Option Informatique

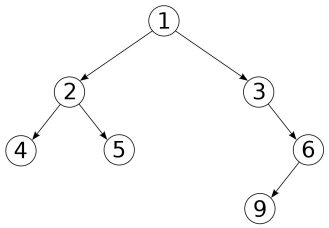
TD $n°03$ – Pratique et théorie sur les arbres binaires

Pour pouvoir copier coller : énoncé disponible à l'adresse http://mpsi.daudet.free.fr/info/TD/TD03.pdf.

On définit le type `arbre_binaire` comme suit :

```
type 'a arbre_binaire = Vide | Noeud of 'a * 'a arbre_binaire * 'a arbre_binaire ;;
```

Exemple : `Noeud(1,Noeud(2,Noeud(4,Vide,Vide),Noeud(5,Vide,Vide)),Noeud(3,Vide,Noeud(6,Noeud(9,Vide,Vide),Vide)))`

peut se représenter : 

; on ne représente pas les sous-arbres vides.

Il y a deux types d'arbres : l'arbre vide et les arbres non vides qui sont tous de la forme `Noeud(r,fg,fd)` où `r : 'a` et `fg, fd : 'a arbre_binaire`, auquel cas `r` s'appelle l'**étiquette** de l'arbre, `fg` s'appelle le **fils gauche** de l'arbre et `fd` s'appelle le **fils droit** de l'arbre.

Q1 Écrire en Caml des fonctions `etiquette : 'a arbre → 'a`, `fils_gauche : 'a arbre → 'a arbre` et `fils_droit : 'a arbre → 'a arbre`. Elle devront provoquer une exception en dehors de leur domaine de définition.

On définit récursivement la hauteur $h(a)$ d'un arbre a en distinguant deux cas : si a est `Vide` alors $h(a) = -1$, sinon a a un fils gauche fg et un fils droit fd, et alors $h(a) = \max\big(h(fg), h(fd)\big) + 1$.

Ainsi : la hauteur l'arbre représenté dans l'introduction est 3.

Q2 Écrire en Caml la fonction `hauteur : 'a arbre → int`.

Pour a un arbre, on appelle **sous-arbre de a**, ou **nœud de a** un arbre non vide obtenu à partir de a en effectuant un nombre arbitraire de fois les opérations `fils_gauche` et `fils_droit` (sans obtenir d'exception). Le nœud a lui-même (obtenu en effectuant zéro fois les opérations) est appelé la **racine** de a. Un nœud de a dont les deux fils sont vides est appelé **une feuille**.

Ainsi : Les nœuds de l'arbre représenté dans l'introduction sont étiquetés par 1, 2, 3, 4, 5, 6 et 9. Les feuilles de cet arbre sont étiquetées par 4, 5 et 9. La racine est étiquetée par 1. La racine est dessinée en haut et les feuilles en bas, c'est l'usage (je préfère ne pas savoir où poussent ces arbres).

Q3 Écrire en Caml la fonction `liste_feuille : 'a arbre → 'a list` permettant d'obtenir la liste des étiquettes des **feuilles** de son argument. La complexité de cette fonction pourra être sous-optimale et en particulier on n'hésitera pas à faire usage de la concaténation @.

Pour a un arbre, on appelle **taille de a**, et on note $|a|$, le nombre de nœuds de a.

Ainsi : l'arbre représenté dans l'introduction est de taille 7.

Q4 Écrire en Caml la fonction `taille : 'a arbre → int`.

On rappelle (on appelle ?) comment démontrer qu'une propriété est vérifiée sur les arbres par induction structurelle. Notons $\mathcal{P}(a)$ l'énoncé indiquant que la propriété est vérifiée pour un arbre a. Si :

— $\mathcal{P}(\texttt{Vide})$ est vraie ;

— pour tout couple d'arbres (fg, fd), pour toute étiquette r, $\Big(\mathcal{P}(fg) \text{ et } \mathcal{P}(fd)\Big) \Rightarrow \mathcal{P}\big(\texttt{Noeud}(r, fg, fd)\big)$;

alors pour tout arbre a, la propriété $\mathcal{P}(a)$ est vraie.

Q5 Montrer par induction structurelle que pour tout arbre a on a $h(a) < |a| < 2^{h(a)+1}$.
En déduire un encadrement de $h(a)$.

Q6 Pour quels types d'arbres a-t-on $|a| = h(a) + 1$?
Pour quels types d'arbres a-t-on $|a| = 2^{h(a)+1} - 1$?

On appelle **profondeur d'un nœud** le nombre de fois où on a dû appliquer l'une des opérations `fils_gauche` ou `fils_gauche` à partir de l'arbre initial pour obtenir ce nœud.

Ainsi : l'arbre représenté dans l'introduction a pour seul nœud de profondeur 0 la racine de l'arbre étiquetée par 1, et a trois nœuds de profondeur 2, étiquetés par 4, 5 et 6.

Q7 Pour tout arbre a, notons $N(a)$ l'ensemble des nœuds de a. Pour $n \in N(a)$, notons $p(n)$ la profondeur de n.
Montrer par induction structurelle que pour tout arbre **non vide** a on a $h(a) = \max\limits_{n \in N(a)} p(n)$.
Remarque : si on remplace `max` par `sup` et qu'on se place dans l'ensemble ordonné $\mathbb{N} \cup \{-1\}$, c'est même vrai pour l'arbre vide.

Q8 Soit $n \in N(a)$ tel que $h(a) = p(n)$. Que dire de n ? La réciproque est-elle vraie ?

Un **nœud interne** d'un arbre a est un nœud de a qui n'est pas une feuille. Enfin, on dit qu'un arbre binaire non vide a est **strict** lorsque tous ses nœuds ont 0 ou 2 fils non vides, c'est-à-dire lorsque tous ses nœuds internes ont leurs deux fils non vides.

Q9 Montrer par induction structurelle que pour tout arbre binaire non vide **strict** a, si a a n nœuds internes, alors a a $n + 1$ feuilles.