INFO0054 - Programmation fonctionnelle

Répétition 6: Graphes et listes profondes

Jean-Michel Begon

10 Mars 2016

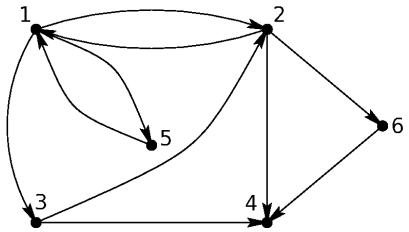
Graphes

Exercice 1.

Nous convenons de représenter un *graphe orienté* comportant n nœuds par une liste de n listes ; la liste correspondant au nœud x a pour premier élément x et pour éléments suivants les successeurs (immédiats) de x.

Écrire le prédicat `(path? n1 n2 g)` prenant deux nœuds $n1$ et $n2$ et la représentation g d'un *graphe orienté* en arguments et retournant `#t` s'il existe un chemin de $n1$ vers $n2$ dans g et `#f` sinon.

Par exemple le graphe



sera représenté par la liste

```
'((1 2 3 5) (2 1 4 6) (3 2 4) (4) (5 1) (6 4))
```

Exercice 2.

Écrire une fonction `inv` qui, à partir d'un graphe orienté, construit le graphe retourné correspondant.

Récursion profonde sur les listes

Exercice 3.

Définir la procédure `count-all` à deux arguments, un élément et une liste, et qui compte, en profondeur, le nombre de de fois que l'élément est contenu dans la liste.

```
(count-all 1 '(0 (1 2 (3 4 (1)) (3 (2 1) 1) 1) 0 (1 2 (1 2 3)))) ⇒ 7
```

Exercice 4.

Écrire une fonction `lpref` prenant comme argument une liste `u` et retournant la liste des préfixes de `u`. (La liste vide et la liste `u` elle-même sont des préfixes de `u`.)

Exercice 5.
Soit un alphabet décrit par une liste de symboles. Écrire une fonction `words` qui engendre la liste (dans un ordre quelconque) des mots de longueur n écrits dans cet alphabet.

```
(words 2 '(a b c)) ==>
((a a) (b a) (c a) (a b) (b b) (c b) (a c) (b c) (c c))
```

Exercice 6.
Écrire une fonction qui renvoie la liste des sous-ensembles d'un ensemble donné.

`(lset '(a b c))` ⇒ `(() (a) (a b) (a b c) (a c) (b) (b c) (c))`

Exercice 7.
Écrire une fonction qui renvoie la liste des permutations d'une liste donnée.

Exercice 8.
Une *tricoupure* d'une liste ℓ est une liste de trois listes non vides dont la concaténation (dans l'ordre) vaut ℓ. Écrire une fonction `tricoup` qui à toute liste ℓ associe la liste des tricoupures de ℓ.

Par exemple, si ℓ est `(a b)` la liste des tricoupures de ℓ est la liste vide ; si ℓ est `(a b c d)` la liste des tricoupures de ℓ comporte, dans un ordre quelconque, les trois listes `((a b) (c) (d))`, `((a) (b c) (d))` et `((a) (b) (c d))`.

Variante

Une *tricoupure* d'une liste ℓ est une liste de trois listes dont la concaténation (dans l'ordre) vaut ℓ. Écrire une fonction `tricoup-lv` qui à toute liste ℓ associe la liste des tricoupures de ℓ.

Par exemple, si ℓ est `(a)` la liste des tricoupures de ℓ comporte, dans un ordre quelconque, les trois listes `((a) () ())`, `(() (a) ())` et `(() () (a))`.