Université Paris 7 - Licence d'informatique - L2

EA4 – Examen du 19 Mai 2011 - durée : 2h30

La qualité de la rédaction sera prise en compte dans la notation. Justifiez toutes vos réponses et expliquez les fondements de vos algorithmes en français avant de les rédiger en pseudo-code compréhensible et commenté où nécessaire. Le barème est donné seulement à titre indicatif.
Les algorithmes donnés sans commentaires ou sans explications ne seront pas pris en compte par le correcteur. Documents non autorisés.

Exercice 1 (4 points).

Soit c_n le nombre d'arbres binaires à n nœuds. Trouver une formule récursive pour la suite $\{c_n\}$ et l'utiliser pour calculer c_2, c_3, c_4. On rappelle que $c_0 = c_1 = 1$.
Écrire ensuite un algorithme **itératif** qui retourne la valeur c_n en fonction de n et calculer sa complexité. La complexité en temps de votre algorithme devra être la plus petite possible et la complexité en espace ne devra pas dépasser $O(n)$.

Exercice 2 (6 points).

On a deux suites d'entiers $X = \{x_1, \ldots, x_n\}$ et $Y = \{y_1, \ldots, y_m\}$. Écrire un algorithme qui vérifie qu'un entier se trouve dans X **si et seulement si** il se trouve dans Y. Votre algorithme devra voir une complexité en temps $O(\max(n^2, m^2))$ mais pas $\Theta(\max(n^2, m^2))$.
Peut-on, avec des hypothèses additionnelles, obtenir un algorithme en $O(\max(n, m))$?
Dans l'affirmative, dites quelles sont ces hypothèses, fournissez l'algorithme et donnez sa complexité.
Répondre ensuite aux mêmes questions si on veut vérifier que X et Y contiennent les mêmes entiers et que tous ces entiers *apparaissent avec le même nombre d'occurrences* dans X et Y.

Exercice 3 (5 points).

Considérer des arbres binaires basés sur une structure nœud de type suivant :

```
struct noeud {int val ; struct noeud * g ; struct noeud * d ;} ;
```

où `g` et `d` sont des pointeurs. L'absence d'un sous-arbre est indiquée par le pointeur `NULL`.
On a vu en cours des procédures récursives de parcours d'arbre (infixe, préfixe, postfixe). Quelle est la complexité en espace de ces algorithmes ?
Ici on se propose de parcourir l'arbre **par niveaux** (l'on parcourt d'abord la racine, puis les nœuds de profondeur 1 de gauche vers droite, puis ceux de profondeur 2 et ainsi de suite) et de le faire de manière **non récursive (itérative)**.
Écrivez un algorithme non récursif qui associe au champ `val` de chaque nœud la position à laquelle il sera rencontré pendant un parcours par niveaux. L'espace mémoire maximal autorisé est $O(n)$, vous donc pouvez utiliser des structures auxiliaires dont vous détaillerez les attributs et les méthodes.

Exercice 4 (5 points)

Construire le tas-max T obtenu par la procédure suivante.
Étape 1. Stocker dans un ABR A les entiers

$$47, 11, 24, 66, 34, 53, 75$$

Étape 2. Sortir ensuite les éléments un par un de l'ABR A en choisissant toujours celui à la racine et les insérer un par un dans le tas-max T.
Puisqu'au moins deux stratégies sont possibles pour supprimer un nœud d'un ABR, indiquez celle que vous choisissez. Détaillez toutes les étapes avec des dessins.