Projet Algorithmique Spé

Flavie Tonon
HETIC - PMD | 2023-2024

Ce projet est à effectuer seul.e ou en binôme.

Le pseudocode à utiliser est celui vu en cours.

Le code peut se faire en C ou en Python.

Le rapport rendu sera passé dans un logiciel anti-plagiat et anti-IA.

Vous devrez rendre votre code ainsi qu'un rapport pdf compilant toutes les informations et analyses demandées. Le tout compressé dans un fichier projet_algo_PMD_nom1 _nom2.zip.

Table des matières

1] Algorithmes de recherche

1.1 Travail à faire

Écrire en pseudocode l'algorithme de recherche par interpolation. Donnez sa complexité algorithmique meilleur cas et pire cas, son variant et son invariant de boucle.

Vous devrez coder les algorithmes suivants en C ou en Python :

- La recherche linéaire
- La recherche dichotomique
- La recherche par interpolation.

Appliquez ces 3 algorithmes à des tableaux de tailles différentes :

- 100
- 100 000
- 1 million
- 100 millions.

Chaque tableau devra être initialisé et trié. Pour chaque taille de tableau, créez deux tableaux qui seront initialisés différemment :

- Un tableau de taille T sera initialisé avec des valeurs allant de 0 à T
- Un tableau de taille T sera initialisé avec des valeurs allant de 0 à T*5
- Vous pouvez tester différentes initialisations pour étayer votre analyse

Notez les temps d'exécution pour chaque taille et type d'initialisation de tableau appliqué à chaque algorithme. Comparez vos résultats à vos analyses théoriques initiales.

Nous n'analyserons pas le tri préalable des tableaux.

1.2 Les algorithmes

Recherche linéaire

Reprendre l'algorithme vu en cours.

Recherche dichotomique

Reprendre l'algorithme vu en cours.

Recherche par interpolation

Une recherche par interpolation s'applique uniquement sur un tableau trié.

Contrairement à la recherche dichotomique qui vérifie toujours l'élément du milieu en premier, la recherche par interpolation utilise un calcul d'estimation de la position de l'élément recherché. Ce calcul vous est fourni ci-dessous.

Comme pour la recherche dichotomique nous utiliserons les variables *debut* et *fin* qui contiennent l'indice de début et l'indice de fin du sous-tableau.

n contient toujours la valeur recherchée et *tab* le tableau trié.

Estimation de la position de *n* dans le tableau trié *tab* :

$$debut + \frac{(\text{fin} - \text{debut}) * (\text{n} - \text{tab[debut]})}{\text{tab[fin]} - \text{tab[debut]}}$$

2] Algorithmique et ADN

L'ADN contient une suite de bases nucléiques : A, T, C et G. Le but de ce projet est de réduire la taille d'un brin d'ADN en mémoire. En effet, compresser cette information peut s'avérer utile en raison de la taille importante de ces données.

Importez la séquence d'ADN fournie (14813 caractères) avec ce sujet et stockez-la dans un tableau *sequence*.

2.1 Algorithme de recherche de motif

Codez une fonction permettant de rechercher une séquence précise d'ADN (exemple : « AAATTCG ») et de retourner son emplacement dans la chaîne.

Codez une nouvelle fonction permettant de récupérer une liste des indices des différentes occurrences de ce motif.

2.2 Algorithme de compression naïf

Vous devrez coder un algorithme simple de compression de l'ADN :

- Identifiez les suites de caractères uniques (exemple : AAA)
- Remplacez ces suites de caractères uniques par le caractère suivant de son nombre de répétition (exemple : A3)

Vous devrez pour cela créer un nouveau tableau *sequence2* avec la nouvelle séquence compressée.

Quelle est la taille de cette nouvelle chaîne ? Expliquez.

2.3 Le nombre de suites de caractères uniques

Plutôt que de stocker les lettres et leur nombre de répétition dans un nouveau programme, nous allons stocker uniquement le nombre de répétitions dans un tableau *occurrences*.

Pour sequence = [A, T, C, G, G, G, T, T, A, A, A, A, A, T, C, C, C, C, C, G] on obtiendra *occurrences* = [1, 1, 1, 3, 2, 2, 5, 4, 5]

Calculez la moyenne du nombre d'occurrences successive d'une lettre à l'aide du tableau *occurrences*.

Comptez le nombre d'occurrences successive d'une lettre égal à 1 (compter le nombre de 1 dans le tableau *occurrences*)

Aidez-vous de ces résultats pour expliquer les résultats de l'algorithme de compression de la partie 2.2.

2.4 Nouvel algorithme de compression

Dans une séquence d'ADN certains motifs sont plus fréquents que d'autres. Déterminez des séquences fréquentes et remplacez-les par des nouvelles lettres.

Quelle est la taille de votre nouvelle séquence ? Analysez.