Programmation avancée

Examen écrit, 15 janvier 2015

Livres fermés. Durée : 3h30.

Remarques

— Répondez à chaque question sur une feuille **séparée**, sur laquelle figurent votre nom et votre section.
— Soyez bref et concis, mais précis.
— Sauf mention explicite, toutes les complexités sont à décrire par rapport au temps d'exécution des opérations concernées. Soyez toujours le plus précis possible dans le choix de votre notation (Ω, O ou Θ).

Question 1 : Vrai ou faux

Pour chacune des affirmations suivantes, déterminez si elle est vraie ou fausse et justifiez brièvement votre choix :

(a) $3n \log_2 n + 10n \in \Theta(n \log_{10} n)$

(b) Pour toutes fonctions $f(n)$, $g(n)$ et $h(n)$, si $f(n) \in O(g(n))$ et $f(n) \in \Omega(h(n))$, alors $g(n) + h(n) \in \Omega(f(n))$.

(c) Soit un arbre binaire dont le nombre de feuilles de tout sous-arbre de gauche diffère d'au plus un du nombre de feuilles du sous-arbre de droite. Cet arbre a une hauteur $\Theta(\log n)$ au pire cas, où n est le nombre total de nœuds.

(d) Le tri par fusion est $O(n^2)$ dans le pire cas.

(e) Si le facteur de charge d'une table de hachage est plus petit que 1, alors il n'y a pas de collisions.

Question 2 : Tri

(a) Décrivez l'algorithme de tri par tas dans le formalisme de votre choix. Donnez sa complexité dans le meilleur et le pire cas.

(b) Illustrez les différents étapes du tri du tableau $[17, 23, 10, 1, 7]$.

(c) Cet algorithme de tri est-il stable ? Est-il en place ? Justifiez vos réponses.

Question 3 : Analyse de complexité

On considére les trois algorithmes ci-dessous pour multiplier deux nombres $x = x_{n-1} \dots x_0$ et $y = y_{n-1} \dots y_0$ représentés sur n bits :

Méthode 0 : On initialise le produit p à 0. Pour j allant de 0 à $n-1$, si $x_i = 1$, on ajoute y décalé de i bits vers la gauche au produit p courant.

Méthode 1 : Soient x_L et y_L les $\lceil n/2 \rceil$ bits de poids les plus forts de x et y respectivement et soient x_R et y_R les $\lfloor n/2 \rfloor$ bits de poids les plus faibles de y, c'est-à-dire tels que

$$x = x_L \cdot 2^{\lfloor n/2 \rfloor} + x_R \text{ et } y = y_L \cdot 2^{\lfloor n/2 \rfloor} + y_R.$$

On calcule récursivement les 4 produits suivants (de tailles $n/2$) :
— $p_1 = x_L \times y_L$,
— $p_2 = x_L \times y_R$,
— $p_3 = x_R \times y_L$,
— $p_4 = x_R \times y_R$.
Le produit recherché s'obtient ensuite en calculant :

$$p = p_1 \cdot 2^{2 \cdot \lfloor n/2 \rfloor} + (p_2 + p_3) \cdot 2^{\lfloor n/2 \rfloor} + p_4.$$

Lorsque $n = 1$, le produit est calculé directement.

Méthode 2 : Avec les mêmes notations que la méthode 1, cette méthode calcule (récursivement) les 3 produits suivants :
— $p'_1 = x_L \times y_L$,
— $p'_2 = (x_L + x_R) \times (y_L + y_R)$,
— $p'_3 = x_R \times y_R$.
Le produit recherché s'obtient ensuite en calculant :

$$p = p'_1 \cdot 2^{2 \cdot \lfloor n/2 \rfloor} + (p'_2 - p'_1 - p'_3) \cdot 2^{\lfloor n/2 \rfloor} + p'_3.$$

Lorsque $n = 1$, le produit est calculé directement.

(a) Sur quelle technique de résolution de problème sont basées les méthodes 1 et 2 ?

(b) Donnez le pseudo-code d'une fonction MULTIPLY2(x, y) implémentant la méthode 2. Vous ne devez pas implémenter explicitement les opérations d'addition et de multiplication par une puissance de 2 (décalage).

(c) Etudiez les complexités aux pire et meilleur cas des trois méthodes. Justifiez vos résultats.
Hypothèses : Pour les méthodes 1 et 2, vous pouvez supposer que la taille n du tableau est une puissance exacte de 2. Pour la méthode 2, vous pouvez supposer que les sommes $x_L + x_R$ et $y_L + y_R$ peuvent toujours être représentées sur $n/2$ bits. Le calcul de la somme ou de la différence de deux nombres représentés sur n bits est $\Theta(n)$ et le produit d'un nombre par 2^n est $\Theta(1)$ (simple décalage de n bits).

(d) Discutez de l'intérêt pratique de ces différentes méthodes les unes par rapport aux autres.

Question 4 : Arbre binaire de recherche

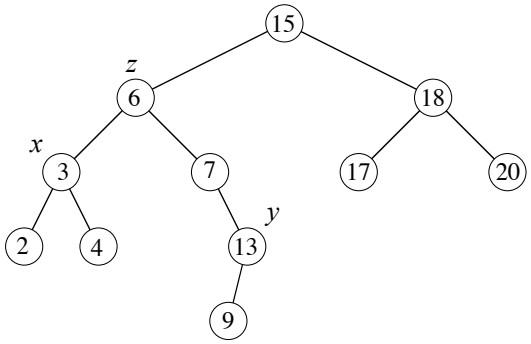
(a) Qu'est-ce qu'un arbre binaire de recherche ?

(b) Le parcours préfixe d'un arbre binaire de recherche donne la séquence suivante :

$$5, 1, 2, 11, 8, 7, 13, 12.$$

Dessinez un arbre correspondant à ce parcours. Cet arbre est-il unique ?

(c) Ecrivez une fonction GETLCA(T, x, y) renvoyant l'ancêtre commun le plus proche des deux nœuds x et y dans un arbre binaire de recherche T. L'ancêtre commun le plus proche de deux nœuds x et y est l'ancêtre z de x et y qui est le plus profond dans l'arbre. Pour cette définition, on considérera un nœud comme un ancêtre de lui-même. Par exemple, pour l'arbre ci-dessous, les deux appels GETLCA(T, x, y) et GETLCA(T, z, y) doivent renvoyer le nœud z.
(suggestion : tirez profit de la propriété d'arbre binaire de recherche)



(d) Analysez la complexité de votre fonction dans le pire et dans le meilleur cas en supposant que l'arbre contient N clés.

Question 5 : Résolution de problèmes

A partir d'un nombre n, on génère une séquence en enlevant un chiffre soit au début, soit à la fin de ce nombre, jusqu'à qu'il ne reste plus qu'un seul chiffre. La profondeur carrée $S(n)$ de n est définie comme le nombre maximum de carrés parfaits [1] qu'on peut obtenir au long d'une séquence à partir de n (y compris n). Par exemple, $S(32492) = 3$ car :

$$32492 \rightarrow \mathbf{3249} \rightarrow \mathbf{324} \rightarrow 24 \rightarrow \mathbf{4}$$

où 3249, 324, et 4 sont des carrés parfaits et il n'y a pas d'autres séquences partant de 32492 contenant plus de carrés parfaits.

(a) Expliquez le principe d'une approche force-brute pour résoudre le problème et donnez sa complexité.

(b) Décrivez un algorithme efficace basé sur la programmation dynamique pour calculer la profondeur carrée $S(n)$ d'un nombre n écrit comme un nombre à d chiffres $a_1a_2 \ldots a_d$. Précisez les équations de récurrence (y compris le/les cas de base) correspondant à votre solution. Analyser la complexité de votre algorithme en fonction de d.
(Suggestion : vous aurez besoin de remplir une table à deux dimensions)

1. Un nombre n est un carré parfait s'il existe un entier m tel que $n = m^2$.