Université de Sherbrooke - Département d'informatique
BIN702 - Algorithmes pour la bioinformatique (Automne 2023)

EXAMEN INTRA

Enseignant : Manuel Lafond Date : le 20 octobre 2023

- Cet examen d'une durée de 3 h 30 est individuel. **Je porterai une attention particulière au plagiat.**
- Vous devez remettre votre examen via turnin: `https://turnin.dinf.usherbrooke.ca/`.

  Vous devez remettre un fichier pdf contenant vos réponses (aucune forme spécifique n'est exigée).
- Toute forme de documentation est permise.
- Vous devez répondre à 7 questions totalisant 107 points. Si votre note dépasse 100, elle sera ramenée à 100.
- Vous pouvez **utiliser les résultats démontrés** en classe, dans les notes de cours et dans les exercices sans justification.
- Pour décrire vos algorithmes, vous pouvez soit fournir un pseudo-code, ou bien des phrases qui décrivent les éléments clé de votre algorithme. Les étapes importantes doivent être claires et leur complexité justifiée de façon appropriée. Sauf indication contraire, le **maximum de points** sera atteint si le code atteint une **complexité en temps optimale**.

QUESTION 1 : quelques questions d'échauffement (15 points)

Répondez aux questions de compréhension suivantes. Chaque question compte pour 3 points.

a. Soient S et T deux séquences et, selon une matrice de scores donnée, soient G leur score d'alignement global et L leur score d'alignement local. Vrai ou faux : G est nécessairement plus petit ou égal à L. Justifiez votre réponse.

b. Pour trouver des régions similaires entre deux séquences, on peut effectuer un alignement local ou utiliser l'algorithme BLAST. Donnez un avantage et un inconvénient d'utiliser BLAST par rapport à un alignement local. Justifiez brièvement vos réponses.

c. Soit S_1 et S_2 deux séquences de la même longueur n. Soit A_1 l'arbre de suffixes pour S_1 et A_2 l'arbre de suffixes pour S_2. Est-ce que A_1 et A_2 ont nécessairement le même nombre de noeuds? Si oui, dites pourquoi. Sinon, décrivez comment obtenir une séquence de longueur n dont l'arbre de suffixes a le nombre maximum possible de noeuds, parmi toutes les séquences de longueur n.

d. Dans l'assemblage de séquences, le graphe de chevauchements (aussi appelé le graphe d'*overlap*) contient une arête de X vers Y avec comme poids le plus long suffixe de X qui est un préfixe de Y. Expliquez l'intérêt de chercher un chemin de poids maximum qui passe par chaque sommet exactement une fois.

e. Lorsque nous discutions de profils HMM, nos modèles de markov cachés incluaient des états pour des *deletions* (D_i) et pour des *insertions* (I_i). Expliquez pourquoi chaque état I_i a une boucle vers lui-même, alors que ce n'est pas le cas pour les états D_i.

QUESTION 2 : alignement global et local (14 points)

Soient les deux séquences suivantes :

$$S = ACGC$$
$$T = GATTAG$$

a. (7 points) On veut calculer la **distance de Levenshtein** entre S et T. Donnez la table de programmation dynamique qui permet de trouver cette distance. Donnez ensuite un alignement global qui atteint cette distance.

Note : Vous devriez ajouter les flèches de trace arrière dans votre table. De cette façon, même si vous faites une erreur de calcul, je pourrai voir si vous avez appliqué les bonnes règles ou non.

b. (7 points) On veut calculer un alignement **local** de *score maximum*, en utilisant les scores +2 pour un *match*, -1 pour un *gap* et -1 pour une *mutation*. Donnez la table de programmation dynamique permettant de calculer l'alignement **local** de score maximum entre S et T. Donnez ensuite **tous** les alignements locaux de score maximum.

QUESTION 3 : arbres de suffixe (16 points)

a. (8 points) Construisez l'arbre de suffixes généralisé pour les trois séquences *balboa*, *alban* et *b* (oui, le dernier mot est seulement la lettre *b*).

b. (8 points) Rappelons que pour deux séquences S et T, on a définit $overlap(S, T)$ comme la longueur du plus long suffixe de S qui est aussi un préfixe de T. Ces valeurs servaient à reconstruire le graphe de chevauchements. Il est possible de calculer $overlap(S, T)$ en temps $O(|S| + |T|)$. Donnez un algorithme qui atteint cette complexité.

Indice: on peut tester chaque position i de S et voir si le suffixe démarrant à la position i est un préfixe de T. Pour atteindre un temps linéaire, il faudrait gérer chaque position en temps $O(1)$. Ma solution utilise des idées qui servaient à trouver les palindromes.

QUESTION 4 : séquençage et assemblage (14 points)

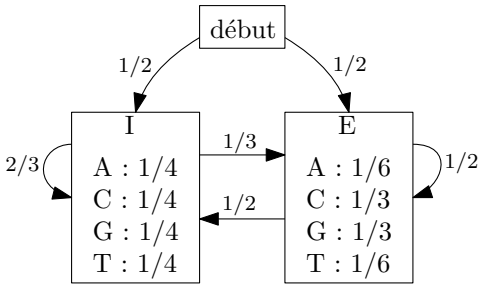
a. (9 points) Soit l'ensemble de 3-mers {GAC, ACT, ACA, CTA, ACG, CAC, TAC}. Construisez le graphe de De Bruijn correspondant à cet ensemble. Donnez ensuite une séquence contenant tous ces 3-mers exactement une fois, ou si ce n'est pas possible, dites pourquoi.

b. (5 points) Donnez la transformée de Burrows-Wheeler de la séquence *amalgama*. Je n'ai pas besoin de la matrice de rotations entière — seulement de la transformée finale.

QUESTION 5 : modèles de Markov (12 points)

a. (6 points) Un *intron* est un segment de gène qui est coupé de la molécule d'ARN après traduction, alors qu'un *exon* est un segment de gène qui y reste. Le modèle de Markov caché suivant classifie chaque position d'une séquence comme faisant partie d'un intron (état I) ou d'un exon (état E). Tout chemin commence à l'état début, qui n'émet aucun symbole.

Donnez la probabilité d'observer la séquence ACT en passant par la séquence d'états IEE. Vous devriez laisser une trace de vos calculs.



b. (6 points) Soit un HMM spécifié par (Q, α, Σ, e) tel que vu en classe. Soit S une séquence.

Nous avons vu que l'algorithme de Viterbi calcule une table V dans laquelle, pour tout $i \in \{1, 2, \ldots, n\}$ et tout état $q \in Q$, $V[i, q]$ est la probabilité maximum qu'un chemin d'états $q_1 q_2 \ldots q_i$ satisfaisant $q_i = q$ génère S. Dans le cas où $i > 1$, on a vu la récurrence

$$V[i, q] = \max_{h \in Q}(V[i-1, h] \cdot \alpha(h, q) \cdot e(q, S[i])).$$

Dans vos mots, expliquez pourquoi cette récurrence calcule correctement $V[i, q]$. Soyez concis(e).

Vous devez choisir **deux (2) questions** parmi les suivantes. Si vous me remettez plus de deux réponses, je corrigerai seulement les deux premières. Indiquez clairement vos choix.

QUESTION 6 : tableaux de suffixes et BWT (18 points)

Soit S une séquence de longueur n sur alphabet $\{A, C, G, T\}$, mis à part le dernier symbole qui est le caractère de terminaison \$. Le symbole \$ est lexicographiquement plus petit que les autres et il vient donc avant A, C, G et T dans l'ordre alphabétique.
On définit par $suff(S)$ la liste de tous les suffixes de S, *triée* en ordre alphabétique (croissant). Par exemple, si $S = ACGA\$$, alors $suff(S) = [\$, A\$, ACGA\$, CGA\$, GA\$]$.

a. (6 points) Supposons que vous avez accès à l'arbre de suffixes A pour la séquence S. Montrez que vous pouvez construire $suff(S)$ en temps $O(n^2)$.

Suggestion: on parcourt A récursivement en visitant les enfants dans un ordre approprié.

b. (6 points) Le *tableau de suffixes* de S est dénoté $SA(S)$, où SA est pour *suffix array*. $SA(S)$ est une liste de n entiers. La i-ème position de $SA(S)$ contient la position de départ du i-ème suffixe en ordre lexicographique. En d'autres termes, $SA(S)[i]$ est la position dans S du suffixe contenu dans $suff(S)[i]$. Par exemple, si $S = ACGA\$$, alors $SA(S) = [5, 4, 1, 2, 3]$. Montrez que si vous avez accès à l'arbre de suffixes A pour S, vous pouvez obtenir $SA(S)$ en temps $O(n)$.

Suggestion: c'est comme à la question précédente, sauf qu'à la sortie on a des entiers.

c. (6 points) Montrez qu'on peut calculer la transformée Burrows-Wheeler de S en temps $O(n)$.

Suggestion: si vous arrivez à trouver le lien entre $SA(S)$ et $BWT(S)$, ceci devient trivial.

QUESTION 7 : alignement non-symétrique (18 points)

Soient S et T deux séquences, avec $|S| = n$ et $|T| = m$.

a. (9 points) Supposons que vous voulez calculer un alignement global de score maximum de S vers T, mais qu'il n'y a eu aucune insertion[1] dans T. Ceci demande de calculer l'alignement global de S vers T, mais en considérant qu'aucun *gap* n'est permis dans S (des gaps sont permis dans T). Supposez qu'une matrice M vous donne tous les scores nécessaires.

Donnez les récurrences et conditions initiales de programmation dynamique permettant le calcul de cet alignement global en temps $O(nm)$.

b. (9 points) En plus des conditions de la question précédente, on suppose maintenant que les suppressions ont une longueur minimum c. Si vous décidez d'ajouter un *gap* dans T, il doit nécessairement être suivi de $c - 1$ autres gaps ou plus. Montrez comment calculer l'alignement global de S vers T sous ces conditions, en donnant les récurrences et conditions initiales. La complexité sous-jacente devrait être $O(n^2 m)$.

[1] Une application est l'alignement d'ARN sur un gène — l'ARN est issu du gène, a pu subir des suppressions, mais pas d'insertion.

QUESTION 8 : recherche de motifs répétés avec erreurs (18 points)

Soit S une séquence et ℓ, k et d, des entiers. On dit qu'un k-mer X est un (k,d)-*motif* de S si S contient ℓ k-mers à des positions distinctes, qui sont chacun à distance de Hamming[2] au plus d de X. En d'autres termes, X est un (k,d)-motif s'il existe des positions $p_1, \ldots, p_\ell$ distinctes telles que $S[p_i \,..\, p_i + k - 1]$ est à distance de Hamming au plus d de X, pour tout $i \in \{1, \ldots, \ell\}$. Notez que X n'est pas nécessairement une sous-chaîne de S.
Par exemple, si $S = ACGCCTACCGACC$ et $\ell = 4$, $X = ACC$ est un $(3,1)$-motif de S car il y a les k-mers ACG, GCC, ACC et ACC, respectivement aux positions $1, 3, 7, 11$ et qui sont tous à distance de Hamming 1 ou moins de ACC.

a. (6 points) Pour un ℓ donné, décrivez un algorithme permettant de trouver un $(k,0)$-motif dans S, s'il y en a un, en temps $O(n)$.

b. (12 points) Pour un ℓ donné, décrivez un algorithme permettant de trouver un $(k,1)$-motif dans S, s'il y en a un. Justifiez la complexité de votre approche par rapport à n et k, en considérant $|\Sigma|$ comme une constante. On suppose que n est beaucoup plus grand que k.

   Vous devriez viser une complexité $O(nk^3)$ ou mieux, mais vous aurez la majorité de vos points pour une complexité $O(n^2k^2)$.

QUESTION 9 : quelques problèmes d'assemblage (18 points)

a. (9 points) Soit $r_1, r_2, \ldots, r_m$ un ensemble de séquences représentant des *lectures* (aussi appelé *reads*) obtenues de données de séquençage. Étant donné un paramètre k, donnez le pseudo-code d'un algorithme qui reconstruit le graphe de Bruijn des k-mer présents dans ces lectures. Donnez ensuite la complexité de votre algorithme, avec justification.

   Vous n'avez pas à viser un algorithme optimal. Si votre algorithme est correct et que votre complexité est en temps polynomial par rapport à la longueur des r_i, vous aurez tous vos points.

b. (9 points) Supposons que G est un graphe de De Bruijn qui n'a pas de chemin Eulérien (un chemin orienté qui passe une fois par arête). Une explication possible est que certains k-mers ne sont pas présents dans l'entrée. Avec un peu de chance, il se peut qu'il manque un seul k-mer, ce qui correspondrait à une arête manquante dans le graphe.

   Donnez un algorithme qui, étant donné un graphe de De Bruijn G, détermine s'il est possible d'ajouter une arête à G pour que le graphe résultant ait un chemin Eulérien. Donnez ensuite la complexité de votre algorithme, avec justification optionnelle.

   Vous aurez un maximum de points pour une complexité optimale (à vous de la déterminer).

   Vous pouvez supposer que pour chaque noeud v, vous pouvez obtenir le nombre de voisins entrants et sortants de v en temps constant.

[2]Rappelons que la distance de Hamming entre S et T est le nombre de positions où les séquences diffèrent. Par exemple, la distance de Hamming entre $S = ACCGG$ et $T = TCCTG$ est 2.