Structures de données et algorithmes

Examen écrit, première session, 7 juin 2019

Livres fermés. Durée : 4h00.

Remarques

— Répondez à chaque question sur une feuille **séparée**, sur laquelle figurent vos noms, prénoms et matricules.
— 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 :

1. Soient deux algorithmes récursifs dont les complexités sont données respectivement par :
   — $T_1(n) = 2T_1(n/2) + 1$,
   — $T_2(n) = 2T_2(n/3) + \log(n)$.
   On a $T_1(n) \in O(T_2(n))$.
2. Soit une clé k dans un arbre binaire de recherche associée à une feuille. Soient les trois ensembles suivants : A, l'ensemble des clés à la gauche du chemin entre la racine et cette feuille, B, l'ensemble des clés sur ce chemin, et C, l'ensemble des clés à la droite de ce chemin. Soient trois clés $a \in A$, $b \in B$ et $c \in C$. On peut montrer que $a \leq b \leq c$.
3. Soit un table de hachage avec comme fonction de hachage $h(k) = (5 + 2k) \mod 13$ et gérant les collisions par sondage linéaire. Lors de l'insertion des clés suivantes dans cet ordre :
   $$24, 2, 5, 11, 20, 12$$
   deux clés nécessiteront un sondage.
4. Soit une file à priorité S implémentée en utilisant un tas-max. Il est possible d'ajouter une fonction Minimum(S) de complexité $O(1)$ renvoyant (sans l'extraire) l'élément de S avec la plus petite valeur de clé, en modifiant éventuellement les autres fonctions de l'interface mais en maintenant leur complexité initiale.

Question 2 : Tri et complexité

1. Montrez que le problème de tri comparatif est $\Theta(n \log n)$.
2. Soit un tableau A de n valeurs qui sont initialement triées par ordre croissant. Ce tableau est ensuite modifié de sorte que k de ses valeurs soient diminuées. Expliquez le principe de deux algorithmes satisfaisant aux contraintes suivantes :
   (a) Un algorithme triant A *en place* de complexité en temps $O(kn)$.
   (b) Un algorithme triant A de complexité en temps $O(n + k \log k)$, pas nécessairement en place.

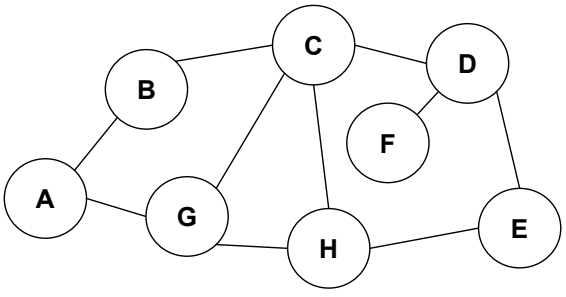
   Ne donnez de pseudo-code que si c'est nécessaire. Justifiez pour les deux algorithmes que les contraintes de complexité sont bien remplies.

Question 3 : Structures de données

1. Qu'est-ce qu'un arbre binaire de recherche ?
2. Donnez le pseudo-code de la fonction TREE-INSERT(T, z) qui permet d'insérer le noeud z à l'arbre binaire de recherche T.
3. Donnez le pseudo-code d'une fonction TREE-TO-LIST(T) qui permet de transformer l'arbre binaire de recherche T en un arbre binaire de recherche dégénéré où chaque noeud n'a qu'un fils à droite (*i.e.* une liste liée). Pour obtenir la cote maximale, votre fonction doit être de complexité linéaire par rapport aux nombres de noeuds de T et ne pas nécessiter d'espace mémoire supplémentaire, en dehors d'éventuels appels récursifs (aucun nouveau nœud ne doit être créé).
   Suggestion : n'oubliez pas qu'une fonction peut renvoyer plusieurs valeurs.

Question 4 : graphes

Soit le graphe ci-dessous, non dirigé et non pondéré :



1. Décrivez dans le formalisme de votre choix l'algorithme le *plus efficace* possible pour trouver le plus court chemin entre deux nœuds de ce graphe, la longueur d'un chemin étant mesurée par le nombre d'arêtes qui le composent. Donnez le nom de cet algorithme.
2. Le diamètre d'un graphe est la plus grande distance possible qui puisse exister entre deux nœuds, où la distance entre deux nœuds est définie par la longueur d'un plus court chemin entre ces deux nœuds. Le diamètre du graphe ci-dessus est 4 (Les nœuds A et F sont à une distance de 4).
   (a) Expliquez, sans donner de pseudo-code, comment exploiter ou adapter l'algorithme donné au point 1 pour calculer le diamètre d'un graphe tel que le graphe ci-dessus.
   (b) Donnez la complexité dans le pire cas de votre solution en fonction du nombre de nœuds $|V|$ et d'arêtes $|E|$ du graphe.
   Remarque : vous pouvez vous contenter de donner une borne supérieure, la plus serrée possible, en notation grand O.

Question 5 : résolution de problème

En tant qu'étudiant, vous êtes embauchés par une société de peinture en bâtiment pour peindre les persiennes d'une série de maisons dans un nouveau lotissement.

Les maisons sont situées sur une ligne et sont numérotées de 1 à N. Vous avez à votre disposition trois couleurs, rouge, vert et bleu, et vous avez comme consigne que deux maisons adjacentes ne peuvent pas avoir des persiennes de la même couleur.

Etant donné que le coût de la peinture est à votre charge, vous souhaitez minimiser celui-ci. Pour vous aider, vous avez estimé que peindre les persiennes de la maison i en couleur *color* coûterait $Cost[i, color]$ ($1 \leq i \leq N$, $color \in \{\text{rouge}, \text{vert}, \text{bleu}\}$). Votre objectif est donc de minimiser le coût total en peinture :

$$c^* = \min_{(color_1,\ldots,color_N)\in\{\text{rouge,vert,bleu}\}^N} \sum_{i=1}^{N} Cost[i, color_i]$$

1. Expliquez brièvement le principe d'une solution gloutonne pour résoudre ce problème et montrez à l'aide d'un contre-exemple (*i.e.* un choix particulier du tableau $Cost$) que cette approche n'est pas optimale.
2. Proposez une solution par programmation dynamique. Pour ce faire :
   (a) Formulez récursivement une fonction de coût M pertinente à minimiser.
   Suggestion : cette fonction devrait avoir deux arguments.
   (b) Déduisez-en le pseudo-code d'un algorithme efficace PAINT($Cost$) permettant de calculer c^*, le coût optimal des peintures, en fonction du tableau de coût $Cost$.
   (c) Analysez la complexité en temps et en espace de votre algorithme.