Programmation avancée

Examen écrit, 18 août 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) $7^{\log_2 N}$ est $O(N^3 + N^2)$.

(b) Pour un ensemble de valeurs données, il existe un unique tas max.

(c) Un arbre binaire complet qui possède k feuilles a une profondeur $\Theta(\log_2 k)$.

(d) Il est impossible de construire un arbre binaire de recherche à partir d'un tableau quelconque contenant N clés en $\Theta(N)$ opérations.

(e) Il est impossible de répondre à l'interface d'une file avec une liste simplement liée.

Question 2 : Tri

(a) Adaptez le pseudo-code du tri par fusion pour trier une liste simplement liée.
Vous pouvez supposer que votre fonction de tri prend en argument directement un pointeur x vers le première élément de la liste à trier et la taille N de cette liste et que la liste est implémentée comme vu au cours (avec pour chaque élément de liste x la clé à trier notée $x.key$ et un pointeur vers l'élément suivant noté $x.next$). La fonction devra renvoyer un pointeur vers le premier élément de la liste triée. La liste de départ ne doit pas être préservée. Pour obtenir tous les points de cette question, votre fonction ne doit pas avoir besoin de créer d'espace mémoire supplémentaire (si ce n'est implicitement pour les appels récursifs) et ne doit pas modifier les valeurs de l'attribut $x.key$ des éléments de liste.

(b) Analysez la complexité de votre algorithme dans le pire et le meilleur cas.

Question 3 : Analyse de complexité

Soit la fonction suivante :

```
ALGO(n)
1  if n == 1
2        return 0
3  else
4        j = 0
5        i = 1
6        while i < n
7              j = j + 1
8              i = 2i
9        return 2 * (ALGO(n/2) + ALGO(n/2) + ALGO(n/2)) + 2^j
```

En supposant que cette fonction prend toujours une puissance de 2 comme argument :

(a) Donnez une forme analytique équivalente à cette fonction.

(b) Donnez une borne asymptotique sur sa complexité en temps (*Suggestion :* Exprimez cette complexité sous la forme d'une récurrence et résolvez la au moyen du *Master theorem*)

Question 4 : Table de hachage

Soit une famille de tables de hachage $\mathcal{T}$ de capacités initiales de $m = 2^p$ pour un certain entier p, doublant de taille lorsque son facteur de charge atteint 75% et où les collisions sont gérées par sondage ouvert avec la fonction

$$h(k, i) = \left(k + \frac{1}{2}(i + i^2)\right) \mod m$$

(a) Décrivez dans le formalisme de votre choix les fonctions d'insertion et de recherche dans cette table de hachage (en ne prenant pas en compte le redimensionnement de la table).

(b) Illustrez l'évolution d'une telle table de hachage $T \in \mathcal{T}$ avec une capacité initiale de 4 au fil de l'insertion des clés $[2, 8, 6, 9, 1, 3, 7]$.

(c) Comparez les arbres binaires de recherche et les tables de hachage en énumérant leurs avantages et défauts respectifs.

Question 5 : Résolution de problème

Soient n maisons placées en ligne, chacune contenant des biens pour une certaine valeur. Un voleur souhaite voler le maximum de valeurs dans ces maisons mais il ne souhaite cependant pas voler dans deux maisons trop proches pour éviter qu'une personne volée ne prévienne ses voisins. On notera $v[i]$ $(i = 1, \ldots, n)$ la valeur des biens à voler dans la maison i, $d[i]$ $(i = 1, \ldots, n-1)$ la distance entre les maisons i et $i+1$, et d_{th} la distance minimale entre deux maisons qui peuvent être volées simultanément. Les maisons i et j, avec $1 \leq i < j \leq n$, ne peuvent pas être volées simultanément si $\sum_{l=i}^{j-1} d[l] \leq d_{th}$.

(a) Ecrivez un algorithme efficace basé sur la **programmation dynamique** pour déterminer la valeur maximale totale, notée $M[n]$, que le voleur peut dérober sur les n maisons étant donné la valeur d_{th}. Procédez en deux étapes :
   — Donnez une formulation récursive de $M[n]$.
   — En déduire le pseudo-code d'un algorithme efficace pour calculer $M[n]$.

(b) Analysez la complexité de cet algorithme en fonction du nombre de maisons n.