Rapport développeur du projet ReadOrganizer

Organisation du code :

Le projet s'articule autour de 5 fichiers dont 3 déjà préparés pendant l'enseignement. Voici la liste des fichiers utilisés lors de ce projet :

- BTW_package.py
- tools_karkkainen_sanders.py
- read.py
- ReadOrganizer.py

Les fichiers BTW_package.py et tools_karkkainen_sanders.py sont utilisés pour générer la table des suffixes, la transformée de Burrows-Wheeler, le dictionnaire d'occurrence et la liste des rangs pour chaque position de la transformée.
Les deux nouveaux fichiers sont read.py et ReadOrganizer.py. Le fichier read.py permet de faciliter la manipulation des reads en donnant des informations sur la position de ce read dans le génome et un état "mapped" qui indique que le read a déjà une position qui lui est attribuée.
ReadOrganizer.py est le fichier principal, il contient le main() et c'est lui qui permet de lancer le script. Il est composé de deux méthodes de mapping, une fonction pour trouver la meilleure position du read parmis toutes les seeds, et une fonction pour réorganiser les différents reads dans un ordre facilitant la compression par gzip.

Présentation de l'exécution du main() :

1. Initialisation

Dans un premier temps, on lit le fichier contenant le génome de référence qui servira de guide, on génère la table des suffixes, la transformée de Burrows-Wheeler, le dictionnaire d'occurrence, la liste des rangs pour chaque position de la transformée ainsi que la séquence reconstruite du génome. On lit ensuite le fichier de reads au format FASTA et chacun d'entre eux est récupéré dans un objet Read.

2. Mapping

Le mapping des reads s'ensuit avec deux méthodes possibles selon l'option "-m" choisie au lancement du script : la méthode 1, celle de l'énoncé, et la méthode 2 qui est notre proposition d'une autre approche.
La méthode 1 explore chaque sous-séquence k pour chaque read et cherche toutes les occurrences de ces sous-séquences dans le génome. Elle stocke toutes les positions des occurrences dans un ensemble afin de retirer les doublons.
La méthode 2 a une approche de comptage d'occurrences, au lieu de stocker chaque occurrence, elle les compte pour chaque position possible dans le génome. Un dictionnaire est créé avec pour clés : les positions relatives dans le génome et pour valeurs le nombre d'occurrences à cette position.

3. Calcul du meilleur alignement

La méthode 1 se poursuit avec la fonction best_position_m1(). Cette fonction utilise l'ensemble des occurrences pour comparer le read au génome. Le nombre de substitution est le critère de sélection du meilleur alignement.

La méthode 2 a elle aussi sa fonction best_position_m2(). Elle se base sur la fréquence des occurrences récupérées dans le dictionnaire. Le meilleur alignement sera le read avec la position la plus récurrentes.

Une fois la meilleure position trouvée, le read est alors considéré comme mappé.

Le même procédé de mapping se passe pour les séquences dites "reverse complement".

4. Tri et sortie

Tous les reads sont ensuite triés via la fonction sort par défaut dans python. L'ensemble des reads triés sont alors stockés dans un fichier de sortie. Le nom de ce fichier est configurable avec le paramètre "-o" lors du lancement du script.