Algorithmique et Bioinformatique
Assemblage de fragment
Rapport

Brohée Jannou & Ledru Santorin
Groupe XX

May 6, 2016

Faculté
des Sciences

Table des matières

1 Introduction

L'objectif de ce projet est de ....

2 Représentation d'un Fragment et de son complémentaire

Étant donné qu'un fragment d'ADN est constitué de 4 nucléotides différentes il nous est apparu que 4 bytes étaient donc suffisant pour représenter un fragment. Tenant compte des GAP au seins d'un fragment ce nombre de 4 byte est maintenant porté à 5. Connaissant la taille d'un fragment avant de vouloir le représenter, nous avons choisi de représenter un fragment comme étant un objet uniquement composé d'un tableau de byte. La représentation de son complémentaire inversé se fait par construction sur base du fragment initial . Nous parcourons le tableau de byte en commençant par la fin et nous inversons comme suite les nucléotide : A $\hookrightarrow$ T , T $\hookrightarrow$ A , C $\hookrightarrow$ G , G $\hookrightarrow$ C , GAP $\hookrightarrow$ GAP . Pour chaque nucléotide trouvée en position i dans le tableau , nous insérons son complémentaire en position lenght-(i+1) ou lenght représente la taille du fragment initial. De fait, comme nous commençons le parcours du tableau par la fin, le premier élément que nous inspection est en réalité de dernier et a comme indice i = (length-1). De la même manière, le premier élément dans le tableau se trouve à l'indice i = 0. L'opération lenght-(i+1) = lenght-(lenght-1+1) = 0. Ainsi le nucléotide qui se trouve en dernière position dans le fragment initial sera complémenté avant d'être mis en première position dans le tableau du fragment représentant le complémentaire. Il en vas de même pour les autres nucléotide au positions restantes.

3 Graph : sommet et arc

A chaque sommet du graph que nous allons créer, nous assicions plusieurs informations que nous détaillons ici. Un sommet est un objet qui contient un fragment et 4 autre variable utilisez lors du chemin Hamiltonien, un boolean in qui représente un point d'entré dans le fragment, un boolean inC qui représente un point d'entré dans le complémentaire inversé fragment, un boolean out qui représente un point de sortie du fragment et un boolean outC qui représente un point de sortie du complémentaire inversé du fragment.

En ce qui concerne les arc, se sont des object composé de 5 variables : int indexSommetSrc l'indice du sommet source, int indexSommetDst l'indice du sommet destination int score le score de l'alignement représenté par l'arc entre le fragment contenu dans le sommet destination et celui contenu dans le sommet source, un boolean srcC qui est vrai si on a pris le complémentaire inversé du fragment dans le sommet source false sinon et un boolean dstC qui est vrai si on a pris le complémentaire inversé du fragment dans le sommet destination false sinon.

4 Construction du graph

La construction des sommet est expliquer dans la section Parser. la construction des arcs se fait comme expliquer ici : pour l'ensemble de nos sommet (qui sont stocké dans une liste et ou l'indice d'un sommet corresponds à son indice dans cette liste), nous créons une tache qui consiste à calculer tous les alignement possible (8 pour chaque paire de fragments) entre le fragment contenu dans le

sommet et les fragments contenu dans les sommets suivants dans la liste. Une fois cette tache créé, nous l'attribuons à un thread (nous avons au total 2 fois le nombre de threads disponible sur une machine et ce afin de combler chaque petit "trous" entre les threads disponible sur la machine). Tous les threads renvoient le résultat de leurs taches dans dans une seule et même file. Une fois que tous les threads ont finis leur travail, nous insérons le contenu de la file dans notre arc. LA raison pour laquelle chaque thread ne renvois par directement les résultats obtenu dans le graph et qu'il y a un phénomène de concurrence sur l'accès au graph.

5 Parser

Nous nous basons sur le format de représentation d'un fragment dans un fichier fasta afin de repérer les différent fragments. Pour ce faire nous faisons la distinction entre les ligne commençant par le caractère '>' et les autres ligne contenant les nucléotide du fragment, nous commençons donc par repérer la première ligne commençant par le caractère mentionné et passons a la ligne suivante, nous enregistrons l'intégralité de cette ligne et passons à la ligne suivante. nous réitérons ce processus jusqu'au moment ou nous arrivons à ne ligne qui commence par le caractère '>'. A ce moment nous savons que nous venons de lire un Fragment que nous enregistrons directement dans un sommet du graph. nous procédons ainsi jusqu'à ce que nous arrivions à la fin du fichier et nous enregistrons le dernier fragment dans le graph.

6 trie

Pur trier les arcs par score décroissant nous avons opté pour le trie par tas, nous ne reviendrons pas sur le fonctionnement du tas. Pour arriver a avoir notre ensemble d'arc trié, nous retirer simplement l'arc au sommet du tas (qui est de score max par rapport a tout les arc dans le tas) et nous insérons cet arc dans une ArrayList. Ensuite nous retirons le tas et recommençons jusqu'à ce qu'il ne reste aucun sommet dans le tas.

Nous avons ....