Option Informatique
Exercices sur les arbres

Sujet

30 mars 2007

1 Arbres peignes

Soit $\mathcal{E}$ un ensemble et $T(\mathcal{E})$ la collection des arbres (stricts) étiquetés par $\mathcal{E} = \mathcal{N} \cup \mathcal{F}$. On définit une classe $\mathcal{P}$ d'arbres par induction. Un arbre T est dans $\mathcal{P}$ si :
– $T = r$ où $r \in \mathcal{F}$
– $\exists T' \in \mathcal{P}, r \in \mathcal{F}, a \in \mathcal{F},\ (T = (a, r, T')) \vee (T = (T', r, a))$.
Les éléments de $\mathcal{P}$ seront appelés arbres peignes.

Question 1

a Donner des exemples d'arbre peigne.

b Montrer que l'on a l'équivalence : T est un arbre peigne ssi. hauteur(T) = taille(T), *où la* taille *est le nombre de noeuds internes de T.*

c que devient cette relation avec le nombre de feuilles de T puis avec le nombre total de noeuds de T ?

d On définit le type des arbres par :

```
type ('n,'f) arbre =
  feuille of 'f
  | noeud of ('n,'f) arbre * 'n * ('n,'f) arbre;;
```

Écrire une fonction CAML `est_peigne` *de type* `('n,'f) arbre → bool`

2 Recherche du minimum et du maximum d'un arbre T

On veut déterminer le maximum et le minimum d'un arbre T étiqueté par $\mathcal{E}$ à l'aide d'une fonction `min_max` qui renvoie le couple $\min(T), \max(T)$. On suppose prédéfinies deux constantes `MaxE` et `MinE` qui sont le minimum et le maximum de $\mathcal{E}$.

Question 2 *Donner une version* CAML *de* `min_max`*. On précisera le type arbre utilisé.*

3 Parcours en largeur d'abord

Les parcours préfixe, infixe, suffixe d'un arbre sont dit en profondeur d'abord : on descend le long d'une branche tant que c'est possible. On veut à présent faire un parcours en *largeur d'abord*. Dans un tel parcours, on traite la racine, puis ses fils, puis les fils de ces fils, etc.

On se place dans le cadre des arbres binaires stricts. On notera donc `fonc_f` et `fonc_n` les fonctions à appliquer aux feuilles et aux noeuds de l'arbre. On notera T l'arbre initial à traiter.

On utilise une liste d'arbres que l'on notera `L`. `L` doit contenir à chaque étape la liste des sous-arbres restant à traiter.

Question 3

a Que doit valoir `L` initialement ?

b On suppose que `L` contient la liste des arbres restant à traiter. On isole la tête de `L` : c'est l'arbre qui doit être traité. On le note (G, x, D). Que doit-on fair epour le traitement, la réactualisation de `L`, et la poursuite du parcours en largeur d'abord ? Proposer une structure plus adéquate pour `L` qu'une liste.

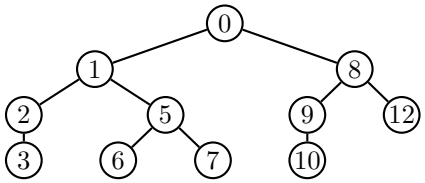
c Quand arrête-t-on le traitement ?

d En déduire une fonction récursive `traite_L fonc_f fonc_n` qui effectue le traitement en largeur d'abord des arbres de `L`.

e En déduire une fonction `largeur fonc_f fonc_n` qui effectue le traitement en largeur d'abord de `T`.

f En déduire une instruction CAML *d'affichage des étiquettes des noeuds de l'arbre en largeur d'abord. On supposera $\mathcal{N} = \mathbb{R}$, et $\mathcal{F} = \mathbb{N}$.*

Exemple :



Parcouru en largeur, cet arbre donne `[0;1;8;2;5;9;12;3;6;7;10]`.

4 Arbres et expressions algébriques

On appelle expression arithmétique une suite de nombres entiers et d'opérateurs $+, *, -, /$. on considère le sous-ensemble composé des expressions arithmétiques postifixées (EAP).

Soit une EAP de longueur n représentée par une liste L de taille n. Un élément de L est soit un nombre a, soit un opérateur e. on définit inductivement l'arbre $T(L)$ par
- Si $L = a$ alors $T(L)$ est une feuille dont la valeur est a.
- Si $L = L'L''e$, où L' et L'' sont deux EAP, alors la racine de $T(L)$ est e, le fils gauche est $T(L')$ et le fils droit est $T(L')$.

Question 4 *a Définir les types* `liste_EAP` *et* `arbre_EAP` *associés à L et $T(L)$.*

b Écrire une fonction `prefixe L` *qui renvoie un couple L, L' où L' est le plus long préfixe strict de L de poids égal à 1 (avec les poids vus en cours), et L'' est le reste de la liste L (i.e. $L = L'L''$).*

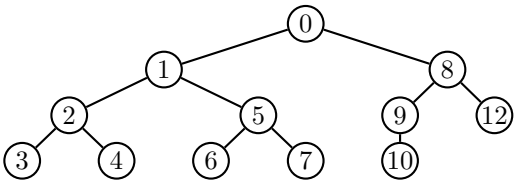
c Écrire une fonciton `liste2arbre L` *qui renvoie $T(L)$. Cette fonction s'arrêtera si L ne représente pas une EAP.*

d Écrire une fonction `arbre2listepre T` *qui renvoie la liste de l'EA préfixée associée à T.*

e Écrire une fonction `listpost2listpre L` *qui convertit une expression postfixée en expression préfixée.*

5 Arbres équilibrés

Un arbre binaire est dit *quasi-complet* si c'est un arbre binaire complet dont il manque éventuellement des noeuds sur le dernier rang.



Un arbre binaire est dit *H-équilibré* si pour tout noeud N de l'arbre, les hauteurs du sous-arbre gauche et du sous-arbre droit de N diffèrent d'au plus 1.

Question 5 *Rappeller la fonction* `hauteur` *qui calcule la hauteur d'un arbre. on précisera le type arbre utilisé.*

Question 6 *La profondeur d'un arbre est la longueur (en nombre d'arêtes) du plus court chemin de la racine à une feuille. Implémenter une fonction* `profondeur` *qui renvoie ce résultat.*

Question 7 *Implémenter une fonction* `est_quasicomplet` *qui vérifie si un arbre est quasi-complet.*

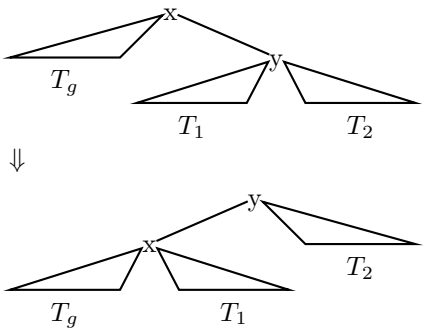
Question 8 *Implémenter une fonction qui vérifie si un arbre binaire est un arbre H-équilibré. Déterminer la complexité de votre algorithme, vous paraît-elle raisonnable ?*

6 Rotations sur un arbre binaire

Soit $T = (T_g, x, T_d)$ un arbre binaire. On suppose que T_d est non vide. On note $T_d = (T_1, y, T_2)$. L'opération G de *rotation gauche* sur l'arbre T produit un arbre T' défini par :

$$T' = G(T) = G([T_g, x, (T_1, y, T_2)]) = ([T_g, x, T_1], y, T_2)$$

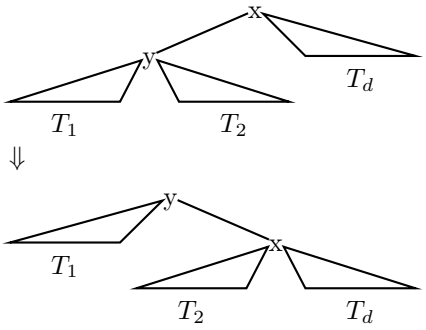
On peut représenter cette information par :



L'opération D de rotation droite sur l'arbre T produit un arbre T' défini par :

$$T' = D(T) = D([(T_1, y, T_2), x, T_d]) = (T_1, y, (T_2, x, T_d))$$

On peut représenter cette information par :



Question 9

a Définir ces deux opérations en CAML*, évaluer leur complexité.*

b On définit la double rotation gauche-droite *par :*

$$GD[(T_g, x, T_d)] = D[G(T_g), x, T_d]$$

et la double rotation droite-gauche *par :*

$$DG[(T_g, x, T_d)] = G[T_g, x, D(T_d)]$$

Illustrer sur un dessin ce que font ces opérations. Donner leur complexité.

*c Montrer que ces opérations conservent la structure d'*arbre binaire de recherche.

d À l'aide de ces opérations, expliquer comment effectuer la supresison dans un arbre binaire de recherche.