Projet Multiprocessing : Anagrammes

**Question 1 : Ecrire une première version naïve, la plus simple possible. (cf main\_naif.py)**

De façon naïve, on parcourt pour chaque mot tout le dictionnaire afin de trouver les anagrammes (complexité en O(n²)). On utilise une structure de listes et de dictionnaires. C'est beaucoup trop lent.

**Question 2 : Réfléchir à des optimisations structurelles capables de réduire la complexité de l’algorithme. Réfléchir aussi à des optimisations plus locales, toujours intéressantes pour gagner du temps ou de la mémoire.**

On peut songer à utiliser des ensembles, qui sont plus efficaces que les listes. Aussi, on peut stocker chaque mot dans un dictionnaire dont la clé est composée des lettres du mot trié par ordre alphabétique. Ainsi, tous les anagrammes auront la même clé car ils sont constitués des mêmes lettres, et ils seront donc dans une même liste. Par exemple, les mots "car" et "rac" seront tous deux situés dans le dictionnaire à la clé "acr".

**Question 3 : Paralléliser l’algorithme et l’implémenter à l’aide du module multiprocessing.**

On constate une bonne amélioration avec toutes ces optimisations et l’implémentation du multiprocessing. On arrive à un temps d’exécution de l’ordre de 0.8 seconde avec 4 processus.