BIF
Rapport développeur

GUTIERREZ-ANDRE Alban

HULOT Valentin

/!\
Toutes les fonctions ont été commentées afin de faciliter la compréhension.
Pour plus d'informations, cf. code.
/!\

Organisation du code

Par soucis de lisibilité et d'organisation, le code à été séparé dans plusieurs classes et fichiers. Ce sont les suivants :

- **argParser.py**
  Contient la fonction *argParse()* permettant de récupérer chaque argument passé en entrée du programme

- **dmLinearMem.py**
  Contient la classe et les méthodes permettant de créer une matrice et d'effectuer un alignement semi-global.
  Suite à l'utilisation d'une matrice de seulement 2 lignes pour l'optimisation, l'index de début de mot est perdu.
  Pour régler ce problème, une classe supplémentaire MatrixValue permet d'affecter un indice de début de mot et une valeur à chaque case de la matrice.

- **karkkainan.py**
  Bibliothèque fournie en TP permettant de créer et trier un tableau des suffixes pour une chaîne de caractères donnée.

- **minimapper.py**
  Fichier contenant le main du projet minimapper, ainsi que quelques fonctions annexes, notamment celles permettant de lire les fichiers en entrée.
  (Main détaillé plus loin).

- **reference.py**
  Classe représentant le génome de référence, permettant de l'indexer, de le sauvegarder et de l'importer depuis un fichier.

- **timer.py**
  Classe permettant de gérer le temps d'exécution d'une fonction.

Enchaînement des divers appels et fonctions clefs.

Le main de minimapper.py permet de réaliser les alignements, ainsi que d'afficher la progression dans la console. Voici les différentes étapes:

1. **Obtenir les arguments du programme**
   (via la méthode argParse de argParser.py)

2. **Lire le génome et l'indexer**
   Si le génome à déjà été indexé, le récupère depuis le fichier "*.dumped.gz".
   Sinon l'index puis le sauvegarde sur le disque.

3. **Initialiser les streams d'entrée et de sortie**
   Créer un inputStream sur le fichier reads et récupérer le premier
   Créer un outputStream pour le fichier de sortie.

4. **Lance le timer (via Timer.py)**

5. **Boucle d'alignement**
   a. **Aligne le read sur le génome de référence précédemment indexé.**
      Appelle la fonction *getBestSemiGlobalAlg(),* qui permet de créer toutes les seeds, les aligner exactement sur la référence, puis aligner semi-globalement le read sur la référence. Renvoi ensuite le meilleur score et la meilleure position.

   b. **Idem avec le reverse complément du même read.**
      Obtiens un autre couple (bestScore, bestPosition)

   c. **Vérifier les résultats**
      Vérifie si un résultat est obtenu et quel est le meilleur alignement entre le read et son reverse complément.
      Puis l'ajouter au fichier de sortie.

   d. **Obtenir le prochain read et recommencer**
      (tant qu'il en reste)

6. **Fin Timer**
   Afficher le temps, fermer les streams et afficher les résultats obtenus