IFT3290 Hiver 2006 — Devoir 1

Miklós Csűrös

27 janvier 2006

À remettre au cours du 10 février.

1 Comparaison d'alignements (20 points)

On voudrait comparer deux alignements A_1 et A_2 de la même paire de séquences S et T. Cette comparaison peut être fait en calculant une distance d'édition. Pour cette distance d'édition, A_1 peut être transformé en A_2 en décalant les rangées pair insertion ou suppression d'un indel à la fois, mais les «vrai caractères» qui viennent de S et T ne peuvent pas être changés. Exemple :

$$A_1 = \begin{bmatrix} \mathtt{A} & \mathtt{C} & - & - & \mathtt{T} \\ \mathtt{A} & \mathtt{T} & \mathtt{A} & \mathtt{A} & \mathtt{T} \end{bmatrix}, \qquad A_2 = \begin{bmatrix} \mathtt{A} & \mathtt{C} & - & - & - & \mathtt{T} \\ \mathtt{A} & - & \mathtt{T} & \mathtt{A} & \mathtt{A} & \mathtt{T} \end{bmatrix}$$

Ici, A_1 peut être transformé en A_2 par une insértion d'un indel dans la deuxième position de la séquence en bas, suivi par l'insértion d'un indel dans la troisième position de la séquence en haut. Donnez un algorithme qui calcule la distance d'édition entre deux alignments. Est-ce que ce problème est comparable à celui du calcul de distance de Levenshtein entre deux séquences ? Analysez le temps de calcul de votre algorithme.

2 On tient compte des meilleurs (20 points)

Après avoir rempli le tableau de programmation dynamique pour trouver la distance d'édition entre deux séquences, on peut retrouver la solution optimale en retraçant les pas de la procédure. Donnez un algorithme efficace pour compter le nombre de solutions optimales, ou plus précisement, solutions avec le même nombre minimal d'opérations d'édition. Quel est le temps de calcul de votre algorithm ?

3 D'ailleurs la pondération la plus simple n'est pas aussi simple après tout (25 points)

En parlant de la pondération de substitutions, on a travaillé souvent avec la pondération simple de $+1$ pour match et -1 pour mismatch. Dans le contexte de maximum vraisemblance («Alignement de deux séquences», acétates 15–18), quelles distributions de nucléotides et probabilités de transition mènent à un tel système de scores ? En d'autres mots, trouvez des solutions pour $\pi_{\sigma'}$ et $p_{\sigma\to\sigma'}$ quand $\sigma, \sigma' \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Les scores sont choisis en utilisant une facteur d'échelle α quelconque, mais sans l'arrondi au plafond. Quelles solutions existent pour une distribution uniforme de nucléotides, c'est à dire quand $\forall\sigma\colon \pi_\sigma = 1/4$?

4 Outil (35 poins)

Ce problème vous demande d'implanter un outil en Java pour la manipulation de fichiers Fasta.

Votre programme doit lire un fichier Fasta de contenant une séquence ADN, possiblement avec des caractères ambigus. La séquence doit être stockée par l'objet `DNASequence` en utilisant un maximum d'un octet par nucléotide. Cette borne (formellement : $\ell + o(\ell)$ octets pour une séquence de longueur ℓ) ne doit jamais être dépassé à aucun moment de l'exécution. Attention : une variable de type `char` utilise quatre octets (un *mot*), et un objet de type `String` prend au moins quatre fois autant d'octets que sa longueur. Vous devez fournir l'implantation de cet objet avec des méthodes `char get(int pos)` et `String getName()` qui fournissent le caractère en position `pos` (premier caractère à `pos=1`), et le nom de la séquence (qui est la partie après le '>' de l'en-tête dans le fichier Fasta), respectivement.

L'outil que vous implantez est pour extraire des sous-séquences d'un fichier Fasta. Il sera appellé par

`java extractFasta` ⟨nom du fichier⟩⟨sous-séquence⟩

L'argument ⟨sous-séquence⟩ est de la forme $a..b$ ou `T`$a..b$, où a et b sont des entiers positifs qui spécifient la sous-séquence demandée. La forme `T` est pour produire des séquence protéiques, la traduction[1] de la sous-séquence d'ADN $a..b$. Si $b < a$, alors utilisez le complément inverse de la sous-séquence $b..a$. Le résultat est écrit à la sortie standard (`System.io`) dans le format Fasta (faites attention à la longueur des lignes), utilisant le même en-tête que la séquence originale, avec l'information sur la sous-séquence ajoutée.

[1] Pour la code génétique, consultez http ://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi ?mode=c. Utilisez le «Standard code».

Exemple :
Fichier d'entrée (`ale.fa`) :

```
>xxx09877 Sequence aleatoire
ATGGCGAGAGAGGTGCCTATAGAGAAATTGAGAAACATAGGTATAGTTGCTCACATTGACGCGGGTAAAA
CTACGACTACCGAGAGAATTCTCTATTACACGGGTAAGACTTACAAGATAGGTGAAGTTCACGAAGGTGC
```

Appels :

```
\% java extractFasta ale.fa 10..16
>xxx09877 Sequence aleatoire /substring=10..16 /length=7
GAGGTGC

\% java extractFasta ale.fa 20..14
>xxx09877 Sequence aleatoire /substring=complement(14..20) /length=7
ATAGGCA

\% java extractFasta ale.fa T63..71
>xxx09877 Sequence aleatoire /substring=63..71 /translated /length=3
G*N
```

Ce dernier exemple montre aussi comment traiter les codons d'arrêt.

5 Boni

1. Trois points de boni pour un rapport imprimé (et pas manuscrit), deux points de boni additionnels si LaTeX est utilisé.
2. Deux points de boni pour démontrer que la distribution de nucléotides doit être uniforme avec les scores ± 1 dans Problème 3.
3. Deux points de boni pour deviner l'origine de la séquence «aléatoire» de Problème 4.