Master de Bioinformatique
de Nantes Université
2022– 2023

Advanced algorithmics and programming for biologists /
Models, methods and algorithms for bioinformatics

Sujet de Travaux Pratiques
Encadrement C. Sinoquet

Distribué le vendredi 07 octobre 2022.
Date de contrôle intermédiaire: mardi 08 novembre 2022 (Génération des jeux de données)
Date limite de remise : mardi 13 décembre 2022 (démonstration en salle sur matériel sur lequel votre TP a été réalisé (portable admis)).
Pénalité de 1 point par jour de retard, ouvré ou non.
Aucun devoir rendu en janvier 2022 ne sera évalué (notation 0).

Sujet à traiter en langage C++ (et R ou Python pour les scripts, et **seulement pour les scripts**[1]), sous Linux.
Sauvegardes nécessaires en cours de travail, sur divers supports.
Sauvegardes nécessaires de versions partielles en cours de travail, sur divers supports.
(Aucun délai ne sera accordé si le projet a été perdu, faute de sauvegarde).

Alignement multiple de traces d'événements

Chaque binôme d'étudiants devra implémenter
- une méthode de génération de jeux de données de complexité contrôlée
- une méthode d'alignement de traces d'événements, suivant le schéma expliqué en cours,
- le protocole d'évaluation de cette méthode.

1) Algorithme d'alignement de traces

Deux fichiers d'entrée sont requis, et deux fichiers de sortie seront générés :
- le fichier (.txt) des traces d'événements
- le fichier (.txt) des paramètres propres à la méthode d'alignement, commenté
- le fichier (.txt) de l'alignement obtenu
- le fichier (.csv) des scores permettant d'évaluer la qualité de l'alignement obtenu.

Fichier des traces d'événements (.txt)
format obligatoire (exemple) :

```
E1 . E2 . . E3 . . . E4 . . E5 . . . E6 E7 E8 . E9
E1 . . . . . . . . E4 . . . E5 . . . E6 E8 . . . E9
E1 . . . . . E3 . . E4 . . E5 . . . E6 . . . E8
E2 . . E3 . . E4 . . . E5 . . . E6 . E8 . . E9
```

[1] Dans le contexte de ce projet, un script est un code destiné à lier des unités de code avancé entre elles, ou à mettre en œuvre des conversions de fichiers d'un format dans un autre.

N.B. 1 : Les traces n'ont pas nécessairement la même longueur.
N.B. 2 : Le séparateur est l'espace, obligatoirement.
N.B. 3 : Chaque ligne du fichier correspond à une trace.
N.B. 4 : Il n'y a pas de « dernière ligne vide » (i.e., correspondant à un symbole de fin de ligne supplémentaire).

Fichier des paramètres propres à la méthode d'alignement, commenté (.txt)
format obligatoire (exemple) :

<description of first parameter>
<possibly spread on several lines>
<identifier of first parameter> <value of first parameter>

<description of second parameter>
<possibly spread on several lines>
<identifier of second parameter> <value of second parameter>

…

<description of last parameter>
<possibly spread on several lines>
<identifier of last parameter> <value of last parameter>

N.B. 1 : Des lignes « vides » séparent les descriptions de paramètres.
N.B. 2 : Il n'y a pas de « dernière ligne vide » (i.e., correspondant à un symbole de fin de ligne supplémentaire).
N.B. 3 : Un identifiant de paramètre est du type (exemple) :
somme_mismatches
N.B. 4 : Un identifiant de paramètre ne comporte pas de majuscules.
N.B. 5 : Un identifiant de paramètre comporte uniquement des « _ » pour séparer des sous-mots.
N.B. 6 : Les identifiants de paramètres sont communs à tous les binômes du M2 Bioinformatique / parcours Bioinformatique pour les Biologistes, et sont *a fortiori* communs entre les deux variantes d'un même binôme.
N.B. 7 : Un paramètre réel se note par exemple 3.14116 et non pas 3,14116.
N.B. 8 : Les chemins de noms de fichiers entrée et sortie ne sont pas indiqués dans le fichier des paramètres, mais sont indiqués dans le script de lancement de la méthode.

Fichier d'alignement obtenu (.txt)
format obligatoire (exemple) :

```
E1 . E2 . . - E3 . . . E4 . . - E5 . . . E6 E7 - - E8 . - - E9
E1 . .  . . - .  . . . E4 . . . E5 . . . E6 -  - - E8 . . . E9
E1 . .  . . . E3 . . - E4 . . - E5 . . . E6 .  . . E8 - - - -
E2 . .  - - - E3 . . - E4 . . . E5 . . . E6 -  - . E8 . . - E9
```

Fichier des scores permettant d'évaluer la qualité de l'alignement obtenu (.csv)
format obligatoire (exemple) :

<identifiant score N°1> <identifiant score N°2> … <identifiant dernier score>
<valeur identifiant score N°1> <valeur identifiant score N°2>… <valeur identifiant dernier score>

N.B. 1 : Un identifiant de score est du type (exemple) :
somme_mismatches
N.B. 2 : Un identifiant de score ne comporte pas de majuscules.
N.B. 3 : Un identifiant de score comporte uniquement des « _ » pour séparer des sous-mots.

Fichier readme.md
Vous fournirez un fichier readme.md (extension obligatoire) pour votre algorithme d'alignement (installation, information sur les librairies requises, compilation, référence au répertoire de l'exemple jouet (nom obligatoire : toy_example) incluant le fichier de paramètres modèle renseigné avec des paramètres par défaut.).

En plus du fichier readme.md, vous fournirez un script Python launch_<method>_toy_example.py dédié à l'exécution de <methode>_<variante> sur l'exemple jouet (script à inclure dans le sous-répertoire toy_example).

Script de lancement
Le script de lancement de la méthode, intitulé launch_<methode>_<variante>.py prendra en paramètres :
- le chemin du fichier des traces d'événements,
- le chemin du fichier des paramètres propres à la méthode d'alignement,
- le chemin du fichier d'alignement à générer,
- le chemin du fichier des scores à générer.

2) Evaluation de la méthode d'alignement

Chaque binôme calculera au moins 5 scores (obligatoires, ci-dessous) par alignement multiple pour les 20 alignements multiples obtenus à partir de 20 jeux de données correspondant à une même complexité donnée, et indiquera la durée d'exécution pour chaque alignement multiple :

L'en-tête du fichier .csv résultant sera :
identifiant_fichier score_nw score_e match_e score_g proj_length rtime

avec :

identifiant_fichier : nom du fichier du jeu de données (sans le chemin complet !!!, exemple : traces_80_50_hard_i.txt)

score_nw : score Needleman-Wunsch

score_e : scores « events »[2]

[2] si une colonne de l'alignement multiple comporte au moins deux occurrences d'événement, on comptabilise le nombre d'occurrences de cet événement sur toute la colonne. Le score « events » est obtenu par sommation des résultats obtenus sur toutes les colonnes concernées.

match_e : scores match_events pour toutes les paires de projections.

score_g : comptage du nombre de gaps

proj_length : longueur commune des projections constituant l'alignement multiple

3) Génération des jeux de données de complexité contrôlée

Chaque jeu de données sera généré à partir d'un fichier spécifiant une expression, le nombre de traces à générer et leur longueur maximum. Un exemple simple d'expression est présenté en Exemple 1.

```
     0  5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 110 temps
T1N  E1 - -  E2 .  .  E3 -  .  .  .  E4 .  .  .  .  .  .  E7 .  .   S
T2N  E1 . .  E2 .  .  .  -  .  .  _  E4 .  .  .  .  .  -  E7 .  -   S
T3N  E1 - -  -  -  -  E3 -  .  .  .  E4 .  .  .  .  -  -  E7 -  -   S
T4N  E1 - -  -  -  -  E3 -  E2 .  -  E4 .  .  .  -  -  -  E7 -  -   S

T5N  E1 . .  E2 E8 .  E3 -  .  .  .  E4 .  .  .  .  .  .  E7 .  .   S
T6N  E1 . .  .  .  -  E3 E8 E2 .  .  E4 .  .  .  .  .  .  E7 -  -   S
T7N  E1 . .  .  -  -  E3 -  E2 E8 -  E4 .  .  .  .  .  .  E7 .  .   S
T8N  E1 . .  .  .  -  E3 -  E2 E8 E2 E4 -  -  -  -  -  -  E7 -  -   S
T9N  E1 . .  -  E8 -  E3 -  E2 E8 E2 E4 -  -  -  -  -  -  E7 -  -   S

T10N -  - -  -  -  -  -  -  -  -  -  - E4 .  E5 .  E6 .  .  E7 .  -   S
T11N -  - -  -  -  -  -  -  -  -  -  - E4 -  E5 .  .  -  -  E7 .  .   S
T12N -  - -  -  -  -  -  -  -  -  -  - E4 .  -  .  E6 .  .  E7 .  -   S
T13N -  - -  -  -  -  -  -  -  -  -  - E4 .  .  _  E6 .  E5 E7 .  .   S

T14N E1 . .  E2 .  . E3  .  .  .  -  E4 -  E5 .  -  .  -  E7 .  -   S
```

Exemple1. Expression <()*> E4 <()*> E7 () S

Les expressions seront générées selon les conventions suivantes :

Ea1 (tr1 - tr2) Ea2 : Entre les événements Ea1 et Ea2, il n'y a jamais d'autre événement, et la durée séparant les horodatages de Ea1 et Ea2 varie dans [tr1 , tr2].

Ea1 <(tr1 - tr2)+> Ea2 : Entre les événements Ea1 et Ea2, il y toujours au moins un événement, et la durée séparant les horodatages de Ea1 et Ea2 varie dans [tr1 , tr2].

Ea1 <(tr1 - tr2)*> Ea2 : Constitue l'union des deux situations précédentes.

Ea1 <(tr1 - tr2) Ei1 | ... | Ei2> Ea2 : l'élément de représentation synthétique de trace symbolisé par <(tr1 - tr2) Ei1 ... Ei2> s'étend sur une durée variant dans [tr1 , tr2], entre les événements Ea1 et Ea2. Un seul des événements Ej ($i1 \leq j \leq i2$) est présent, par trace. Deux traces peuvent partager le même événement dans l'intervalle [tr1 , tr2].

Ea1 <(tr1 - tr2) Ei1%p1 | ... | Ei2%p2> Ea2 : même signification que dans l'expression ci-dessus, avec la précision suivante : p1% des traces comporteront l'événement Ei1, p2% des traces comporteront l'événement Ei2.

Ea1 <(tr1 – tr2) Ei1Xki1 … Ei2Xki2 Ei3 … Ei4> Ea2 : Tous les événements Ej ($i1 \leq j \leq i2$, ($i3 \leq j \leq i4$)) sont présents dans toutes les traces, et aucun autre ; l'événement Ej(kj) ($i1 \leq j \leq i2$) peut apparaître jusqu'à ki1 fois dans chaque trace concernée ; l'événement Ej ($i3 \leq j \leq i4$) apparaît exactement une fois dans chaque trace concernée. L'ordre Ei1 … Ei2 Ei3 … Ei4 est indifférent (entre les traces).

Le logiciel de génération de données data_generation prendra en entrée :
un nom de répertoire de sortie
le nombre de fichiers à générer
le préfixe commun aux identifiants des fichiers
le fichier guidant la génération (expression, nombre de traces à générer, longueur maximum des traces).

Vous fournirez un script Python launch_data_generation_toy_example.py.
Vous fournirez un fichier readme.md.
Vous fournirez un script Python launch_data_generation_<xxx>.py pour chaque type de jeu de données généré.

4) Notation du projet

La notation prendra en compte les points suivants :

- correction des algorithmes,
- lisibilité du code produit,
- modularité du code et des tests,
- preuve de réalisation de tests, production des jeux d'essais et résultats (copies d'écran, fichiers texte – correctement présentés),
- facilité de maintenance du code produit,
- séparation claire des définitions de fonctions par des lignes du type ********** **(OBLIGATOIRE)**,
- présence de commentaires en anglais correct dans les programmes et/ou choix d'identifiants « parlants »,
- *si nécessaire*, mention des préconditions et postconditions, en commentaire, immédiatement après le prototype de la fonction définie,

• livrables à rendre:

- un rapport, d'au plus 4 pages, présentant
  - les structures de données utilisées,
  - un bilan concis récapitulant la réalisation effective par rapport au texte du devoir, et récapitulant ce qui reste à faire (par rapport au texte du devoir), ou au regard des améliorations que vous apporteriez si vous aviez plus de temps,
  - éventuellement, un bilan concis des erreurs que vous n'avez pas réussi à corriger (il vaut mieux le placer ici, plutôt que nous le découvrions nous-mêmes…)

Le rapport devra comporter en annexe les résultats d'exécution de votre algorithme sur plusieurs jeux d'essai significatifs.

- une archive exhaustive de tous les codes sources, portant explicitement le nom des étudiants concernés <nom1_nom2.tar.gz> (en minuscules), à télécharger.

• Un rapport redécrivant les fonctionnalités des diverses fonctions est totalement inutile: tout doit être lisible dans le code source.

• Le nombre de trinômes est limité à 1.

• Une importante modulation selon la part réelle effective apportée au travail en binôme sera éventuellement appliquée.

• Les étudiants seront évalués selon les critères détaillés ci-dessus, ainsi que sur la vérification effective que le logiciel élaboré fonctionne totalement ou partiellement.