Algorithmique et bioinformatique : Rapport de projet

Collin Arnaud, Galina Alicia

December 14, 2016

Contents

1 Introduction

Le but de ce projet est de concevoir un programme d'assemblage de segments données pour fournir une séquence. Pour chaque collection de segments une séquence cible nous est fournie pour nous assurer de l'efficacité de notre programme. Celui-ci sera développé en JAVA et abordera des notions théoriques vues en cours.

2 Explication de la démarche

2.1 Structures de données

2.1.1 CollectionFragments

2.1.2 Graphe

Il représente le graphe de segments avec leurs scores d'alignement. Il contient 2 listes chaînées : une d'objet Node et l'autre d'objet Link. Il implémente aussi un algorithme de tri par tas qui classe les Link par ordre de scores décroissant.

2.1.3 Node

Il représente un noeud du graphe, il contient différentes informations :

- id : l'id du fragment (les fragments sont numérotés dans l'ordre d'apparition du fichier),
- in : un booléen indiquant si nous sommes déjà rentré dans ce noeud (si son côté gauche est libre ou non),
- out : un booléen indiquant si nous sommes déjà sorti de ce noeud (si son côté droit est libre ou non),
- compl : un booléen indiquant si ce segment a déjà été choisi en complémentaire inversé.

2.1.4 Link

Il représente un lien du graphe, il contient différentes informations :

- sourceID : l'id du fragment source,
- destinationID : l'id du fragment destination,
- value : le score d'alignement de ce lien,
- chaineSourceCompl : un booléen indiquant si le segment source est choisi en complémentaire inversé,
- chaineDestinationCompl : un booléen indiquant si le segment destination est choisi en complémentaire inversé.

2.2 Algorithmes

2.2.1 Lecture du fichier fasta

2.2.2 Semi-global

Pour semi-global nous appliquons le même algorithme que vu en cours. Nous devons l'effectuer 2 fois par paire de fragments. En effet, il y a 8 façons d'arranger chaque paire. Il y a 2 formes possible pour chaque fragment : normal ou complémentaire inversé. Ce qui nous donne 4 combinaisons. Pour chaque combinaison, on peut les arranger de 2 manières différentes : segment s suivi de segment t ou l'inverse, ce qui donne également 2 scores différents. Nous avons donc au total bien 4x2 = 8 arrangements.

Pour une combinaison, l'algorithme semi-global nous donnera ces 2 scores. Besoin à priori de faire 4x semi-global. Or il existe des combinaisons qui donnent les mêmes scores :

1. s normal et t normal = s complémentaire inversé et t complémentaire inversé.
2. s normal et t complémentaire inversé = s complémentaire inversé et t normal.

Au final, nous avons bien besoin d'effectuer 2x semi-global pour nos 8 arrangements.

2.2.3 Arrangement des liens

Nous avons vu qu'il existe 8 arrangements. Cependant, nous savons que certains ont le même score. Il est inutile de stocker 8 liens quand seulement 4 pourraient être utilisés. En effet, pour un certain arrangement, nous savons quel autre arrangement lui est égale en score. Ce travail se fera au niveau de l'algorithme Greedy qui regardera quel arrangement peut convenir s'il y en existe un.

2.2.4 Greedy

Nous trions d'abord les liens du graphe par score décroissant. Pour chaque lien, suivant sa catégorie, nous allons vérifier si une façon d'arranger les 2 fragments est acceptable, si oui alors on ajoute ce lien à notre chemin hamiltonien. Lors de cette vérification il faut être prudent car pour chaque catégorie d'arrangements, nous avons 2 scores. Si c'est l'autre facon d'arranger le lien qui doit être prise alors nous créons un nouveau lien avec les bonnes caractéristiques pour le mettre dans notre chemin. Il faut dans ce cas échanger la source et la destination pour avoir le score de même valeur dans cette dispostion.

3 Répartition des tâches

4 Points forts, points faibles et erreurs connues

5 Interprétation des résultats obtenus

6 Conclusion