagram.py:43(anagramization)
        1    0.000    0.000    0.000    0.000 Anagram.py:67(Node)
   110722    0.098    0.000    0.098    0.000 Anagram.py:68(__init__)
   110721    0.025    0.000    0.025    0.000 Anagram.py:73(addChild)
   363878    0.131    0.000    0.181    0.000 Anagram.py:76(findChild)
        1    0.024    0.024   42.744   42.744 anagramMain.py:4(<module>)
        1    0.000    0.000   42.719   42.719 anagramMain.py:6(main)
   350937    0.052    0.000    0.052    0.000 len
       10    0.000    0.000    0.000    0.000 method 'count' of 'str' objects
        1    0.000    0.000    0.000    0.000 method 'disable' of '_lsprof.Profiler' obje
    20278    0.046    0.000    0.046    0.000 method 'format' of 'str' objects
 50039160    7.384    0.000    7.384    0.000 method 'get' of 'dict' objects
  6412450    2.532    0.000    2.532    0.000 method 'items' of 'dict' objects
  1598039    0.639    0.000    0.639    0.000 method 'join' of 'str' objects
    45428    0.008    0.000    0.008    0.000 method 'lower' of 'str' objects
  1928698    0.799    0.000    0.799    0.000 method 'split' of 'str' objects
    45427    0.009    0.000    0.009    0.000 method 'strip' of 'str' objects
        1    0.000    0.000    0.000    0.000 method 'translate' of 'str' objects
        1    0.000    0.000    0.000    0.000 open
        1    0.000    0.000    0.000    0.000 strop.maketrans
```

2. After Switching to defaultdict:

statprof:

```
  %   cumulative      self
 time    seconds   seconds  name
 28.56      3.71      3.71  Anagram.py:61:anagramization
 16.63     12.51      2.16  Anagram.py:67:anagramization
 12.24      1.59      1.59  Anagram.py:63:anagramization
 11.02      1.43      1.43  Anagram.py:64:anagramization
  6.03      0.78      0.78  Anagram.py:68:anagramization
  4.38      0.57      0.57  Anagram.py:66:anagramization
  3.47      0.45      0.45  Anagram.py:46:anagramization
  3.35      0.44      0.44  Anagram.py:52:anagramization
  2.98      0.39      0.39  Anagram.py:50:anagramization
  2.31      0.30      0.30  Anagram.py:48:anagramization
  1.95      0.25      0.25  Anagram.py:56:anagramization
  1.64      0.21      0.21  Anagram.py:54:anagramization
  1.46     12.51      0.19  Anagram.py:59:anagramization
  1.04      0.13      0.13  Anagram.py:38:insert
  0.73      0.10      0.10  Anagram.py:71:__init__
  0.37      0.48      0.05  Anagram.py:15:__init__
  0.30      0.04      0.04  Anagram.py:79:findChild
  0.30      0.04      0.04  Anagram.py:80:findChild
  0.30      0.04      0.04  Anagram.py:73:__init__
  0.24      0.04      0.03  Anagram.py:40:insert
  0.18      0.02      0.02  Anagram.py:55:anagramization
  0.18      0.10      0.02  Anagram.py:37:insert
  0.06      0.01      0.01  Anagram.py:72:__init__
  0.06      0.01      0.01  Anagram.py:36:insert
  0.06     13.00      0.01  anagramMainStat.py:22:<module>
  0.06      0.01      0.01  Anagram.py:41:insert
  0.06      0.01      0.01  Anagram.py:76:addChild
  0.00      0.14      0.00  Anagram.py:39:insert
  0.00      0.48      0.00  anagramMainStat.py:7:main
  0.00     12.51      0.00  anagramMainStat.py:8:main
  0.00     12.51      0.00  Anagram.py:24:findAnagrams
---
Sample count: 1642
Total time: 13.000000 seconds
```

cProfile:

```
         17808362 function calls (11395913 primitive calls) in 23.482 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.001    0.001    0.005    0.005 Anagram.py:1(<module>)
        1    0.000    0.000   22.527   22.527 Anagram.py:17(findAnagrams)
        1    0.000    0.000    0.000    0.000 Anagram.py:28(Trie)
        1    0.000    0.000    0.000    0.000 Anagram.py:29(__init__)
    45427    0.453    0.000    0.850    0.000 Anagram.py:33(insert)
        1    0.000    0.000    0.000    0.000 Anagram.py:4(Anagram)
6412450/1   18.602    0.000   22.527   22.527 Anagram.py:46(anagramization)
        1    0.055    0.055    0.924    0.924 Anagram.py:5(__init__)
        1    0.000    0.000    0.000    0.000 Anagram.py:70(Node)
   110722    0.198    0.000    0.198    0.000 Anagram.py:71(__init__)
   110721    0.025    0.000    0.025    0.000 Anagram.py:76(addChild)
   363878    0.125    0.000    0.174    0.000 Anagram.py:79(findChild)
        1    0.025    0.025   23.482   23.482 anagramMain.py:4(<module>)
        1    0.000    0.000   23.452   23.452 anagramMain.py:6(main)
        1    0.000    0.000    0.000    0.000 bisect.py:1(<module>)
        1    0.002    0.002    0.004    0.004 collections.py:1(<module>)
        1    0.000    0.000    0.000    0.000 collections.py:25(OrderedDict)
        1    0.000    0.000    0.000    0.000 collections.py:356(Counter)
        1    0.002    0.002    0.002    0.002 heapq.py:31(<module>)
        1    0.000    0.000    0.000    0.000 keyword.py:11(<module>)
   350937    0.053    0.000    0.053    0.000 len
       10    0.000    0.000    0.000    0.000 method 'count' of 'str' objects
        1    0.000    0.000    0.000    0.000 method 'disable' of '_lsprof.Profiler' obje
    20278    0.041    0.000    0.041    0.000 method 'format' of 'str' objects
   363878    0.049    0.000    0.049    0.000 method 'get' of 'dict' objects
  6412450    2.357    0.000    2.357    0.000 method 'items' of 'dict' objects
  1598039    0.630    0.000    0.630    0.000 method 'join' of 'str' objects
    45428    0.008    0.000    0.008    0.000 method 'lower' of 'str' objects
  1928698    0.844    0.000    0.844    0.000 method 'split' of 'str' objects
    45427    0.010    0.000    0.010    0.000 method 'strip' of 'str' objects
        1    0.000    0.000    0.000    0.000 method 'translate' of 'str' objects
        1    0.001    0.001    0.001    0.001 open
        1    0.000    0.000    0.000    0.000 strop.maketrans
```

3. After switching to dictionary.iteritems():

statprof:

```
  %   cumulative      self
 time    seconds   seconds  name
 23.80      2.82      2.82  Anagram.py:61:anagramization
 16.63     11.32      1.97  Anagram.py:67:anagramization
 13.25      1.57      1.57  Anagram.py:64:anagramization
 12.17      1.44      1.44  Anagram.py:63:anagramization
  5.54      0.66      0.66  Anagram.py:68:anagramization
  5.41      0.64      0.64  Anagram.py:66:anagramization
  3.65      0.43      0.43  Anagram.py:52:anagramization
  3.52      0.42      0.42  Anagram.py:50:anagramization
  3.25      0.38      0.38  Anagram.py:46:anagramization
  2.91      0.34      0.34  Anagram.py:48:anagramization
  1.96      0.23      0.23  Anagram.py:56:anagramization
  1.76     11.32      0.21  Anagram.py:59:anagramization
  1.69      0.20      0.20  Anagram.py:54:anagramization
  1.28      0.15      0.15  Anagram.py:38:insert
  1.22      0.14      0.14  Anagram.py:71:__init__
  0.47      0.10      0.06  Anagram.py:37:insert
  0.34      0.04      0.04  Anagram.py:80:findChild
  0.27      0.51      0.03  Anagram.py:15:__init__
  0.27      0.03      0.03  Anagram.py:73:__init__
  0.14      0.20      0.02  Anagram.py:39:insert
  0.07      0.01      0.01  Anagram.py:36:insert
  0.07     11.84      0.01  anagramMainStat.py:22:<module>
  0.07      0.01      0.01  Anagram.py:79:findChild
  0.07      0.01      0.01  Anagram.py:55:anagramization
  0.07      0.01      0.01  Anagram.py:74:__init__
  0.07      0.01      0.01  Anagram.py:44:insert
  0.07      0.01      0.01  Anagram.py:40:insert
  0.00      0.51      0.00  anagramMainStat.py:7:main
  0.00     11.32      0.00  anagramMainStat.py:8:main
  0.00     11.32      0.00  Anagram.py:24:findAnagrams
---
Sample count: 1479
Total time: 11.840000 seconds
```

cProfile:

```
         17808362 function calls (11395913 primitive calls) in 23.452 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.001    0.001    0.005    0.005 Anagram.py:1(<module>)
        1    0.000    0.000   22.451   22.451 Anagram.py:17(findAnagrams)
        1    0.000    0.000    0.000    0.000 Anagram.py:28(Trie)
        1    0.000    0.000    0.000    0.000 Anagram.py:29(__init__)
    45427    0.518    0.000    0.889    0.000 Anagram.py:33(insert)
        1    0.000    0.000    0.000    0.000 Anagram.py:4(Anagram)
6412450/1   19.671    0.000   22.451   22.451 Anagram.py:46(anagramization)
        1    0.061    0.061    0.970    0.970 Anagram.py:5(__init__)
        1    0.000    0.000    0.000    0.000 Anagram.py:70(Node)
   110722    0.161    0.000    0.161    0.000 Anagram.py:71(__init__)
   110721    0.026    0.000    0.026    0.000 Anagram.py:76(addChild)
   363878    0.134    0.000    0.184    0.000 Anagram.py:79(findChild)
        1    0.026    0.026   23.452   23.452 anagramMain.py:4(<module>)
        1    0.000    0.000   23.421   23.421 anagramMain.py:6(main)
        1    0.000    0.000    0.000    0.000 bisect.py:1(<module>)
        1    0.002    0.002    0.004    0.004 collections.py:1(<module>)
        1    0.000    0.000    0.000    0.000 collections.py:25(OrderedDict)
        1    0.000    0.000    0.000    0.000 collections.py:356(Counter)
        1    0.002    0.002    0.002    0.002 heapq.py:31(<module>)
        1    0.000    0.000    0.000    0.000 keyword.py:11(<module>)
   350937    0.051    0.000    0.051    0.000 len
       10    0.000    0.000    0.000    0.000 method 'count' of 'str' objects
        1    0.000    0.000    0.000    0.000 method 'disable' of '_lsprof.Profiler' obje
    20278    0.038    0.000    0.038    0.000 method 'format' of 'str' objects
   363878    0.050    0.000    0.050    0.000 method 'get' of 'dict' objects
  6412450    1.211    0.000    1.211    0.000 method 'iteritems' of 'dict' objects
  1598039    0.644    0.000    0.644    0.000 method 'join' of 'str' objects
    45428    0.008    0.000    0.008    0.000 method 'lower' of 'str' objects
  1928698    0.835    0.000    0.835    0.000 method 'split' of 'str' objects
    45427    0.011    0.000    0.011    0.000 method 'strip' of 'str' objects
        1    0.000    0.000    0.000    0.000 method 'translate' of 'str' objects
        1    0.001    0.001    0.001    0.001 open
        1    0.000    0.000    0.000    0.000 strop.maketrans
```