IFT 2015

TP 1

January 22, 2014

( To be submitted before midnight on February 9 )

The objective of this TP is to familiarize yourself with both Python and give you a first insight of the importance of data structures. To do this, you will implement 3 classes for storing words and compare the effectiveness of three methods; are inserting, deleting, and research, when implanted in the following three structures:

1 . A class “liste” that will use an instance of the class “list” of Python ;

2 . A class "ListeTriee" which will use an instance of the "list" class, but maintain sorted (in ascending lexicographical order) and use a binary search (which you must implement);

3 . A dictionary class that uses an instance of the class Python “dict” ( hash table ) .

The shells of these three classes is provided to you. The \_\_ str\_\_ function should return a list of tuples (key, tally) words contained in the object, as str. You will also create a class “Mot”. “Mot” will contain a str cle, as well as a int to tally the number of times that the word is present in the structure. You must also implement in the class “Mot” the methods “incrimenter”, “decrementer”, “get\_compte”, and “get\_cle”. The constructor of your “Mot” class, when it does not receive a key parameter, will generate a random one, for example by dipping a randomly selected number of randomly letters from a “str” as “abcdefghijklmnopqrstuvwxyz.” You also need to over rider the methods "\_\_eq\_\_" and "\_\_lt\_\_" for they compare their argument with the "key". A shell of your class “mot” is also provided.

While your classes "Liste" and "ListeTriee" will store "Mot" in an instance of the "list" Python class, your class "Dictionary" store them in an instance of "dict", which consists of a hash table. In addition, while methods "inserer", "supprimer" and "trouver" your class list may use the methods provided by the "list" class (and similarly for your class dictionary), your class “ListeTree” will implement binary search. One would therefore expect that some methods of your class "ListeTriee" are more effective than those of your class "List", and some methods of your class "Dictionary" are even more effective. This is what you will test empirically.

Test

(1 Point)

Your files must run under Python 3, no error message

(3 points)

A validation function (unit testing) is at the end of each module of your classes. It tests the validity of your implementations using test1.txt file and two other files. Txt provided. When you drive one of your classes, for example python3 Liste.py, "No huge bug detected." should appear. After delivery, your scripts will be tested on another text file, and possibly with a different validation function that will test your code more thoroughly.

(6 points)

Create an instance of each of your 3 classes, using the shell TP1.py. For each instance, generate different numbers of words (eg powers of 10) with Mot () (that is to say, without argument, that the constructor generates randomly), and insert them into the instance. To generate the same words for the 3 classes, do not forget to re-seeder your generator (with the same root) for each of three instances (eg random.seed (1234)). You must calculate the time taken with each of these insertions 3 structures, using time.time () (do not forget to import the random module and time).

Similarly, calculate the time taken deletions and research in each of the three structures. Produce curves of the three methods in the three structures (one graph per method, 3 curves graphically for three structures). The axis of the number of operations should be log. What do you conclude?

3 points will be awarded for 3 graphics and short conclusions, and 3 points for the quality of the code. The quality of the code includes the efficiency of your code. It is important that your graphs clearly show the execution time in order to properly measure the effectiveness of your code. Avoid several times consecutively to the same operation (search by example).

You must submit your files. Py (executables under Python 3), and a. Pdf file describing / explaining briefly the results / conclusions of your analysis. Put them in Studium as a compressed folder.

Addendum

Please note that the methods “inserer” and “supprimer” of your class “Liste”, “ListeTriee” and “Dictionairre” must increment or decrement the account of a word if present (and delete it if the count drops to 0). Furthermore, these two methods, and the method “Trouver”, must return True if the word is present, or false otherwise. You also need to implement a method “get\_mot” that returns the “Mot” if present or None otherwise.