Agrégation de Mathématiques option Informatique
TP3

Loïg Jezequel

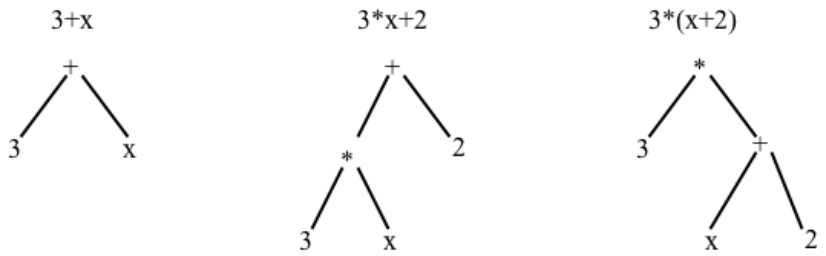
Mardi 3 Janvier 2012

Exercice 1 : arbres d'expressions

Dans cet exercice nous considérons la syntaxe suivante pour les expressions (avec une seule variable) :

```
expr ::= x | entier | expr + expr | expr * expr
```

De telles expressions peuvent se représenter sous forme d'arbres. Par exemple :



Question 1. Écrivez une fontion `expr2func` qui prend en entrée un arbre d'expression et rend la fonction Caml correspondante. La fonction `expr2func` aura le type suivant : `expr -> int -> int`.

Question 2. Écrivez une fonction `derive` qui dérive une expression par rapport à `x`. La fonction `derive` aura le type suivant : `expr -> expr`.

Question 3. La fonction `derive` retourne parfois des expressions inutilement compliquées. Par exemple : `x*0` au lieu de `0`. Proposez une méthode pour simplifier les expressions. Écrivez une fonction `simplifie` qui simplifie une expression.

Exercice 2 : arbres binaires de recherche

Si A est un arbre binaire non vide, on note Ag et Ad ses sous-arbres gauche et droit. Pour tout sommet x, on note A(x) le sous-arbre de racine x, et $Ag(x)$ et $Ad(x)$ les sous-arbres gauche et droit de $A(x)$. Un arbre binaire de recherche (ABR) est un arbre muni d'une fonction clé c de l'ensemble des sommets vers l'ensemble des entiers naturels, vérifiant :

$$c(y) < c(x) < c(z)$$

pour tout sommet x et tous sommets y de $Ag(x)$ et z de $Ad(x)$.

Question 4. Écrivez une fonction qui insère un entier dans un ABR (en préservant sa structure d'ABR).

Question 5. Écrivez une fonction qui transforme un ABR en une liste triée d'entiers.

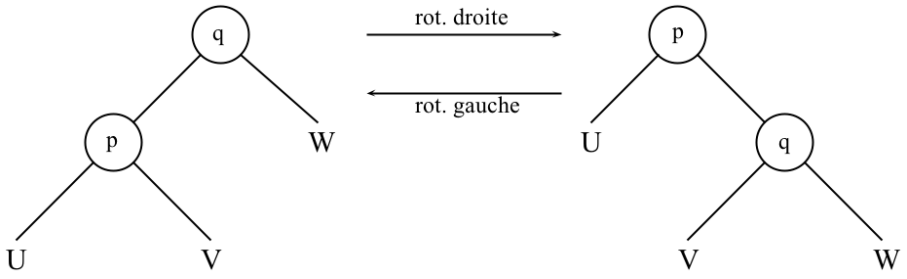
Un arbre binaire A est AVL si, pour tout sommet x de l'arbre, les hauteurs des sous-arbres $Ag(x)$ et $Ad(x)$ diffèrent d'au plus 1.

Question 6. Écrivez une fonction qui détermine si un ABR est AVL.

Exercice 2bis : un peu plus loin sur les arbres AVL

(pour ceux qui ont fini le TP en avance)

Si on veut préserver la structure AVL d'un arbre il est nécessaire, lors de l'ajout ou de la suppression d'un noeud, de vérifier si cela désequilibre l'arbre et, le cas échéant de le réequilibrer. Le réequilibrage se fait à l'aide des opérations (rotations) suivantes :



Question. Écrivez deux fonctions – `rotG` et `rotD` – qui implémentent ces opérations.

Question. Écrivez une fonction qui insère un élément dans un arbre AVL.

Exercice 3 : parcours d'arbres

Un parcours d'arbre générique peut se décrire de la manière suivante : on a un ensemble de nœuds, contenant initialement la racine de l'arbre uniquement. À chaque étape du parcours on retire un élément de l'ensemble puis on ajoute tous ses fils à l'ensemble. Lorsque l'ensemble est vide le parcours est terminé. Selon la structure de l'ensemble (file ou pile) ce parcours sera un parcours en largeur ou en profondeur.

Question 7. Écrivez une fonction qui génère aléatoirement un arbre à n nœuds étiquetés par des entiers compris entre 1 et n.

Question 8. Définissez un type pour les ensembles et implémentez les opérations de base sur ces ensembles : obtenir un élément d'un ensemble, vérifier la vacuité d'un ensemble, ajouter un élément dans un ensemble, créer l'ensemble vide.

Question 9. Écrivez une fonction générique de parcours d'un arbre, basée sur votre implémentation des ensembles.

Question 10. Modifiez votre implémentation des ensembles de façon à ce que votre fonction de parcours des arbres réalise un parcours en profondeur, puis un parcours en largeur.

Question 11. Améliorez votre fonction de génération d'arbres de façon à assurer que chaque entier compris entre 1 et n est présent une et une seule fois dans l'arbre généré.