Dr.Erdem KAYA
ekaya@metu.edu.tr
Mehmet Taha ŞAHİN
staha@metu.edu.tr

Assignment 03

METU CENG310 Spring 2021
Data Structures and Algorithms with Python

Start Date: June 3rd, 2021
Due Date: June 25th, 2021

1 Spell Checker

In this assignment, you are expected to implement a spell checker class (`SpellChecker`) which is described as follows.

The spell checking operation is basically a look up operation of a typed word in a dictionary where correct words are stored. If the typed word is not found in the dictionary, then a trial set for the typed word is created and for each trial string, a check is made from the dictionary. Hopefully, a subset of these checks will result in a hit in the dictionary and returned as recommended a word to the user.

To put it more formally, let the word typed by the user be w, and the correct version of that word be v. Let the trial set for w be S^w, and the dictionary is D. The spell checking operations should be performed as follows:

1. In the dictionary D, if the item $D[w]$ exists, then this means $w = v$ and spelling is correct.
2. Otherwise, a set of strings derived by making use of w is created (S^w). For each of the string s in S^w, if $D[s]$ exists, then s will be reported as a recommendation.

Editing is a crucial functionality supported by almost all operating systems and office applications. Such editing applications are expected to be very responsive, due to that reason, the spell checking operation should take place very quickly. As a result, we will be using a hash table for the dictionary so that checking each of the items in the trial set can be finished in $O(|S^w|)$. In this hash table, we will be using the words as the keys, and a negligible value (such as `None` or `object()`) as the value in the dictionary as we will not be using them.

1.1 The Dictionary

As the basis for the spell checking functionality, we will use the most common 10.000 words of English (https://github.com/first20hours/google-10000-english). In order to make life easier, we will remove the words that are 5 characters or less long from the dictionary, which will leave the dictionary with 6399 words. For the dictionary, the classes `ChainHashMap` or `ProbeHashMap`, whose implementations are provided in the textbook, can be used. After creating an instance of the dictionary (a hash table instance), all the words having six characters or more from the aforementioned link will be inserted to make the dictionary ready for spell checking.

1.2 The Trial Set

In case of $w = v$ evaluating to false, we will need to create a trial set S^w in hopes of finding some recommendations as a replacement for w. In order to derive new strings to try against the dictionary, your imagination may lead to very interesting approaches but for the sake of simplicity, we recommend you to use the following two approaches:

1. Insert a single character in between two adjacent characters of w. This operation must yield $26(|w| - 1)$ derived strings.
2. Drop a single character from w. This operation should result in $|w|$ derived strings.

For example let w be the word `inerest`, then some of the items in set S^w would be `ianerest`, `ifnerest`, `incerest`, `interest`, `nerest`, `ierest`, etc.

1.3 The `SpellCkecker` Class

The `SpellChecker` class will have a public method called `check` which will take a file name string as its sole parameter and create an output file with a list of spell checking recommendations. At each line of the output file, there has to be one of the input word followed by one of the following:

1. The word 'OK' in case of an exact match (please safely assume that all the input words will all be lower case), or
2. At most two recommended words in case of no matches, or
3. The phrase 'No Recommendation' when there is no recommendation for the input word.

An example input file content:

```
murphys online murphy
product system
policy number please
available
copyright
```

An example output file content:

```
murphys --> murphy
online --> OK
murphy --> OK
prduct --> product
sstem --> system
policy --> OK
numbre --> No Recommendation
please --> OK
navailable --> available, unavailable
copyright --> OK
```

2 Delivery Instructions

Please hand in your module as a single file named as `spellchecker.py` over ODTUClass by 11:59pm on due date. An Assignment-03 page will be generated soon after the start date of this assignment. Should you have any questions pertaining to this assignment, please ask them in advance (rather than on the due date) for your own convenience. Whatever IDE you use, you have to make sure that your module could be run on a Python interpreter:

```
sc = SpellChecker()
sc.check('Assignment3Input.txt')
```