>

CodeKata

Because experience is the *only* teacher

- RSS

Search

Navigate…

- PragDave
- Kata
- Archives

Kata06: Anagrams

Back to non-realistic coding this week (sorry, Martin). Let's solve some crossword puzzle clues.

In England, I used to waste hour upon hour doing newspaper crosswords. As crossword fans will know, English cryptic crosswords have a totally different feel to their American counterparts: most clues involve punning or word play, and there are lots of anagrams to work through. For example, a recent Guardian crossword had:

```
1   Down:
2     ..
3     21. Most unusual form of arrest (6)
```

The hint is the phrase 'form of,' indicating that we're looking for an anagram. Sure enough 'arrest' has six letters, and can be arranged nicely into 'rarest,' meaning 'most unusual.' (Other anagrams include raster, raters, Sartre, and starer)

A while back we had a thread on the Ruby mailing list about finding anagrams, and I'd like to resurrect it here. The challenge is fairly simple: given a file containing one word per line, print out all the combinations of words that are anagrams; each line in the output contains all the words from the input that are anagrams of each other. For example, your program might include in its output:

```
1   kinship pinkish
2   enlist inlets listen silent
3   boaster boaters borates
4   fresher refresh
5   sinks skins
6   knits stink
```

```
7   rots sort
```

If you run this on the word list here you should find 20683 sets of anagrams (a total of 48162 words), including all-time favorites such as

```
1 crepitus cuprites pictures piecrust
2 paste pates peats septa spate tapes tepas
3 punctilio unpolitic
4 sunders undress
```

For added programming pleasure, find the longest words that are anagrams, and find the set of anagrams containing the most words (so “parsley players replays sparely” would not win, having only four words in the set).

Kata Objectives

Apart from having some fun with words, this kata should make you think somewhat about algorithms. The simplest algorithms to find all the anagram combinations may take inordinate amounts of time to do the job. Working though alternatives should help bring the time down by orders of magnitude. To give you a possible point of comparison, I hacked a solution together in 25 lines of Ruby. It runs on this wordlist in 1.8s on a 1.7GHz i7. It’s also an interesting exercise in testing: can you write unit tests to verify that your code is working correctly before setting it to work on the full dictionary.

Posted by Dave Thomas (@PragDave) Dec 24th, 2013

« Kata07: How'd I Do? Kata05: Bloom Filters »

Comments

MarkF · a year ago

To offer another point of comparison, this can be done with just 4 lines of LINQ:

```
File.ReadAllLines("wordlist.txt")
.GroupBy(w => String.Concat(w.OrderBy(c => c)))
.Where(g => g.Count() > 1)
.ToList().ForEach(g => Console.WriteLine(String.Join(" ", g)));
```


noodlefrenzy ➚ MarkF · a year ago

Hahaha, that's awesome, I was just implementing this Kata and wrote almost the exact same code - only difference is I used "new string(w.OrderBy(c => c).ToArray())", since Concat is perf-impaired (or at least used to be: http://stackoverflow.com/quest....

Recent Posts