INFO0054 - Programmation fonctionnelle

Répétition 5: diviseurs et arbres

Jean-Michel Begon

10 Mars 2016

Les diviseurs

Exercice 1.
Un entier naturel est *parfait* s'il est égal à la somme de ses diviseurs positifs propres. Ecrire un prédicat `perfect?` déterminant si son argument est un nombre parfait.

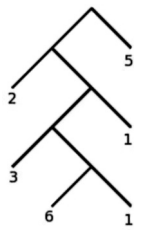
Le nombre 28 est parfait parce que $28 = 1 + 2 + 4 + 7 + 14$;
le nombre 32 n'est pas parfait parce que $32 \neq 1 + 2 + 4 + 8 + 16$.

Exercice 2.
Écrire un prédicat `prime?` qui détermine si un entier strictement positif est premier ou non.

Les arbres binaires complets

Exercice 3.
Ecrire une fonction `depth-first` qui prend comme argument un arbre binaire complet — chaque nœud a 0 (une feuille) ou 2 fils (un nœud interne) — dont uniquement les feuilles sont étiquetées, et renvoie la liste des étiquettes des feuilles, obtenue par un parcours en profondeur d'abord, et de gauche à droite, de l'arbre.

Par exemple, `depth-first` appliquée à l'arbre



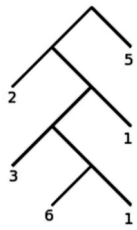
renvoie la liste `(2 3 6 1 1 5)`.

On choisira et spécifiera une représentation adéquate des arbres.

Exercice 4.

Ecrire une fonction `breadth-first` qui prend comme argument un arbre binaire complet — chaque nœud a 0 (une feuille) ou 2 fils (un nœud interne) — dont uniquement les feuilles sont étiquetées, et renvoie la liste des étiquettes des feuilles, obtenue par un parcours en largeur d'abord, et de gauche à droite, de l'arbre.

Par exemple, `breadth-first` appliquée à l'arbre



renvoie la liste `(5 2 1 3 6 1)`.

On choisira et spécifiera une représentation adéquate des arbres.

Variante : même exercice avec un arbre non plus binaire mais avec un nombre quelconque de fils, dont tous les nœuds sont étiquetés.

Exercice 5.

Tout nœud interne d'un `arbre arithmétique` a exactement deux fils et est étiqueté par l'un des symboles `add`, `sub`, `mul` et `div` ; toute feuille est étiqueté par un nombre entier.

Écrire la fonction `value` qui prend comme argument un arbre arithmétique et qui renvoie la valeur de l'expression arithmétique associée si l'évaluation de celle-ci n'implique aucune division par 0 ; sinon, `value` renvoie `#f`.

`(value '(mul (add (add 3 5) (sub 3 4)) (div 3 2.0)))` $\Rightarrow$ `10.5`

Exercice 6.

On considère des arbres binaires complets (chaque nœud a exactement 0 ou 2 fils) dont les feuilles sont étiquetées par des symboles atomiques. Les nœuds internes ne sont pas étiquetés.

Soit `a` un tel arbre. Simplifier `a` consiste à supprimer dans cet arbre les feuilles redondantes. Ainsi tout nœud ayant deux fils étiquetés par le même symbole est remplacé par ce symbole et ainsi de suite.

Ecrire une fonction `simplify` qui prend pour argument un arbre binaire complet `a` et qui retourne l'arbre binaire complet simplifié correspondant.

Exercice 7.

Le nombre de Strahler d'un arbre binaire complet $S(t)$ est défini comme suit :

$$S(t) = \begin{cases} 1, & \text{si } t \text{ est une feuille.} \\ \max\{S(l), S(r)\}, & \text{si } t \text{ n'est pas une feuille et } S(l) \neq S(r). \\ S(l) + 1, & \text{si } t \text{ n'est pas une feuille et que } S(l) = S(r). \end{cases}$$

où l et r sont respectivement les fils gauche et droit de t.

Ecrire une fonction `strahler` prenant un arbre binaire complet `t` dont seules les feuilles sont étiquetées et renvoyant le nombre de Strahler de cette arbre.

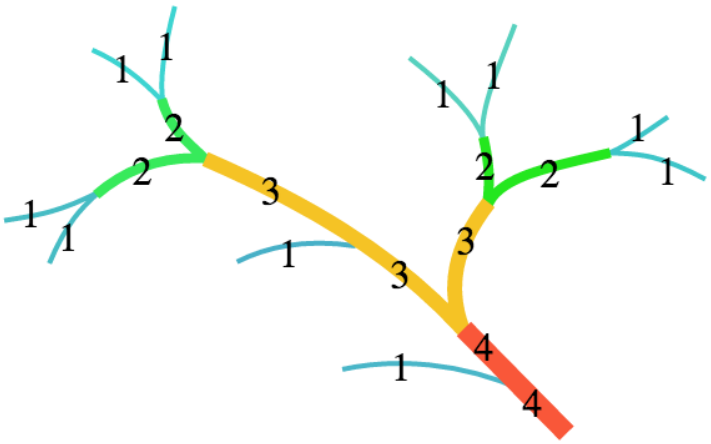


FIGURE 1 – Strahler number (Wikipedia)

Les arbres n-aires

Exercice 8.
Réécrire la fonction `simplify` dans le cas d'arbres n-aires dont seules les feuilles sont étiquetées.

Les structures de données

Exercice 9.
Proposer une implémentation d'un arbre binaire de recherche.