Algorithmique

Feuille suplementaire

Eric Colin de Verdiere, Yan Jurski, Sylvain Peyronnet

Janvier 2003

Les arbres que l'on considère sont des arbres étiquetés par des entiers.

On appelle un arbre binaire quasi parfaits (ABQP) un arbre binaire qui vérifient les propriétés suivantes:

- tous les sommets ont 0 ou 2 fils, sauf éventuellement un seul sommet qui n'a qu'un fils;
- pour un arbre de hauteur h, toutes les feuilles sont à la distance h ou $h-1$ de la racine;
- les feuilles à distance h sont les plus à gauche possible, les feuilles à distance $h-1$ les plus à droite possible.

On considère un ABQP à n feuilles ($n \geq 1$). Donnez en fonction de n:

1. le nombre de sommet (feuilles incluses) de cet arbre;
2. la hauteur de l'arbre.

   On associe à un ABQP $\mathcal{A}$ le tableau $T(\mathcal{A})$ de la manière suivante

   - la dimension de $T(\mathcal{A})$ est le nombre de sommets de $\mathcal{A}$.
   - la racine de $\mathcal{A}$ est à l'indice 0 dans $T(\mathcal{A})$.
   - le sommet à l'indice i dans $T(\mathcal{A})$ a pour fils les sommets aux indices $2i+1$ et $2i+2$.

   Soit $\mathcal{A}$ un ABQP.

3. Dessiner l'ABQR $\mathcal{A}_1$ associe au tableau suivant

   $T(\mathcal{A}_1) =< 12, 34, 6, 4, 8, 41, 45, 98, 67, 28 >$
4. Si $\mathcal{A}$ a un sommet qui n'a qu'un fils, quel est dans $T(\mathcal{A})$, l'indice de ce sommet?
5. Écrire une procédure qui affiche les feuilles de $\mathcal{A}$ à partir de la feuille de plus petite hauteur la plus à gauche (jusqu'à la feuille de plus grande hauteur la plus à droite)
6. Quelle est la complexité de la procédure précédente?

Septembre 2003

On considère un arbre binaire dont les noeuds contiennent des nombres entiers.

1. Ecrire une fonction qui teste si toutes les valeurs des noeuds d'un arbre binaire appartiennent à l'intervalle $[1..N]$, pour un entier N fixé.
2. Quelle est la complexité de cette fonction dansle pire et le meilleur des cas ?
3. Ecrire une fonction qui retourne `vrai` si l'arbre binaire vérifie la condition *les noeuds de l'arbre ont pour valeur ds entiers compris entre 1 et 100, et les valeurs de ces noeuds sont toutes distinctes.* (On pourra utiliser un tableau auxilliaire)

Halloween

On dispose d'un fichier historique de 200 000 fiches, chaque fiche comportant le nom, la date de naissance et la date de décès d'un homme ou d'une femme célèbre. Ce fichier est un tableau `Tab` trié par ordre croissant des dates de naissance, et on désire obtenir les mêmes données, mais dans l'ordre des dates de décès. On sait qu'il n'y a jamais eu plus de 2 000 hommes ou femmes célèbres qui aient vécu en même temps.

1. Écrire en pseudo-langage un algorithme efficace pour résoudre ce problème, en utilisant un tableau `TabAux` pouvant contenir 2 000 fiches. On supposera avoir défini:

   - un objet `date` contenant n'importe quelle date; on admet pouvoir comparer deux dates (par exemple, en pseudo-langage:
     `SI (date_1 < date_2) ALORS...`);
   - un objet `fiche` qui contient un champ `nom` (de type chaîne de caractères), et deux champs de type `date` appelés `naissance` et `décès`.
     On pourra librement utiliser l'affectation des fiches (par exemple
     `TabAux[j] <- Tab[i]` ou `Ma_Fiche <- Mon_Autre_Fiche`).

2. Évaluer la complexité de l'algorithme en nombre de comparaisons de dates et d'affectations de fiches, dans le cas le pire.

Petit jeu

On dispose d'un tableau d'entiers indexé par [0..N], et on voudrait que la case d'indice i contienne le nombre de i dans l'ensemble du tableau.

Par exemple si N vaut 3 le tableau $t = [1, 2, 1, 0]$ est convenable :
$t[0]$ vaut 1, et il y a bien une occurrence de 0 dans le tableau, il y a deux 1 comme

le confirme la valeur de $t[1]$, il y a un 2 et aucun 3, comme l'indiquent les cases $t[2]$ et $t[3]$.

La proposition $[2, 1, 0, 0]$ est elle incorrecte, puisqu'il y a un 2 mais que $t[2] = 0$.

- Que pensez vous de $[2, 1, 2, 0, 0]$?
- Écrivez un programme qui prend en entrée un tableau et vérifie sa validité.

Pour un tableau indexé par $[0..6]$ une solution correcte est $[3, 2, 1, 1, 0, 0, 0]$. Plus généralement pour quels N existe t'il une solution ?

problème de tableau

soit t un tableau d'entiers trié, indicé de 1 à n. (Les éléments peuvent être négatifs, on pourra les supposer touts différents). Donner un algorithme, le plus efficace possible, qui trouve un indice i pour lequel $t[i] = i$, s'il existe. Quelle est sa complexité dans le pire et le meilleur des cas ?