Algorithmique et bioinformatique

Enoncé du projet

Université de Mons - Année académique 2020-2021

Enseignants : O. Delgrange & C. Tamines

1 Objectifs du projet

A partir d'une collection de fragments (chacun d'une longueur variant entre 500 et 700 pb), il vous est demandé de concevoir un programme d'assemblage de ceux-ci, qui utilise l'approximation de type Greedy et l'alignement semi-global et qui fournit une séquence cible attendue. Les objectifs du projet sont donc :

- Avoir un programme fonctionnel qui assemble une collection de fragments en une unique séquence et la comparer avec la séquence initiale. Plus de détails seront fournis dans la suite de l'énoncé.
- Appliquer les notions d'algorithmique vues lors de votre cursus.
- Appliquer les bonnes pratiques de programmation.
- Optimiser votre code face à de grosses données à traiter.
- Gérer votre temps et organiser les tâches à effectuer dans votre groupe.

2 Consignes

2.1 Informations générales

- Le projet se fera par groupe de 2 étudiants. Veillez à répartir correctement le travail entre chaque étudiant. En cas de nombre impair d'étudiants, un étudiant formera à lui seul un groupe et aura un projet simplifié. Un numéro sera attribué à chaque groupe.
- Les notes théoriques sur l'assemblage de fragments sont disponibles sur la plate-forme moodle.
- L'implémentation du projet se fait en *Java*.

2.2 Projet

Comme dit précédemment, le but du projet est de concevoir un programme qui, à partir d'une collection de fragments, renvoie une séquence unique issue de l'assemblage de ces fragments à l'aide de plusieurs techniques. La ressemblance des fragments se détecte grâce à l'alignement semi-global vu au cours théorique, et l'assemblage des fragments se fait grâce à une approximation de type Greedy. Il vous est ensuite demandé de comparer votre séquence obtenue avec la séquence cible. Plus votre séquence obtenue y ressemble, mieux sont vos résultats.

Dès lors, afin de tester la qualité de vos résultats, l'utilisation de l'outil *dotmatcher* (outil de dotplot de deux séquences) est demandée. Cet outil fonctionne en ligne de commande et est disponible :

- sous linux : télécharger et installer le package **emboss** (aussi disponible dans les dépôts) ;
- sous Windows : téléchargeable à cette adresse : `http://emboss.sourceforge.net/download/`.

L'utilisation en ligne de commande est la suivante :

```
dotmatcher SEQUENCE_1 SEQUENCE_2 -threshold 50
```

où :

- `SEQUENCE_1` est le nom du fichier contenant votre séquence résultat ;

— `SEQUENCE_2` est le nom du fichier contenant la séquence attendue ;
— `-threshold 50` est un paramètre nécessaire pour obtenir une bonne visualisation du résultat.

Une démonstration de *dotmatcher* sera réalisée lors de la première séance de travaux pratiques.

Voici quelques consignes et informations pratiques.

— Quatre collections de fragments, ainsi que les séquences dont ils ont été pris, sont disponibles sur la plate-forme moodle [1].
— Les fragments sont donnés dans le format fasta : la séquence, sous forme de lignes de 80 caractères maximum, est précédée d'une ligne de titre qui doit commencer par le caractère '>'. Plusieurs séquences peuvent être mises dans un même fichier.
— Les complications prises en compte lors du séquençage des séquences sont : les erreurs de séquençage et l'orientation inconnue.
— Les séquences cibles fournies par votre programme seront données dans le format fasta. La ligne de titre sera formatée comme suit :

```
> Groupe-num_groupe Collection num_collection Longueur longueur_sequence_cible
```

— Dans le cas où un élève forme à lui seul un groupe, les collections à considérer sont les collections simplifiées où la complication de l'orientation inconnue est relaxée (i.e. l'orientation des fragments est connue, la même pour tous).
— Lors du dépot de votre projet (davantage d'informations ci-dessous), il vous sera demandé de founir un fichier .jar pour exécuter votre programme, nommé "FragmentAssembler.jar" et pouvant être lancé en utilisant la commande "java -jar FragmentAssembler.jar <fichier.fasta> –out <sortie.fasta> –out-ic <sortie-ic.fasta>", où "<fichier.fasta>" (resp. "–out <sortie.fasta>" "–out-ic <sortie-ic.fasta>"), est le chemin vers le fichier fasta d'entrée (resp. fasta résultant créé, fasta résultant inversé et complémenté créé).

2.3 Dépôt du projet

Voici les consignes relatives au dépôt du projet. La date limite de cette remise est donnée plus loin dans l'énoncé de ce projet. **Tout plagiat (relatif au code ou au rapport) sera évidemment lourdement sanctionné.**

- *Code* : vous déposerez via la plate-forme e-learning une archive (zip ou tgz), nommée "*Groupe<votrenumero>-<Nom1>-<Nom2>*", contenant
  — les fichiers .java.
  — les fichiers .fasta reprenant les séquences cibles demandées.
  — le fichier .jar permettant de lancer votre programme.
- *Rapport* : vous déposerez via la plate-forme e-learning un PDF de votre rapport. Celui-ci, nommé "*Groupe<votrenumero>-<Nom1>-<Nom2>*", contiendra :
  — La répartition des tâches au sein du groupe.
  — Une explication de chaque étape de votre démarche (approche, optimisation, changements apportés, ...).
  — Les points forts, les points faibles et les erreurs connues de votre programme.
  — Une interprétation des résultats obtenus.
  — Une conclusion comprenant une réflexion sur le projet (apports, difficultés rencontrées, ...).

Notez que le document « Eléments de rédaction scientifique en informatique » est mis à votre disposition par Mr. Mélot reprenant les bonnes pratiques de rédaction.

1. `https://moodle.umons.ac.be/course/view.php?id=138`

3 Dates importantes

- **mardi 29/09/2020 : composition des groupes**
  Pour cette date, vous aurez communiqué la composition de votre groupe à Clément Tamines (`clement.tamines@umons.ac.be`).
- Votre présence sera **obligatoire** lors de la séance du jeudi **05/11/2020 à 13h30**. Un contrôle de l'avancement du travail y sera effectué.
- **Vendredi 18/12/2020 à 16h** : remise des fichiers du code et du rapport.
- **Evaluation orale** : Une évaluation orale du travail sera réalisée pour chaque groupe. Un horaire vous sera communiqué par la suite.

4 Evaluation du projet

4.1 Conditions d'acceptation du projet

Certaines consignes ont été spécifiées lors de cet énoncé :
— Code fonctionnel pouvant être lancé à partir d'un .jar FragmentAssembler.jar à l'aide de la ligne de commande précédemment donnée.
— Respect du format utilisé pour les .fasta.
— Respect des consignes de dépôt de l'archive contenant le code et du rapport.
— Respect de la date limite de dépôt.

Ces consignes sont strictes et doivent **toutes** être respectées. **Un projet non recevable n'est pas corrigé et entraîne une note de 0/20**.

4.2 Critères d'évaluation

Rappelons tout d'abord que votre note finale est la moyenne de votre note à l'examen théorique (**T**) et de votre note du projet (**P**), à moins qu'une de ces deux notes soit inférieure à **8/20**, dans quel cas, votre note est le minimum de vos deux notes. Donc, votre note est calculée de la manière suivante : $\text{Note} = \begin{cases} \min(T, P) & \text{si } T < 8 \text{ ou } P < 8 \\ (T+P)/2 & \text{sinon} \end{cases}$.

Dès lors, ne sous-estimez pas le travail à fournir et ne négligez pas le projet. Les critères d'évaluation de votre projet sont les suivants :
— La qualité de vos résultats et la manière de les obtenir.
— L'efficacité du programme.
— La qualité du code.
— La qualité du rapport.
— Votre défense orale.

5 Remarques

— Si besoin, vous pouvez demander à avoir un accès à une machine de l'université afin de tester votre programme sur des entrées volumineuses. Ceci est prévu au cas où vous disposeriez d'une machine de faible puissance.
— Si vous avez des questions, pendant toute la durée du projet, vous prendrez contact par e-mail avec Clément Tamines (`clement.tamines@umons.ac.be`).