Structures de données et algorithmes

Projet bioinformatique:
Recherche de motifs dans des séquences d'ADN

Jean-Michel Begon – Romain Mormont – Pascal Fontaine

4 mai 2020

L'objectif de ce projet est d'écrire un programme permettant de trouver la plus longue sous-séquence contiguë commune à deux séquences d'ADN, qu'on supposera de mêmes longueurs, notées N. Ce problème trouve de nombreuses applications en bioinformatique, comme par exemple la comparaison des génomes de différentes espèces.

Par sous-séquence contiguë, on veut dire une séquence constituée de nucléotides positionnés de manière contiguë dans les deux séquences comparées. Par exemple, soient les deux séquences suivantes :

— `ATGCAAGGGTGCCGA`
— `ACCATGCACTGATCT`

La plus longue sous-séquence **contiguë** est `ATGCA` (démarrant à la position 1 dans la première séquence et à la position 4 dans le seconde séquence). Dans la section 1 ci-dessous, on décrit quatre solutions potentielles au problème, qu'on vous demande de comparer d'abord théoriquement et puis empiriquement.

1 Algorithmes

On notera $G_1[1\,..\,N]$ et $G_2[1\,..\,N]$ les deux séquences (représentées par des tableaux) qu'on souhaite comparer de taille N toutes les deux. On se propose d'étudier les quatre approches suivantes pour résoudre le problème [1] :

Force brute. Pour chaque taille de sous-séquence L allant de N à 1, on énumère toutes les paires (S_1, S_2) de sous-séquences de taille L tels que S_1 apparaisse dans G_1 et S_2 apparaisse dans G_2 et on renvoie la première paire tels que $S_1 = S_2$. Vu qu'on considère des tailles de sous-séquences décroissantes, on trouvera effectivement la sous-séquence commune la plus longue.

Recherche dichotomique. Cette approche est identique à la précédente si ce n'est que plutôt que d'itérer sur toutes les valeurs de L allant de N à 1, on effectue une recherche dichotomique pour trouver L. Soit p et r les bornes entre lesquelles on recherche L.

1. On initialise p à 1 et r à N

1. Pour information, il existe une solution plus efficace faisant cependant appel à une structure de données non vue au cours (un arbre de suffixe).

2. Tant que $r >= p$:
   (a) On fixe q à $\lfloor \frac{p+r}{2} \rfloor$
   (b) On énumère toutes les paires de sous-séquences (S_1, S_2) de taille q comme pour l'approche force brute.
   (c) S'il existe une paire (S_1, S_2) telle que $S_1 = S_2$, on fixe p à $q + 1$.
   (d) Si aucune ne correspond, on fixe r à $q - 1$.
3. En sortie de boucle, on renvoie la dernière paire (S_1, S_2) avec $S_1 = S_2$ générée, NIL si aucune telle paire n'a été trouvée.

Table de hachage. Cette approche consiste à garder la recherche dichotomique sur la taille L des sous-séquences mais remplace la recherche exhaustive d'une paire (S_1, S_2) à l'intérieur de la boucle (étape 2.(b)) par l'approche suivante :

— On parcourt toutes les sous-séquences S_1 de taille q de G_1 et on les stocke dans une table de hachage H.
— On parcourt toutes les sous-séquences S_2 de taille q de G_2 et on renvoie la première qui se trouve dans la table de hachage H.

Soit $S[1\,..\,q]$ une sous-séquence, où on supposera que les nucléotides A, C, G, et T sont encodés respectivement par les entiers 0, 1, 2, et 3. On utilisera comme fonction d'encodage f d'une séquence, la fonction classique suivante :

$$f(S) = \sum_{i=1}^{q} S[i] \cdot 4^{q-i}. \tag{1}$$

L'avantage de cette fonction d'encodage est qu'elle permet de calculer de manière incrémentale l'encodage d'une sous-séquence de G_1 à partir de l'encodage de la sous-séquence précédente. Soient les sous-séquences $G_1[i\,..\,i+q-1]$ et $G_1[i+1\,..\,i+q]$ successives de G_1, on a la relation suivante :

$$f(G_1[i+1\,..\,i+q]) = (f(G_1[i\,..\,i+q-1]) - G_1[i] \cdot 4^{q-1}) \cdot 4 + G_1[i+q]. \tag{2}$$

En utilisant cette relation pour encoder les séquences successives de G_1 et de G_2, on peut fortement améliorer les temps de calcul de cette solution.

Programmation dynamique. Dans le cours (slide 400), une solution par programmation dynamique est proposée pour la recherche d'une sous-séquence commune *non-contiguë* la plus longue. Cette solution peut être adaptée pour trouver la sous-séquence *contiguë* la plus longue. Pour ce faire, la fonction de coût correspondante, $C[i, j]$, devient la longueur de la plus longue sous-séquence commune qui se termine à la position i dans G_1 et à la position j dans G_2. En ignorant les cas de base, $C[i, j]$ est égal à $C[i-1, j-1]+1$ si $G_1[i] = G_2[j]$ et à 0 sinon (puisque, dans ce dernier cas, aucune sous-séquence commune ne peut se terminer à ces positions). Une fois la matrice C calculée, on peut trouver la longueur de la plus longue sous-séquence en retrouvant la valeur maximale de C et retrouver la plus longue sous-séquence correspondante en remontant dans la matrice comme vu dans le cours.

2 Analyse théorique

Avant d'implémenter ces solutions, on vous demande de répondre aux questions suivantes dans le rapport accompagnant votre code :

1. Donnez un pseudo-code pour les solutions par recherche dichotomique et par table de hachage.
2. Donnez la formulation récursive complète de la fonction de coût $C[i, j]$ de la solution par programmation dynamique (c'est-à-dire en prenant en compte les cas de base).
3. En déduire le pseudo-code de la solution par programmation dynamique (y compris l'algorithme permettant de retrouver la sous-séquence la plus longue).
4. Donnez et justifiez les complexités en temps et en espace dans le meilleur et dans le pire cas en fonction de N des solutions suivantes :
   (a) force brute,
   (b) recherche dichotomique,
   (c) table de hachage sans calcul incrémental de la fonction d'encodage,
   (d) table de hachage avec calcul incrémental de la fonction d'encodage (Équation 2),
   (e) programmation dynamique.

3 Implémentation

On vous demande d'implémenter les solutions suivantes dans le langage de votre choix :
— La solution par recherche dichotomique (fichier `lcsdicho`)
— La solution par table de hachage avec calcul incrémental de la fonction d'encodage (fichier `lschash`)
— la solutions par programmation dynamique (fichier `lcsdp`).

Pour la solution par table de hachage, afin d'implémenter la solution incrémentale, vous devez calculer la fonction d'encodage vous-même mais vous pouvez ensuite utiliser la table de hachage fournie par le langage choisi. Si celui-ci n'en offre pas (ce qui n'est pas le cas du Perl), vous ne devez pas implémenter cette solution–vous devez, en revanche, faire l'analyse théorique quoiqu'il en soit.

Dans tous les cas, votre code doit prendre en entrée deux fichiers contenant les séquences génomiques à comparer, G_1 et G_2, dans le format FASTA, dont les noms seront fournis en ligne de commande. Une troisième argument en ligne de commande fournira le nombre de nucléotides des deux séquences à prendre en compte pour la comparaison. Cet argument permet ainsi de travailler sur des séquences de tailles variables tout en réutilisant les mêmes fichiers.

Dans les trois cas, votre code doit renvoyer en sortie la plus longue sous-séquence trouvée sous la forme d'un fichier FASTA, nommé par facilité respectivement `"subseqdicho.fa"`, `"subseqhash.fa"`, et `"subseqdp.fa"` avec comme commentaire dans l'entête du fichier l'information suivante :

`> L: XXX G1: YYY G2: YYY`,

où XXX est la longueur de la plus longue sous-séquence, YYY est sa position de départ dans G_1 et ZZZ sa position de départ dans G_2. Dans l'exemple ci-dessus, le fichier généré devrait contenir :

```
> L: 5 G1: 1 G2: 4
ATGCA
```

Si nécessaire, indiquez dans votre rapport comment utiliser les programmes fournis et expliquez vos choix d'implémentation.

4 Analyse empirique

Dans votre rapport, répondez aux requêtes suivantes :

1. Sur base de votre code et pour chacune de vos implémentations, générez une courbe montrant l'évolution des temps de calcul lorsque vous appliquez votre code sur des séquences de tailles croissantes. Vous pouvez utiliser pour cela les deux fichiers `g1.fa` et `g2.fa` fournis.
2. Pour chacune des méthodes, comparez ces courbes avec les résultats théoriques.

5 Deadline et soumission

Le projet est à réaliser **par groupe de deux maximum** pour le **08 mai 2020, 23h59** au plus tard. Il doit être soumis via la plateforme de soumission (`http://submit.montefiore.ulg.ac.be/`).

Le projet doit être rendu sous la forme d'une archive `tar.gz` contenant :

1. Votre rapport au format PDF. Soyez bref mais précis et respectez bien la numération des (sous-)questions.
2. Les deux/trois programmes demandés.

Bon travail !