DEUG MIAS 2e année

TD de programmation fonctionnnelle n°3

Les arbres (suite)

Pierre BOULET Jean-Marc NICOD Frédéric VIVIEN

Le but de ce TD-TP est d'utiliser la structure d'arbre pour trier une liste de valeurs. Pour cela, on utilise la construction d'un arbre de recherche. *Dans un souci de simplification et de généralisation, les arbres seront représenté par une liste. Le premier élément de cette liste est l'étiquette et la suite est la liste des fils. Un arbre binaire est donc un arbre donc chaque noeud est une liste à trois éléments.*

TD : Tri avec les arbres de recherche

Définition

On définit un arbre de recherche de la manière suivante : l'étiquette de tout nœud ou de toute feuille appartenant au sous arbre **gauche** d'un arbre de recherche est inférieure à la valeur de l'étiquette de cet arbre; au contraire, l'étiquette de tout nœud ou de toute feuille appartenant au sous arbre **droit** d'un arbre de recherche est supérieure ou égale à la valeur de l'étiquette de cet arbre. Cette définition s'applique à tous les sous-arbres d'un arbre de recherche. La figure 1 est un exemple d'arbre de recherche.

1 Construction d'un arbre de recherche

Question 1.1 : Les fonctions *make-arbre*, *make-feuille*, *etiquette*, *fg*, *fd* et *hauteur* sont données ici pour la représentation que nous avons adoptée pour les arbres binaires.

```
(define make-arbre
  (lambda (etiq fg fd)
    (list etiq fg fd)))
```

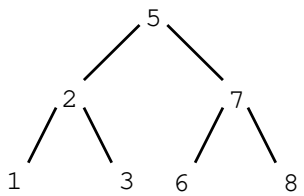

FIG. 1 - *Définition d'un* arbre de recherche

```
(define make-feuille
  (lambda (etiq)
    (list etiq '() '())))

(define etiquette
  (lambda (a)
    (if (null? a)
        '()
        (car a))))

(define fg
  (lambda (a)
    (if (null? a)
        '()
        (cadr a))))

(define fd
  (lambda (a)
    (if (null? a)
        '()
        (caddr a))))

(define hauteur
  (lambda (a)
    (if (null? a)
        0
        (+ 1 (max (hauteur (fg a)) (hauteur (fd a)))))))
```

Question 1.2 : Ecrire la fonction *insere* qui permet l'insertion d'un élément dans un arbre de recherche.

Question 1.3 : Ecrire la fonction *make-arbre-rech* qui permet la création d'un arbre de recherche à partir d'une liste de valeurs. Etant donné que le premier élément de la liste s'insère dans un arbre de recherche vide, la récurcivité peut être placée soit dans une autre fonction, soit dans la fonction *make-arbre-rech*, dans un **letrec**.

2 Utilisation des arbres de recherche

Question 2.1 : Ecrire la fonction *in-arbre-rech?* qui permet de chercher un élément dans un arbre de recherche. Il faut exploiter la manière dont sont organisées les informations dans un tel arbre pour que cette recherche soit la plus rapide.

Question 2.2 : Ecrire une fonction *tri-liste* qui permet de trier une liste de valeurs dans l'ordre croissant. Pour cela, utilisez les fonctions qui ont été écrites précédemment.

3 Equilibrage des arbres de recherche

Suivant l'ordre dans lequel les valeurs de la liste à trier sont placées, les arbres de recherche peuvent être très différents. Le cas le plus flagrant est la construction d'un arbre de recherche à

partir d'une liste déjà triée. Dans ce cas, l'arbre de recherche ressemble plus à une liste chaînée qu'à un arbre binaire. Or plus une valeur doit être insérée profondément dans l'arbre, plus cette insertion est longue. Il est donc intéressant de pouvoir équilibrer des arbre de recherche ou de savoir les construire directement équilibrés.

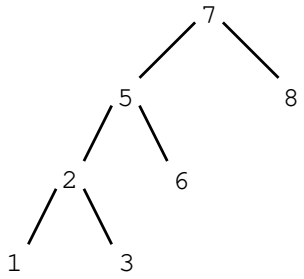

FIG. 2 - *Exemple d'un* arbre de recherche *non équilibré*

La figure 2 illustre le cas d'un arbre de recherche non équilibré. Dès que la différence en les hauteurs des deux branches d'un sous-arbre est supérieure ou égale à 2, il est possible d'effectuer une *rotation*. La figure 3 montre comment se fait une *rotation gauche*.

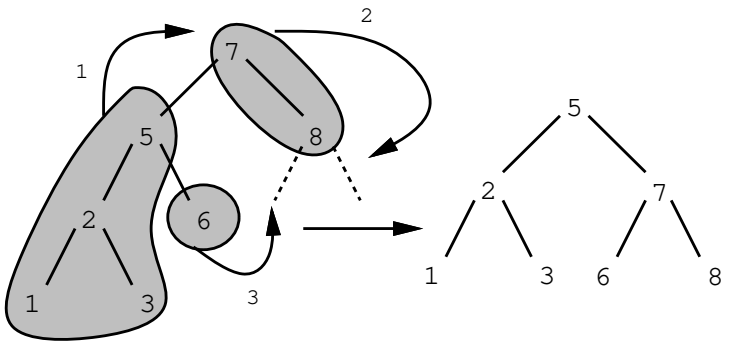

FIG. 3 - *Exemple d'un* arbre de recherche *non équilibré*

Question 3.1 : Ecrire les fonctions *rotation-g* et *rotation-d* qui permettent de rétablir l'équilibre entre les deux branches d'un sous arbre déséquilibré.

Question 3.2 : Ecrire la fonction *equilibre-arbre-rech* qui permet d'équilibrer un arbre de recherche créé par la fonction *make-arbre-rech* à partir d'une liste de valeur.

Question 3.3 : Ecrire la fonction *insere-eq* qui permet d'insérer une valeur dans un arbre de recherche équilibré; l'arbre de recherche résultat est équilibré. C'est la fonction *insere-eq* qui sera utilisée à l'avenir dans la fonction *make-arbre-rech*. Les rotations se font en remontant de la feuille d'insertion jusqu'à la racine, si nécessaire. Il faut utiliser la fonction *eq-ssarbre* dans la fonction *insere-eq*. La fonction *eq-ssarbre* est la version non récurcive de *equilibre-arbre-rech*.

Question 3.4 : Ecrire une fonction qui donne comme résultat la liste triée de la liste de valeur donnée en paramètre.