Option Bioinformatique L3
TP6

2016

Le support de cours peut être consulté à cette adresse:

https://docs.google.com/presentation/d/1AswTLepd2WwFNf9nEwN3sHFmgplQiHcoWY-Nfpym5nM/edit

Le code source réalisé pendant ce TP sera ajouté au code source des TPs précédents. A la fin de tous les TPs, le résultat final sera un seul outil, “bioseq”, comprenant les fonctionnalités implémentées dans chaque TP. Il est à rendre, le **18/03 (avant 23h59; pénalités si rendu après..)**, à l’adresse suivante:

rayan.chikhi@univ-lille1.fr

avec le sujet de mail suivant (remplacer Nom1, Nom2 par chaque nom de famille; monomes autorisés):

[BI] [TP] Nom1 Nom2

Il est demandé de, soit, créer une archive .zip ou .tar.gz se décompressant dans un repertoire “BI-Nom1-Nom2”, et d’attacher cette archive en pièce jointe à l’email; soit de créer un repository Github (au nom de votre programme principal, par exemple “bioseq”) et d’envoyer le lien par email en précisant les noms du binome/monome. Dans ce repertoire (ou dans le repository), un fichier script nommé **execute-tp6** exécutera (et auparavant, compilera si besoin) tout ce qui est demandé dans les sections de ce TP nommées “Pour le compte-rendu [..]”. De telle manière qu’exécuter:

./**execute-tp6**

dans le répertoire BI-Nom1-Nom2 suffira à l’évaluateur du TP pour tester tous vos programmes. **Ce script (ainsi que le code source de votre programme) fera office de compte-rendu du TP.**

Détaillons un peu comment organiser le script de compte rendu. Par exemple, pour ce TP, le script execute-tp6 affichera quelque chose comme:

```
$./execute-tp6

Partie 1:

[les 10 premieres valeurs du tableau de suffixes de L. phage]
[les 10 premièrs caractères de la BWT de L. phage]

Partie 2:

“Found” [si GGCGCGCAGTGACACTGCGCTGGATCGTCTGATG est trouvé dans L. phage, sinon “Not found”]
“Found” [ou “Not found”]
```

En guise d’exemple, si vous utilisez Java, le contenu de execute-tp6 pourra ressembler à:

```
#!/bin/bash

echo “Partie 1”

java Bioseq suffix-array lambda_phage.fasta | cut -d\  -f 1-10
java Bioseq bwt lambda_phage.fasta > lambda_phage.bwt
cat lambda_phage.bwt | cut -b 1-10

echo “Partie 2”

java Bioseq match-bwt lambda_phage.bwt GGCGCGCAGTGACACTGCGCTGGATCGTCTGATG
java Bioseq match-bwt lambda_phage.bwt AGGCGCGCAGTGACACTGCCGCTGGGATCGTCTGATG
```

Rappel, pour rendre execute-tp6 exécutable, tapez:

```
chmod +x execute-tp6
```

De telle sorte que la commande suivante permet d’executer le script:

```
./execute-tp6
```

L’écriture de scripts bash est une compétence quasi-indispensable en informatique.. Pour en savoir plus, il existe de nombreux tutoriels, dont celui-ci: http://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux/introduction-aux-scripts-shell

Note: ne pas rendre ce TP seul. L’ensemble des TPs sera à rendre, d’un seul bloc (car il s’agira d’un seul code source commun), lors du dernier TP.

Les langages acceptés sont: **Java, C, C++, Python**

Partie 1: Construction du tableau des suffixes et de la BWT pour de plus grands génomes

Dans les TPs précédents, vous avez réalisé des fonctions `suffix-array` et `bwt` de votre outil bioseq.

Dans ce TP, il vous est demandé d'améliorer ces fonctions pour pouvoir construire le tableau des suffixes et la BWT pour de plus grands génomes que des virus. Pour cela, il sera nécessaire de ne pas représenter tous les suffixes explicitement. Par exemple, pour construire le tableau des suffixes, triez le tableau [1, 2, …, |s|] (correspondant aux positions de départ de chaque suffixe) avec une fonction de comparaison personnalisée (pour Java, voir technique vue en cours). Vous pouvez sauter cette partie si, lors des TPs précédents, vous avez déjà implémenté une telle technique, ou si votre propre technique permet de calculer la BWT pour la bactérie S. thermophilus.

Améliorer les fonctionnalités **suffix-array** et **bwt** en conservant la même ligne de commande:

```
./bioseq suffix-array <nom du génome g>.fasta
./bioseq bwt <nom du génome g>.fasta
```

(Remplacez les anciennes fonctions `suffix-array` et `bwt` du TP1 et TP2, vu que les nouvelles fonctions que vous écrirez auront les mêmes fonctionalités, "seulement" plus efficaces. Si vous le souhaitez, vous pouvez conserver vos anciennes fonctions sous une forme telle que `suffix-array-naive` et `bwt-naive`)

Appliquez ces fonctions aux génomes suivants:

- Virus Lambda phage: voir TP1 pour le téléchargement
- Bactérie S. Thermophilus: http://www.ncbi.nlm.nih.gov/nuccore/NC_006448.1
- (facultatif) Chromosome humain 14: http://www.ncbi.nlm.nih.gov/nuccore/NC_000014.8
- (très facultatif) Génome humain complet: http://biyoenformatik.blogspot.fr/2014/04/download-human-reference-genome-hg19.html (nécessite probablement une machine à haute mémoire, telle que "bacchus", c.f. l'affiche décrivant les serveurs sur le mur de la salle A11)

Pour le compte-rendu, retourner les 10 premiers indices du tableau des suffixes de Lambda phage sur une seule ligne, puis les 10 premiers caractères de la BWT de Lambda phage sur une seule ligne. Pour ce faire, incluez le génome de L. phage dans le répertoire du code source, ou faites-le télécharger par le script execute-tp6. Facultatif: si vous avez réussi pour le S. Thermophilus, chromosome 14 ou le génome humain, recopiez seulement le résultat (les 10 premiers indices du tableau des suffixes sur une ligne, et 10 premiers caractères de la BWT sur une ligne) dans execute-tp6 (ne pas inclure de fichier FASTA d'un génome de plus de 1 Mbp dans le compte rendu TP, à cause de la trop grande taille des données).

Optionnel: afficher les temps d'exécution.

Partie 2: Recherche dans la BWT

Ecrire une fonctionalité **match-bwt** s'exécutant de la manière suivante:

`./bioseq match-bwt` `<nom du fichier>.bwt <séquence de la lecture r>`

qui implémente l'algorithme de recherche d'une sous-chaîne r dans la BWT d'un génome, vu en cours.

match-bwt affichera "Found" si la séquence est trouvée, et "Not found" sinon.

Exemple: "`./bioseq match-bwt`" appliqué à un fichier BWT de la séquence ACTTGATT et recherchant la séquence TTG affichera:

Found

Pour le compte-rendu, appliquer match-bwt à la recherche de deux sous-chaînes:
GGCGCGCAGTGACACTGCGCTGGATCGTCTGATG
et AGGCGCGCAGTGACACTGCCGCTGGGATCGTCTGATG dans le virus Lambda phage.

Indication: si votre algorithme prend trop de temps, il peut être intéressant de précalculer C(c) et rank(i,c) pour toutes les valeurs possibles de i et de c, et les stocker dans un tableau.

Optionnel: afficher le temps d'exécution.

Pour aller plus loin (facultatif): créer une fonction "search-bwt" qui affiche la position dans la chaîne de départ de chaque occurrence trouvée de la sous-chaîne.