Xavier Grand
15 Avril 2020
HMSN205
Master SNS parcours BCD

INITIATION A L'ALGORITHMIQUE POUR LA BIO-INFORMATIQUE

Projet *mapper*

MASTER SCIENCES
ET NUMERIQUE POUR LA SANTÉ

Avant-propos.

La programmation, c'est comme les cigarettes. Lorsque l'on commence avec une marque, on n'en change quasiment jamais. J'ai commencé à programmer avec le langage python, et ce qui me paraît évident voire presque naturel en python, me demande des heures de recherche et de tâtonnement en C++. J'ai l'impression d'avoir appris à courir sur un tapis de course dans une salle climatisée et que l'on me demande de faire un trek en montagne en plein mois de juillet ! N'en serai-je jamais capable ? Tout est une question d'entrainement. Je ne saurais dire aujourd'hui si je me lancerai ce défi ou non. Toutefois, cela n'est pas insurmontable, mais nécessiterait un effort important et une bonne dose de patience.

I. Introduction

Dans cette partie, nous traiterons de l'intérêt, bien qu'évident, du séquençage de nouvelle génération en short-read, à travers trois exemples. Un premier exemple est l'assemblage de génome complet d'un individu, dans le cas où il existe un génome de référence pour cette espèce. Un second exemple relatif à la détection de variation génétique, nous nous intéresserons aux SNP, pour « Single Nucleotid Polymorphism », et son utilisation en amélioration des plantes. Et enfin un dernier exemple traitera de l'expression de gènes en génome complet par séquençage. Il s'agit, pour ce dernier, d'un séquençage d'ADNc, dit complémentaire, qui est en réalité la rétrotranscription des ARN messagers ; ainsi, il s'agit bien de séquençage ADN bien que l'origine des acide nucléiques soit de l'ARN.

1. Assemblage de génome

Le séquençage ADN, en short-read, peut être utilisé afin de séquencer un individu d'une espèce dont un génome de référence est disponible. Il est également possible de séquencer en short-read un individu d'une espèce dont il n'existe pas de génome de référence, or ce cas de figure est à la marge du travail effectué lors de ce projet, aussi nous n'en discuterons pas.

Si les approches short-read ne permettent pas toujours de réaliser un assemblage complet d'un génome en raison de la faible longueur des reads, elles n'en sont pas pour autant un outil inutile, bien au contraire. En effet, elles permettent d'obtenir une profondeur de séquençage importante, rapidement, à coût modéré. Aussi, plus la profondeur de séquençage est importante, plus les erreurs techniques de séquences sont « diluées » dans des séquences sans erreur. Ainsi, le mapping de résultats de séquençage ADN short-read est un outil de choix pour l'amélioration de séquence de génome complet.

Par conséquent, s'il est possible d'identifier des erreurs de séquençage, il est aussi possible d'identifier du polymorphisme entre les séquences de différents individus. Ou plus simplement, le polymorphisme d'un individu par rapport à un génome de référence. Il existe de nombreuses applications à l'identification de polymorphismes génétiques, nous en discuterons à propos d'un exemple dans le paragraphe suivant.

2. Détection de SNP et amélioration des plantes

Le secteur de l'amélioration des plantes, emblématique de la recherche effectuée par les semenciers, utilise les nouvelles technologies de séquençage ADN afin d'identifier du polymorphisme dans les différentes variétés d'espèces à intérêt agronomique. En effet, le polymorphisme génétique, entre autres en amélioration des plantes, est utilisé afin de cartographier le génome des individus/variétés/cultivars/écotypes d'une espèce.

Une approche très en vogue dans cette industrie aujourd'hui est la sélection génomique. Elle a pour but d'associer, sur un panel de diversité de variétés relativement réduit (une à quelques centaines), un ou plusieurs caractères phénotypiques à un ensembles d'allèles représentés par des marqueurs génétiques. Grâce à cette association, il est possible de réaliser une prédiction statistique de la part d'un allèle à un marqueur dans un caractère. Ainsi, le sélectionneur pourra à loisir associer dans son processus de sélection les allèles aux marqueurs qui lui permettront de choisir les caractères de la plante qu'il désire obtenir.

Les marqueurs les plus « fins » aujourd'hui, c'est-à-dire les marqueurs permettant d'avoir la plus fine définition pour une carte génétique, sont les SNPs. De ce fait, le séquençage short-read dans ce cas de figure est le plus adapté. En effet, les espèces à fort intérêt agronomique telles que le maïs ou le riz sont des espèces séquencées depuis de nombreuses années (< 2010), elles ont donc un génome de référence de bonne qualité. Il est donc possible, par le séquençage short-read, de séquencer un panel de diversité. Ainsi, pour chaque lignée du panel, l'alignement des reads permettra d'identifier les SNP par rapport à la séquence de référence. Chaque lignée du panel est phénotypées pour les caractères d'intérêt, il est alors possible d'associer génotype (ensemble des allèles aux marqueurs SNP) et phénotype.

Le coût réduit du séquençage et la capacité à réaliser un alignement de qualité des reads sont un outil d'une grande valeur pour l'industrie semencière.

3. Expression de gènes, transcriptomique

Une application du séquençage short-read, et du mapping de ces reads, est la mesure de l'expression des gènes. En effet, de la même façon que lorsque l'on souhaite mesurer l'expression de gènes par Northern-Blot, RT-qPCR ou encore par puce à ADN, il est nécessaire de réaliser une étape de rétrotranscription des ARN messagers extrait du tissu vivant dans lequel on souhaite mesurer l'expression des gènes. Ainsi, des ANDc, dit complémentaires, sont obtenus. Il est alors possible de séquencer ces ADNc.

De plus, le séquençage short-read est quantitatif. Ainsi, il est possible de mapper (aligner le read, et déterminer sa position sur la séquence de référence) les reads issus du séquençage des ADNc et de quantifier le nombre de reads qui mappent sur une même position d'un génome de référence. Lorsque cette position correspond à un gène exprimé, il est possible d'extrapoler l'accumulation des ARN messagers, c'est-à-dire de l'expression du gène transcrit.

II. Formats de fichier et importance des standards

1. Fasta :

Les fichiers Fasta sont des fichiers textes plats standardisés de séquence nucléique ou protéique très utilisés en bio-informatique et en biologie.

Ils sont construits de la manière suivante, une première ligne, l'entête, débute par un caractère ' > ' et est suivi généralement d'identifiants de la séquence ainsi que d'éventuels commentaires. Bien qu'ils soient optionnels, les identifiants et commentaires sont fortement recommandés afin de pouvoir identifier la séquence en question. Le caractère ' ; ' peut également être substitué au caractère ' > ' pour indiquer un entête, correspondant à un ancien standard. Puis cette première ligne est suivie de la séquence à proprement parler. Elle est composée de caractères de l'alphabet IUPAC (Table 1) et généralement découpée en tronçons de 80 caractères. Toutefois, ce nombre de caractères par ligne n'est pas contraint, il peut être de n'importe quelle taille inférieure ou égale à 120 caractères. Un fichier Fasta peut contenir plusieurs séquences, chaque séquence est précédée d'un entête. La fin d'une séquence est identifiée par le caractère ' > ' de la suivante. Pour de bonnes pratiques, il est recommandé d'introduire un saut de ligne entre chaque séquence (entête + séquence).

Alphabet IUPAC des Acides nucléiques	Base
A	Adenine
C	Cytosine
G	Guanine
T (ou U)	Thymine (ou Uracile)
R	A ou G
Y	C ou T
S	G ou C
W	A ou T
K	G ou T
M	A ou C
B	C, G ou T
D	A, G ou T
H	A, C ou T
V	A, C ou G
N	base quelconque
. ou -	gap

Table 1 : Alphabet IUPAC des acides nucléiques.

2. Fastq :

Les fichiers Fastq sont également des fichiers textes plats standardisés de séquences nucléiques. Ils sont construits de la même façon qu'un fichier Fasta, avec l'information de la qualité de séquençage en plus ainsi que quelques différences dans les caractères d'identification des séquences. En effet, l'entête d'une séquence est identifié par un ' @ ', suivi généralement par l'identifiant de la séquence ainsi que d'éventuels commentaires. Puis les lignes suivantes sont la séquence à proprement parler. Un caractère ' + ' indique la fin de la séquence et annonce le codage de la qualité du séquençage. Ce caractère peut être suivi de l'identifiant et des commentaires de l'entête. La ligne qui suit est le codage de la qualité, suivant différentes possibilités qui ne seront pas détaillée dans ce rapport. Toutefois, il est important de préciser que selon le format de codage de la qualité choisi, les caractères ' @ ' et ' + ' peuvent être utilisés.

3. Intérêt de la standardisation :

Dans notre cas de figure, la lecture des fichiers Fasta et Fastq, nous cherchons à automatiser l'identification des informations telles que les entêtes, les séquences ou encore la qualité de séquençage. Pour cela, nous utilisons les caractères récurrents comme des balises dans le corps du fichier. Ainsi, il est très important que les formats de fichiers soient standardisés, c'est-à-dire que ces éléments qui nous intéressent soient identifiés toujours de la même façon.

Dans le cas du format Fasta, les choix de caractères balises ne posent pas de problèmes particuliers, mis à part la possibilité, de plus en plus rare, de rencontrer un ' ; ' en guise de balise à la place d'un ' > ' plus actuel.

En revanche, en ce qui concerne le format Fastq, certains choix ne sont pas particulièrement judicieux. Et effet, la balise ' @ ' n'est sans doute pas le meilleur choix possible pour deux raisons. La première est qu'il est possible de retrouver ce caractère dans les commentaires de l'entête, un problème qui est cependant aisément contourné en cherchant un début de ligne commençant par un ' @ ' et non seulement un ' @ ' comme identifiant de nouvelle séquence. Le second problème, plus pernicieux, est la possibilité de coder la qualité de séquençage par ce caractère. Aujourd'hui encore, je cherche comment contourner ce défaut.

Par conséquent, la standardisation des formats est capitale pour l'automatisation des traitements informatiques, et de mauvais choix, à l'instar de celui des fichiers Fastq, peuvent entrainer des difficultés qui auraient pu être très facilement évitées.

III. Stratégie du projet

L'objectif de ce projet est de définir une stratégie de mapping de reads sur un génome de référence donné.

1. Paradigme et niveau de langage

Comme évoqué précédemment, nous allons traiter des fichiers plus ou moins standardisés, c'est-à-dire des fichiers structurés regroupant des séquences composées de manière systématique, d'un entête, de la séquence *sensu stricto* et, pour le cas des fichiers Fastq, d'un codage de la qualité de séquençage de ladite séquence.

Nous pouvons ainsi voir une séquence comme un objet, comportant les attributs ' entête ', ' séquence ' et éventuellement ' qualité '. De plus, les fichiers que nous traiterons contiennent plusieurs objets. Nous opterons donc naturellement pour un langage de programmation orienté objet.

D'autre part, une autre contrainte du projet relative au volume potentiel des données à traiter, i. e. des génomes complets reséquencés sur des profondeurs importantes, est à prendre en compte. En effet, si l'on considère un séquençage d'un individu de petit pois récemment séquencé (Kreplak *et al.*, 2019), dont le génome est de l'ordre de 4,42 Gpb (Giga paires de bases), soit 4,42 milliards de caractères, avec un octet par caractère, le stockage du génome complet en mémoire à lui seul représente 4,42 Go (Giga-octets, 10^9 octets) en unicode UTF-8 avec des caractères ASCII-US. Si ce volume n'est pas une contrainte pour les disques durs actuels, lorsque l'on va vouloir travailler sur la séquence il faudra la charger en mémoire vive qui est plus limité en espace. Une stratégie de compression de l'information a été utilisée, elle est décrite dans le paragraphe suivant. De plus, une grande quantité d'information à traiter implique nécessairement une grande quantité d'opérations à réaliser. Ainsi, nous avons intérêt à choisir un langage de programmation efficace en temps comme en espace. Par conséquent, les langages dits de bas-niveau, dont le fonctionnement plus proche de la machine va accroitre leur vitesse d'exécution, sont à privilégier.

Le langage de programmation, répondant à ces critères, qui nous a donc été proposé est le langage C++.

2. Encodage du génome de référence

- *4 bases ADN, en un minimum de bits*

Comme évoqué dans la partie précédente, le génome complet d'un individu peut nécessiter un grand volume de mémoire. Une des solutions possibles est de recoder les caractères ' A ', ' C ', ' G ', ' T ' et ' U ' représentant les acides nucléiques (faisant fi de la casse). Ainsi, il est possible avec un nombre minimal de 2 bits de coder ces 4(5) bases comme suit :

- 00 = ‘ A ‘,
- 01 = ‘ C ‘,
- 10 = ‘ G ‘
- 11 = ‘ T ‘ ou ‘ U ‘ sans distinction.

De cette façon, alors qu’un caractère tel que ‘ A ‘ en unicode UTF-8 est codé en 8 bits, il n’en occupera que 2 sous cette forme recodée. Par conséquent, un génome de 4,42 Gpb n’occupera que 1,1 Go et non 4,42.

Toutefois, il est nécessaire de mentionner que le recodage du génome, s’il permet une division par 4 de la quantité de mémoire nécessaire, entraine un surcoût en temps de calcul.

- *Que faire des bases non-séquencées ou indéfinies ?*

Il existe un cas de figure à évoquer, il s’agit des bases dans le génome qui sont inconnues, incertaines, non-séquencées. Elles sont représentées par un caractère ‘ N ’ dans l’alphabet IUPAC. Si l’on désire recoder chaque base du génome en n’utilisant que 2 bits par base, il n’est pas possible de représenter plus de 4 bases différentes, soit ‘ A ’, ‘ C ’, ‘ G ’, ‘ T ’ (ou ‘ U ’). Ainsi l’information manquante représenté par le caractère ‘ N ’ est perdue. En effet, il est choisi d’attribuer au hasard une base, ‘00’, ‘01’, ‘10’ ou ‘11’, aux bases inconnues. La profondeur de séquençage permet par la suite d’identifier et de réattribuer le « bon » nucléotide.

3. Compression de l’information, indexation du génome

- *Table des suffixes*

La table des suffixes est un tableau des positions des suffixes d’un mot dans l’ordre lexicographique. Elle se construit en établissant tous les suffixes d’un mot, ainsi un mot de 10 caractères comme le mot ‘‘MISSISSIPI’’ a 10 suffixes. Ls positions de chaque suffixe sont ordonnées selon l’ordre lexicographique de ces suffixes (Figure 1).

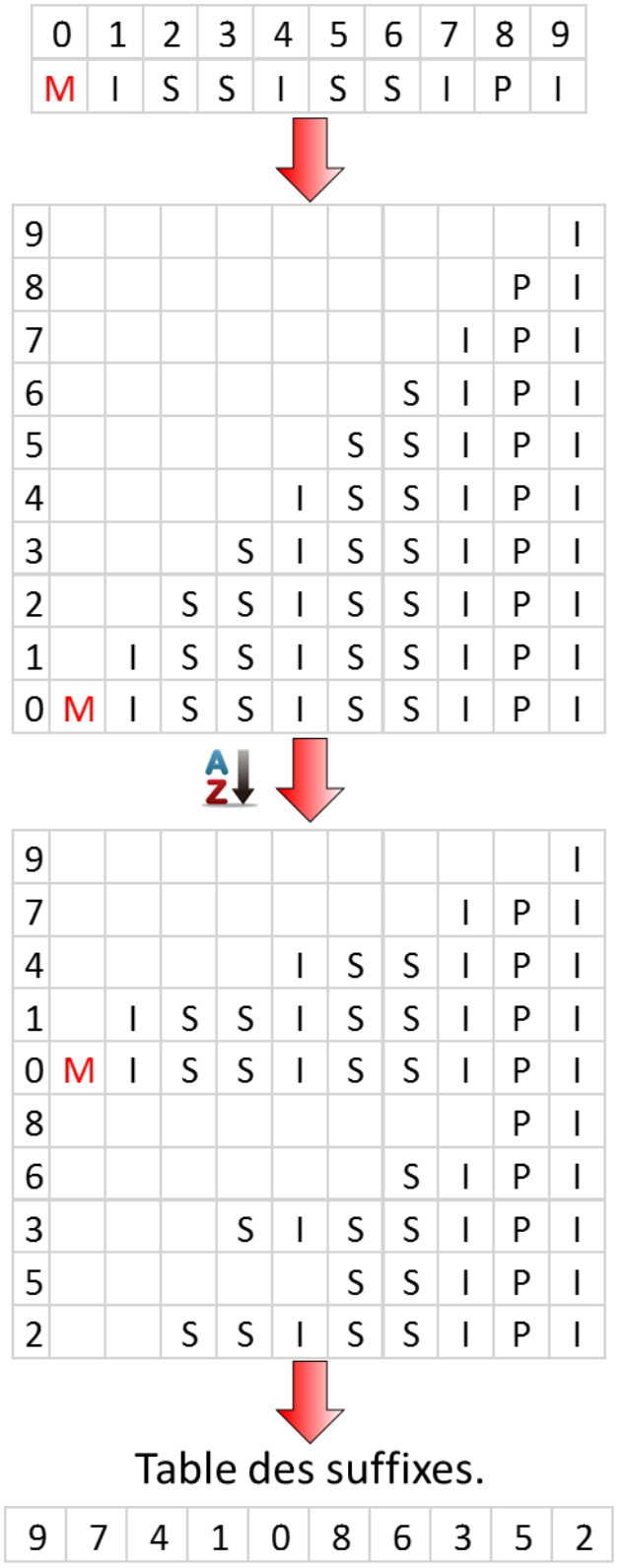

Figure 1 : Création de la table des suffixes du mot ‘‘MISSISSIPI’’.

Pour être utilisé, la table des suffixes peut être associée à une table des plus longs préfixes communs. Il s'agit d'une table qui contient la longueur du plus long préfixe commun entre deux suffixes consécutifs de la table des suffixes. Par exemple, nous venons de voir que le mot ''MISSISSIPI'' contient 10 suffixes, que l'on classe par ordre lexicographique pour créer la table des suffixes. Ainsi, les deux suffixes ''ISSIPI'' et ''ISSISSIPI'' sont consécutifs dans cette table et possèdent un préfixe commun ''ISSI'' de 4 caractères (Figure 2).

SA	LCP	Suffixes									
9	0	I									
7	1	I	P	I							
4	1	I	S	S	I	P	I				
1	4	I	S	S	I	S	S	I	P	I	
0	0	M	I	S	S	I	S	S	I	P	I
8	0	P	I								
6	0	S	I	P	I						
3	2	S	I	S	S	I	P	I			
5	1	S	S	I	P	I					
2	3	S	S	I	S	S	I	P	I		

Figure 2 : SA = Table des suffixes, LCP = Table du plus long préfixe commun.

- *FM-index*

Basée sur la transformée de Burrows-Wheeler (BWT), le FM-index est une méthode de compression de l'information. Dans le cas de figure qui nous intéresse, une séquence ADN est représentée par une chaine de caractère d'un alphabet, IUPAC en l'occurrence. Une matrice carrée est créée à partir de cette chaine de caractère, dont la première ligne correspond à la chaine elle-même, puis la suivante cette même chaine décalée d'un caractère, le dernier caractère de la chaine originale se retrouvant en première position de la ligne, et ainsi de suite pour chaque caractère de la séquence, exemple avec le mot ''MISSISSIPI'' (Figure 3, étape 1). Puis la matrice est classée par ordre lexicographique (Figure 3, étape 2). Enfin, ne sera conservée que la dernière colonne ainsi que la position de la chaine originale (Figure 3, étape 3).

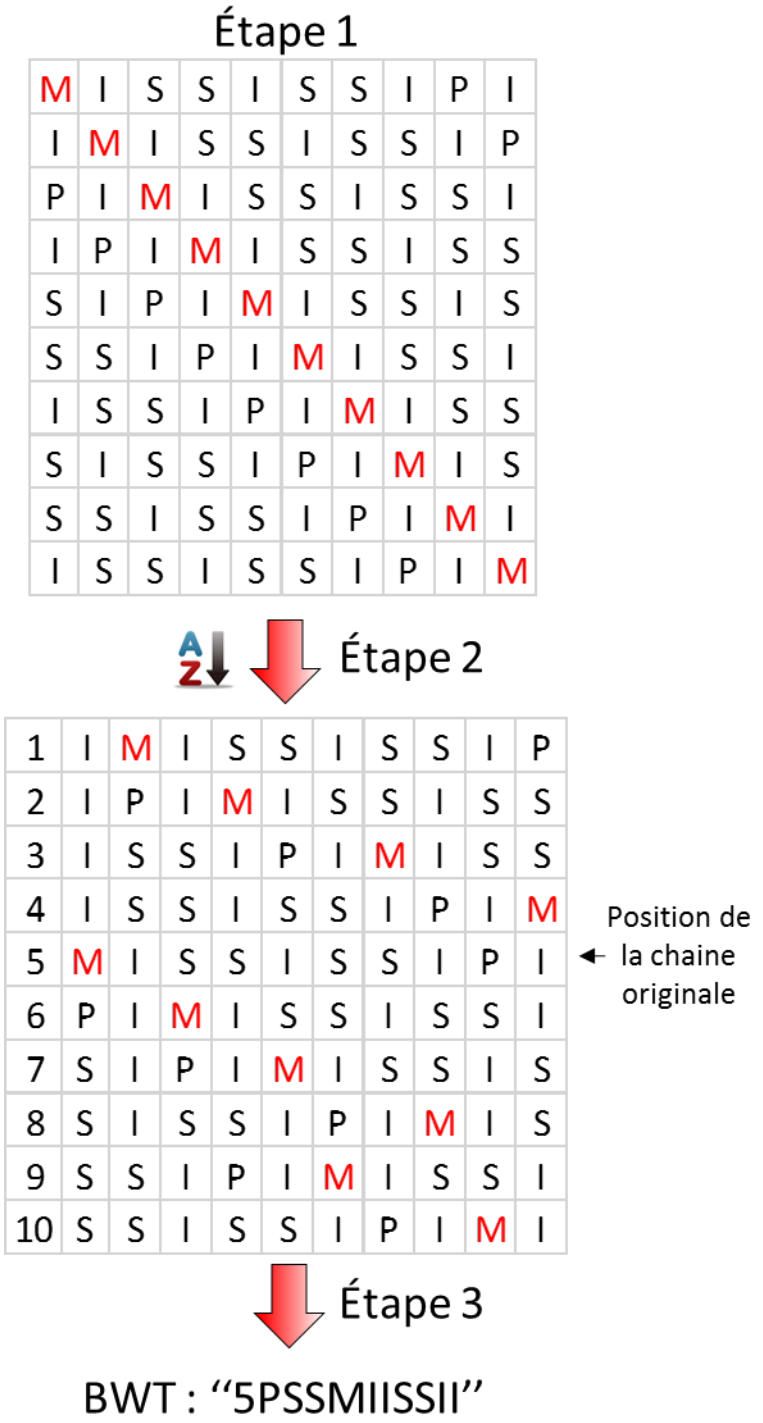

Figure 3 : Transformée de Burrows-Wheeler de la chaine ‘‘MISSISSIPI’’.

Ainsi, à partir de cette chaine transformée, il est facile de retrouver la première colonne de la matrice (celle de l’étape 2), puisqu’il s’agit simplement des mêmes caractères dans l’ordre lexicographique. Ainsi, par une succession de reconstitution par ordre lexicographique, il est possible de reconstruire la matrice et retrouver la chaine d’origine (Figure 4).

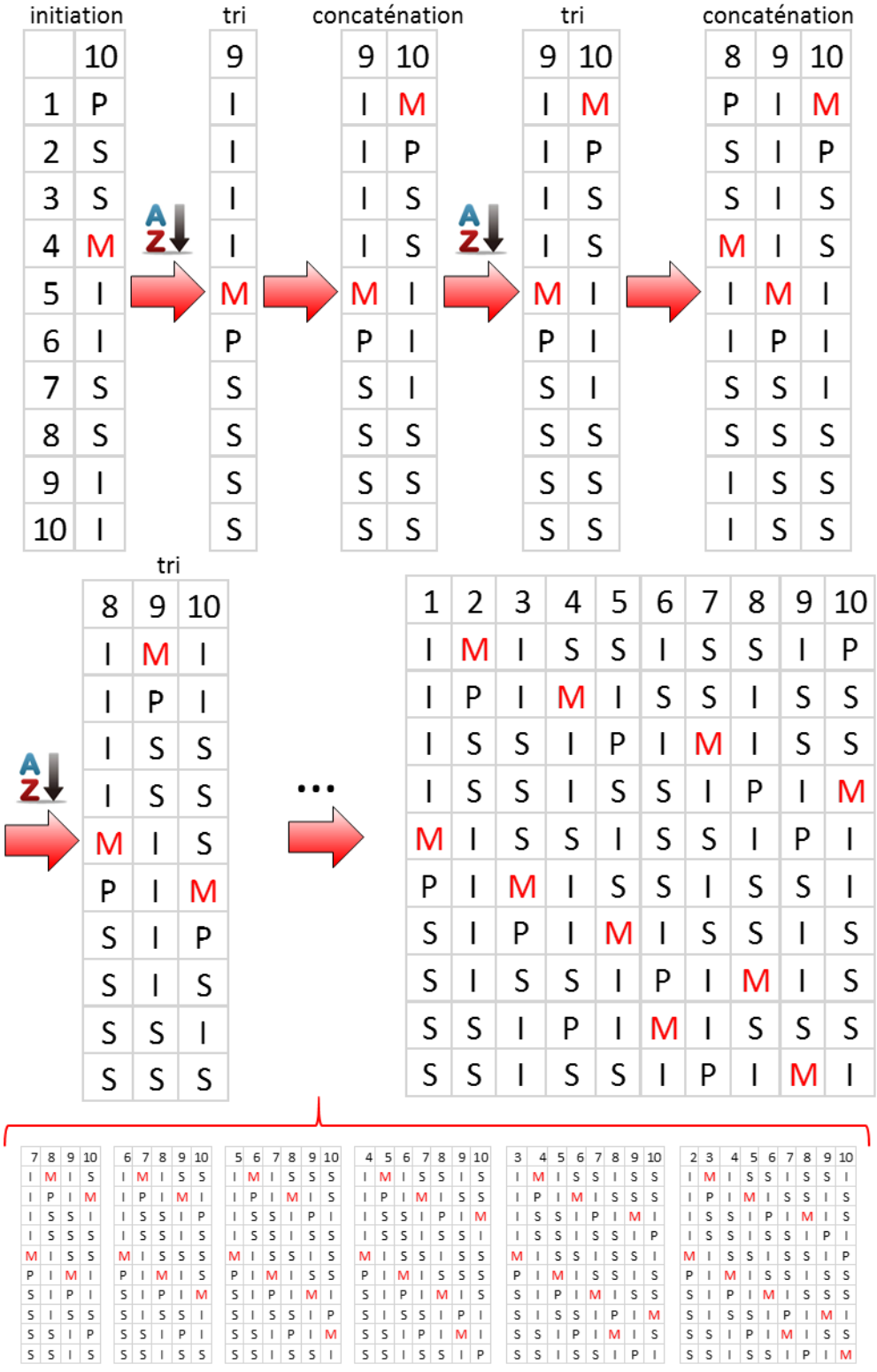

Figure 4 : Reconstruction de la chaine ‘‘MISSISSIPI’’ à partir de la BWT. En position 5ème ligne, la chaine d’origine est retrouvée.

- *Recherche de motifs*

La recherche de motifs dans la structure indexée est utilisée pour mapper les reads sur la séquence de référence. Dans ce projet, il est proposé d'utiliser la table des suffixes associée à la table du plus long préfixe commun. Il est alors possible de procéder par dichotomie dans ces tables pour comparer le motif recherché et les préfixes indexés.

De plus, les reads peuvent être découper en mots plus petits, de longueur k déterminée de manière empirique, que l'on nomme ''k-mers''. Ainsi, il est possible de comparer la position des k-mers dans le read et dans les index pour déterminer la présence, la position et le nombre d'occurrences du reads dans la séquence de référence.

IV. Architecture et implémentation

1. Modélisation de l'API *mapper*

La modélisation d'une API en amont de son implémentation est une étape à ne pas négliger. Mais comme tous débutants, je n'ai dérogé à la règle en me lançant directement dans le code. Or, de nombreuses questions viennent très rapidement lorsque le projet n'est pas modélisé avec soin. J'ai donc perdu beaucoup de temps en faisant et défaisant mon travail par manque d'anticipation.

Finalement, je suis revenu sur l'architecture du projet en envisageant le problème dans sa globalité. La formalisation par diagramme de classe UML est un outil de choix dans l'élaboration de la structure d'une API (Figure 5).

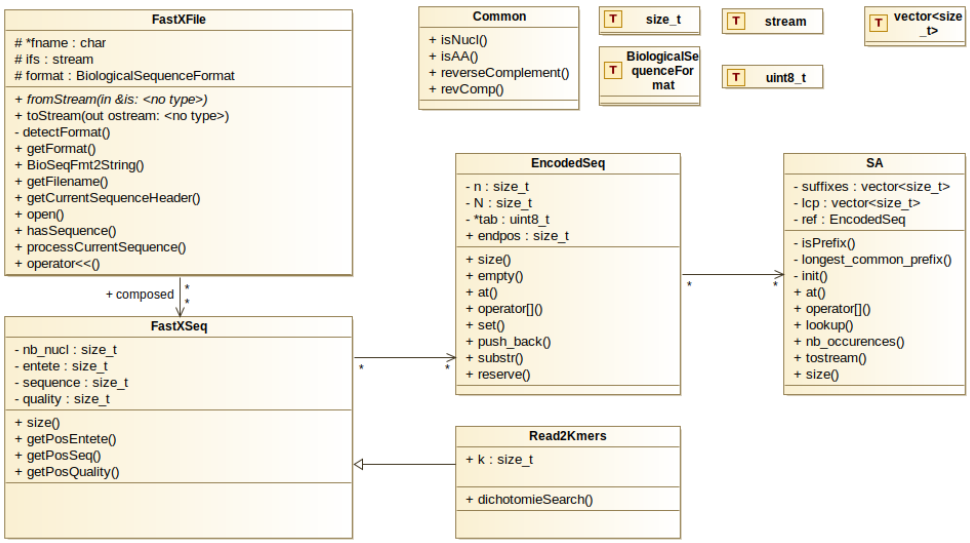

Figure 5 : Diagramme de classe UML, modélisation de l'API *mapper*.

2. Description des différentes classes

- *La classe FastXFile*

La classe FastXFile est une classe qui permet de lire et de traiter les fichiers d'entrée (Fasta et Fastq), de les analyser, ou parser, pour en identifier les différents entêtes, séquences, séquence-qualités éventuelles. La fonction ''processCurrentSequence'' crée des objet FastXSeq à partir des fichiers d'entrée.

- *La classe FastXSeq*

La classe FastXSeq permet l'instanciation d'objets contenant les informations relatives à chaque séquence. C'est-à-dire, la position dans les fichiers d'entrée de l'entête, de la séquence, éventuellement de la séquence-qualité pour les séquences des fichiers Fastq, et la longueur des séquences.

Le but de cette classe est de stocker l'information relative à chaque séquence mais pas les séquences en elles-mêmes. Ensuite, selon qu'il s'agisse des séquences de référence ou des séquences des reads, le traitement sera différent. En effet, les séquences de référence seront encodées puis indexées, alors que les séquences des reads seront « mappées » sur les séquences de référence indexées.

- *La classe EncodedSequence*

La classe EncodedSequence a pour objet d'encoder les séquences de références, généralement de grandes tailles dans un format décrit en partie III.2. Dans cette partie du projet, il faudra être particulièrement attentif à la gestion de la mémoire, puisqu'il s'agit potentiellement de manipuler des objets de très grand volume. D'autre part, les séquences encodées devront être traitées pour créer la table des suffixes de la classe SA.

- *La classe SA*

La classe SA, pour ''Suffixe Array'', est une classe qui va permettre de créer une table des suffixes à partir des séquences de référence. En effet, il s'agit de la méthode d'indexation choisie pour ce projet. La création d'une table des suffixes est décrite en partie III.3.

- *La classe Read2Kmer*

La classe Read2Kmer est envisagée comme une classe fille de la classe FastXSeq. En effet, il s'agit ici de traiter chacun des reads stockés sous la forme d'objet FastXSeq, en les découpant en K-mer de longueur k, et de les mapper à l'aide d'une fonction représentée par la fonction dichotomieSearch() dans le diagramme UML (Figure X).

- *Main, ou point d'entrée du programme*

Le ''main'' est la partie du programme qui va faire le lien entre toutes les classes définies. Dans un premier temps, les fichiers d'entré, doivent être passés en

arguments. Nous faisons le choix, et c'est un parti pris, que le premier fichier passé en argument correspond à un fichier Fasta contenant la ou les séquences de référence, et le second est un fichier Fastq, contenant le, mais plus vraisemblablement les séquences des reads à mapper sur les séquences de référence.

Ainsi, le fichier Fasta est alors parsé en premier. Les séquences de références sont identifiées. Un objet FastXSeq par séquence est créé, contenant les positions des entêtes et des séquences de références. Ces séquences sont encodées dans des objets EncodedSequence, puis la table des suffixes est créée.

Ensuite, le fichier Fastq est traité. Les différents reads sont identifiées. Un objet FastXSeq par read est créé, contenant les positions des entêtes, des séquences et des séquences-qualités.

Enfin, il reste encore à réaliser la recherche des séquences des reads dans les séquences de références en utilisant la table des suffixes créée précédemment, et en introduisant le principe de découpage des séquences à rechercher en k-mer.

V. Réflexions, prise de recul, idées

1. Gestion du temps et organisation dans la réalisation du projet

L'organisation de mon temps la réalisation de ce projet a été la partie la plus difficile, la moins réussi, et par conséquent a eue de nombreuses répercussions sur le résultat final.

En effet, face à la complexité de ce nouveau langage, je n'ai su comment m'y prendre pour me former correctement. La bonne méthode aurait sans doute été de suivre un tutoriel en ligne sans me soucier dans un premier de l'implémentation des fonctions, même simples, du projet. En fin de compte, j'ai perdu beaucoup de temps à faire et défaire sans parvenir à régler les problèmes de mon code.

Mon autre erreur a été de prendre les problématiques les unes derrières les autres sans chercher à avoir une vision globale du projet. Je savais que je partais des fichiers d'entrée Fasta et Fastq, et tant que je ne suis pas arrivé à les lire correctement (ce qui n'est toujours pas le cas), je n'ai pas réfléchi à la suite du projet. De ce fait, je n'ai su m'approprier correctement les codes établis lors des séances de TP.

Comme évoqué dans une partie précédente, la modélisation et la réflexion globale du projet m'aurait grandement facilité l'implémentation. La détermination des structures des données est largement plus importante à déterminer rapidement que la façon de lire un fichier ou de détecter une nouvelle séquence par un caractère donné.

2. Réflexion sur la complexité du programme

Un des éléments dont j'ai pu prendre toute la mesure est la notion de complexité. Nous avons, dans les enseignements précédents, réalisé des scripts sans trop nous soucier des ressources matérielles. Or, le cas de figure (théorique) qui nous était proposé dans ce projet, imposait de prendre en compte la possibilité de manipuler des données particulièrement volumineuses ainsi que des problématiques dont les résolutions naïves ne sont pas envisageables.

A titre d'exemple, nous avons vu qu'il y a une balance entre cout en temps de calcul et cout en espace mémoire selon le mode d'indexation choisi. En effet, l'encodage du génome de référence permet un gain d'espace mémoire considérable, d'un facteur 4, mais il exige un coût en temps de calcul de l'ordre de O(n), avec n la taille du génome. Ainsi, il est peut-être possible d'établir une conditionnelle en fonction de la taille du génome de référence, s'il est petit, pas de recodage (gain de temps de calcul), s'il est grand, recodage (gain de mémoire).

D'autre part, pour de très grands génomes, le FM-index peut prendre un grand volume en mémoire. Or, il existe des algorithmes de compression compatibles avec la transformée de Burrows-Wheeler tels que le codage de Hoffman qui consiste à coder le nombre d'occurrences successives d'un caractère dans un mot. Par exemple, pour le mot ''MISSISSIPI'', il est recodé de la façon suivante : ''MI2SI2SIPI''. Dans ce cas de figure il n'y a aucun gain en espace, mais il est possible d'imaginer qu'il puisse y avoir un gain sur un mot plus long tel qu'un génome par exemple. C'est une assertion qui a sans aucun doute été testée, j'argue que cela dépende grandement de la composition du génome, s'il présente de long ''stretchs'' ou non.

Une partie cruciale dans le design d'une API est donc la prise en compte du coût en temps de calcul et du coût en mémoire.

VI. Références

1. Kreplak, J., Madoui, M., Cápal, P. *et al.* A reference genome for pea provides insight into legume genome evolution. *Nat Genet* **51,** 1411–1422 (2019). https://doi.org/10.1038/s41588-019-0480-1

VII. Après l'effort… Un peu de détente.

"Si debugger, c'est supprimer des bugs, alors programmer ne peut être que les ajouter. " Edsger Dijkstra. Mathématicien et informaticien.

"Avec C il est facile de se tirer dans le pied. En C++, c'est plus difficile, mais lorsque vous le faites, vous pulvérisez votre cuisse entière." Bjarne Stroustrup. Créateur de C++.

"Cinquante ans de recherches sur les langages de programmation, et nous nous retrouvons avec C++ ?" Richard A. O'Keefe. Informaticien.

"C++ est un langage horrible. Même si on utilisait C pour ne rien faire, cela aurait l'avantage de se débarrasser des programmeurs en C++." Linus Torvalds, inventeur de Linux.

"Il n'y a que deux sortes de langages de programmation : ceux dont les gens disent toujours du mal et ceux que personne n'utilise." Bjarne Stroustrup.

Mais également, par rapport au projet en lui-même :

"LE DUC D'AQUITAINE _ Ce que vous nous proposez, c'est ce que j'ai envie d'appeler une mise en scène de l'échec. Parce que comme vous le dites, il y a peu de chances que ça marche. Mais ce qu'il faut savoir c'est que nous derrière, on va gérer cet échec. Et je ne parle pas de l'échec notion, je parle bel et bien de l'échec épreuve.

BOHORT _ Ou de l'échec, échec, si j'ose audacieusement paraphraser notre ami…''.

Kaamelott, Livre V, épisode 23 Le Forfait, Alain Chabat, Nicolas Gabion ; Alexandre Astier.