

Algorithmique — M1 — session 2

Examen du 17 juin 2010

Université Paris Diderot

Le sujet est recto-verso ! Documents autorisés : Deux feuilles de papier format A4
Durée : 3h (de 15h30 à 18h30)

On applique un résultat de cours

Exercice 1 – Récurrence

Étant donné que

$$T(n) = 25T(\lfloor n/5 \rfloor) + 36n^2$$

trouvez le comportement asymptotique de $T(n)$. Justifiez votre réponse.

On adapte un algorithme de cours

Exercice 2 – Une maison à louer

M et Mme Le Glouton cherchent à louer leur maison de campagne pour le mois d'août. Il souhaitent avoir le nombre maximal possible de locataires. Il est impossible d'avoir deux locataires en même temps, mais le jour de départ d'un locataire peut coïncider avec le jour d'arrivée d'un autre. Chaque locataire potentiel a envoyé une requête avec une date d'arrivée et une date de départ (ces dates ne peuvent pas être modifiées).

1. Trouvez le planning optimal pour les requêtes suivantes : 1-23 ; 4-17 ; 7-9 ; 9-16 ; 5-31 ; 26-30 ; 16-19 ; 18-20.
2. Proposez un algorithme glouton pour un ensemble de requêtes arbitraire : d_1-f_1 ; d_2-f_2 ; ... ; d_n-f_n
3. Donnez une ébauche de preuve de correction de votre algorithme.
4. Analysez la complexité de votre algorithme.

On invente des algorithmes avec des méthodes imposées.

Exercice 3 – Produit

Pour un tableau d'entiers $C = (c_s, \dots, c_f)$ on cherche à calculer leur produit. La méthode imposée est diviser-pour-régner, en coupant le tableau en 2 moitiés.

1. Écrivez un algorithme naïf et trouvez sa complexité.
2. Écrivez un algorithme Diviser-pour-régner.
3. Trouvez sa complexité.

Exercice 4 – Des lettres et des chiffres – backtracking

Étant donné un multiset (ensemble avec répétitions) de lettres, on cherche à trouver le mot le plus long composé de certaines de ces lettres. Par exemple, pour $A = \text{ERUIGHURH}$ le mot le plus long est RIGUEUR. On suppose donnée une fonction booléenne $\text{estUnMot}(w)$ qui répond est-ce que la chaîne w est un mot français.

1. Programmez une fonction récursive $\text{chercher}(u: \text{chaîne}, A: \text{multiset}, k: \text{entier})$ qui trouve et renvoie un mot de k lettres composé du préfixe u suivi de plusieurs lettres de A . Par exemple, $\text{chercher}(\text{PA}, \text{SETATTAH}, 7)$ pourrait renvoyer "PATATES". Si un tel mot n'existe pas il faut renvoyer NULL.
2. En utilisant la fonction chercher du premier point, écrivez la fonction $\text{jouer}(A: \text{multiset})$ qui trouve le mot le plus long composé de lettres de A .
3. Expliquez le fonctionnement de votre algorithme (fonctions chercher et jouer) en termes de recherche dans un arbre (quel est l'arbre, qu'est-ce qu'on cherche, comment).
4. Supposons qu'on dispose aussi d'une fonction booléenne $\text{estUnPréfixe}(w)$ qui répond si la chaîne w est un préfixe d'un mot français. Programmez $\text{chercher2}(u, A, k)$ plus rapide que chercher en profitant de la fonction $\text{estUnPréfixe}(w)$ pour ne pas regarder des chaînes non-prometteuses.

Exercice 5 – La meilleure période - encore une fois

Le bénéfice net (positif ou négatif) de la société D&Q pendant n dernières années est représenté par le tableau (b_1, b_2, \dots, b_n) . Pour une période contiguë $i..j$ on définit le bénéfice cumulé $BC = (b_i + b_{i+1} + \dots + b_j)$. On s'intéresse à la meilleure période de l'histoire de la société où le bénéfice cumulé est maximal, et surtout à la valeur BCM de ce bénéfice cumulé maximal.

En révisant l'examen précédent vous avez trouvé un algorithme naïf en $O(n^2)$ et un algorithme diviser-pour-régner en $O(n \log n)$. Aujourd'hui vous ferez mieux en utilisant la programmation dynamique.

On propose d'étudier deux fonctions

- $f(i)$: le bénéfice cumulé maximal pour la meilleure période entre b_1 et b_i
- $g(i)$: le bénéfice cumulé maximal pour la meilleure période entre b_1 et b_i avec la contrainte que cette période se termine à l'année i

(dans les deux cas cette période peut être vide, et les fonctions égales à 0).

1. Pour $\beta = (-5, 1, 3, -1, 10, -2, 0)$ trouvez $f(3), f(4), g(3), g(4)$.
2. Écrivez des équations de récurrence pour calculer $f(i)$ et $g(i)$ en sachant les valeurs de ces fonctions pour les arguments $< i$. N'oubliez pas le cas de base
3. Proposez un algorithme de programmation dynamique (récursif avec marquage ou itératif) pour calculer le BCM (Conseil : calculez les deux fonctions f et g ensemble). Écrivez un pseudocode.
4. Analysez la complexité de votre algorithme.