Université Paris Diderot Année 2020-2021

L3 M1 - Bio Info

Algorithmique

Exercice 1 : Plus longue sous-séquence commune

On appelle *mot* une séquence de caractères. Étant donné un mot w, une *sous-séquence* de w est un mot w' dont les lettres apparaissent dans l'ordre dans w, mais pas nécessairement consécutivement. Par exemple,

$$u' = \text{GTCGTCGGAAGCCGGCCGAA}$$

est une sous-séquence de

$$u_1 = \text{ACCGGTCGAGTGCGCGGAAGCCGGCCGAA}$$

car les lettres de u' peuvent être lues dans l'ordre dans u_1 par exemple de la façon suivante :

$$\text{ACCG}\mathbf{\text{GTCG}}\text{A}\mathbf{\text{GT}}\text{G}\mathbf{\text{CG}}\text{CG}\mathbf{\text{GAAGCCGGCCGAA}}.$$

Étant donnés deux mots w_1, w_2, on s'intéresse au calcul de la plus longue sous-séquence commune, c'est-à-dire du plus long mot w' tel que w' soit une sous-séquence d'à la fois w_1 et de w_2. Par exemple, étant donné

$$u_2 = \text{GTCGTTCGGAATGCCGTTGCTCTGTAAA},$$

on peut vérifier que u' est la plus longue sous-séquence commune de u_1 et u_2. Dans cet exercice, on écrit un algorithme basé sur la programmation dynamique pour calculer la longueur de la plus longue sous-séquence commune de deux mots. Soit deux mots v_1 et v_2 de longueurs n_1 et n_2. On note $\text{MaxSSC}(v_1, v_2)$ la longueur de la plus longue sous-séquence commune.

1. Si n_1 ou n_2 est nul, quelle est la valeur de $\text{MaxSSC}(v_1, v_2)$?
2. En supposant que la première lettre de v_1 ne fait pas partie de la plus longue sous-séquence commune de v_1 et v_2, exprimer la valeur de $\text{MaxSSC}(v_1, v_2)$ en fonction de $\text{MaxSSC}(\tilde{v}_1, v_2)$ où $\tilde{v}_1$ est un mot de taille strictement inférieure à celle de v_1. Même question si on suppose cette fois-ci que la première lettre de v_2 ne fait pas partie de la plus longue sous-séquence commune de v_1 et v_2.
3. En supposant que la première lettre de v_1 et la première lettre de v_2 sont égales et sont la première lettre d'une sous-séquence commune maximale, exprimer la valeur de $\text{MaxSSC}(v_1, v_2)$ en fonction de $\text{MaxSSC}(\tilde{v}_1, \tilde{v}_2)$ où $\tilde{v}_1$ et $\tilde{v}_2$ sont des mots de tailles strictement inférieures à celles de v_1 et v_2 respectivement.
4. Déduire une formule de récurrence pour calculer $\text{MaxSSC}(v_1, v_2)$.
5. En déduire un algorithme de type programmation dynamique pour calculer $\text{MaxSSC}(v_1, v_2)$. On précisera les dimensions du tableau utilisé pour stocker les résultats intermédiaires ainsi que la complexité de l'algorithme.

Exercice 2 : Problème du sac à dos

On souhaite mettre dans un sac à dos un certain nombre d'objets parmi une liste de N objets. Pour $i \in \{1, \dots, N\}$, l'objet i a une valeur $v_i \in \mathbb{N}$ et un poids $p_i \in \mathbb{N}$. Le sac à dos peut contenir un sous-ensemble d'objets de la liste des N objets dont le poids total ne dépasse pas une valeur limite $P \in \mathbb{N}$. La valeur d'un sous-ensemble d'objets de la liste des N objets est simplement la somme des v_i des objets i appartenant à ce sous-ensemble. On cherche à connaître la valeur maximale parmi les valeurs des sous-ensembles d'objets qu'il est possible de mettre dans le sac à dos en respectant la contrainte de poids. Pour $n \in \{0, \dots, N\}$ et $p \in \{0, \dots, P\}$, on note $\text{MaxSac}(n, p)$ la valeur maximale parmi les valeurs des sous-ensembles d'objets qu'il est possible de mettre dans un sac à dos de poids p parmi les objets $1, \dots, n$.

1. Si $n = 0$ ou $p = 0$, quelle est la valeur de MaxSac(n, p) ?
2. En supposant que l'objet n fait partie d'un sous-ensemble optimal parmi les objets $\{1, \ldots, n\}$ pour un sac de poids p, exprimer MaxSac(n, p) en fonction de MaxSac(n', p') pour $n' \leq n$ et $p' \leq p$ bien choisis.
3. En supposant que l'objet n ne fait pas partie d'un sous-ensemble optimal parmi les objets $\{1, \ldots, n\}$ pour un sac de poids p, exprimer MaxSac(n, p) en fonction de MaxSac(n', p') pour $n' \leq n$ et $p' \leq p$ bien choisis.
4. En déduire une formule de récurrence pour calculer MaxSac(n, p).
5. En déduire un algorithme de type programmation dynamique pour calculer la valeur de MaxSac(N, P). On précisera les dimensions du tableau pour sauvegarder les résultats intermédiaires ainsi que la complexité de l'algorithme.

Exercice 3 :

Le kayak est un sport nautique qui se pratique sur une embarcation appelée kayak, et la navigation se fait sur une rivière dans le sens du courant (on ne peut pas naviguer dans le sens contraire du courant). Il y a N stations de location de kayak le long d'une rivière, et on souhaite se rendre de la station 1 à la station N. Les stations se font concurrence et pratiquent des tarifs variables. Une table Tarif$[i, j]$ donne le tarif de location d'un kayak pris à la station i pour se rendre jusqu'à la station j. En pratique, il peut être plus économique, pour aller de la station 1 à la station N, de faire plusieurs arrêts entre 1 et N et louer sur plusieurs segments consécutifs.

1. Donner une récurrence qui permet de calculer le coût minimal Coût(i) de location pour aller de la station i à la station N.
2. Donner les dimensions du tableau utilisé pour faire le calcul du coût minimal, et expliquer comment celui-ci doit être rempli.
3. Donner l'algorithme de type programmation dynamique qui calcule la valeur du coût minimal.
4. Donner la complexité en temps et en espace de cet algorithme.