DIT 374 Python for Data Science

Assignment 2 by Glenn Svanberg

Problem 1

The simple solution to this problem was too simply put all the words in one list. In order to access words with a specific char at a given place the program will then iterate over the list and put all the words that match the pattern in a new list called matches. This solution works fine as an example but for a big list of words or for many iterations it becomes very slow.

One solution that was used to make it more efficient was to store the words in a dictionary instead of in a list. This way the words can be accessed without the need to iterate through an entire list for every search. The dictionary holds a single char as a key that can be used to access it's value. The value of the dictionary is another dictionary, this dictionary uses the positions the char occurs in as it's keys. Values for the inner dictionary is a list of words that matches both of the above keys. Using this data structure makes no iterations needed for search, all the program has to do is go into the dictionary and the nested dictionary and return the list that is given.

As the evaluation shows this is a much faster operation to perform. The first evaluation iterated through a small list with only 10 words millions of times and the dictionary performed ~20 times faster than the list. However for a bigger list of words (words.txt) the difference was way more significant. This test was iterated 1000 times and the dictionary performed ~200000 times faster than the list solution.

However the dictionary takes more time to initialize, the list is very straightforward to structure in the program and that is a process required for both solutions, but the process of splitting the list into dictionaries takes a lot of time for big lists. The tests that were being performed for words.txt showed ~2 sec of initialization time. With that said the solution of a dictionary becomes mostly useful in a situation when the program can hold the list in memory for long times without and not have to initialize it for every use.

Problem 2

For the anagram problem a dictionary structure was used to find out what words are anagrams and not. The key for this dictionary is an alphabetically sorted version of the word and the value is a list of all the anagrams to the sorted word. However a dictionary does not keep track of the order of its children and this dictionary is therefore not optimal for sorting and printing out the anagrams. This dictionary of anagrams is therefore transformed into another dictionary that instead holds the length of each anagram list as each keys. The keys

are then stored in a list that is sorted and using that list a sorted list can be printed from the dictionary. Using a list to keep track of the keys for the dictionary combines the structure of the list and the dictionary in a way that solves this problem in an efficient flexible way.

Problem 3

The data structure used for this problem is a plain list. Since there are no iterations or manipulations needed a list of words is a simple way to store the words. It also makes it easy to pick a random word from the list by just generating a random number in the range of zero to the length of the list.