Arbres Généraux

Nous n'utiliserons ici que des arbres généraux étiquetés.

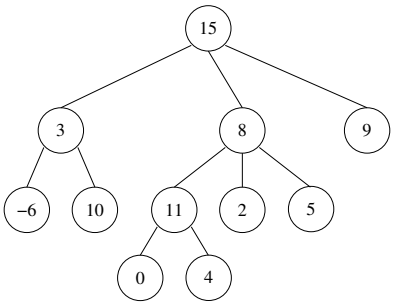

FIGURE 1 – A General Tree

1 Des représentations

Exercise 1.1 (Statique-dynamique : par n-uplets de pointeurs)

1. Décrire la représentation par n-uplets de pointeurs. Quels sont ses défauts ?
2. Donner la représentation par n-uplets de l'arbre de la figure 1.
3. Écrire le type correspondant.

Exercise 1.2 (Premier Fils - Frère Droit)

1. Comment peut-on représenter un arbre général sous forme d'arbre binaire ?
2. Donner la représentation *premier fils - frère droit* l'arbre de la figure 1.
3. Écrire le type correspondant.

---

2 Des parcours

Exercise 2.3 (Parcours en profondeur)

1. Quel est le principe du parcours en profondeur d'un arbre général ?
2. Donner les listes des éléments rencontrés dans l'ordre préfixe et suffixe dans l'arbre de la figure figure 1. Quels autres traitements peut-on faire ?
3. Écrire une fonction *modèle* du parcours en profondeur pour la représentation par n-uplets (insérer les traitements).
4. Modifier le parcours de la question précédente pour écrire une fonction qui recherche si un élément est présent dans l'arbre.
5. Écrire une fonction *modèle* du parcours en profondeur pour la représentation premier fils-frère droit (insérer les traitements).

Exercise 2.4 (Parcours en largeur)

1. Quel est le principe du parcours en largeur d'un arbre général ?
2. Comment repérer les changements de niveaux lors du parcours en largeur ?
3. Écrire un algorithme qui affiche toutes les clefs d'un arbre général en représentation "n-uplets" niveau par niveau (chaque niveau sur sa propre ligne.)
4. Écrire le même algorithme pour la représentation premier fils-frère droit.

---

3 Applications

Exercise 3.5 (Hauteur d'un arbre général – *cont. 01 nov. 2001*)

1. Donner la définition de la hauteur d'un arbre général.
2. Écrire une fonction qui calcule la hauteur d'un arbre général représenté par n-uplets de pointeurs.
3. Écrire une fonction qui calcule la hauteur, avec cette fois ci la représentation *premier fils-frère droit*.

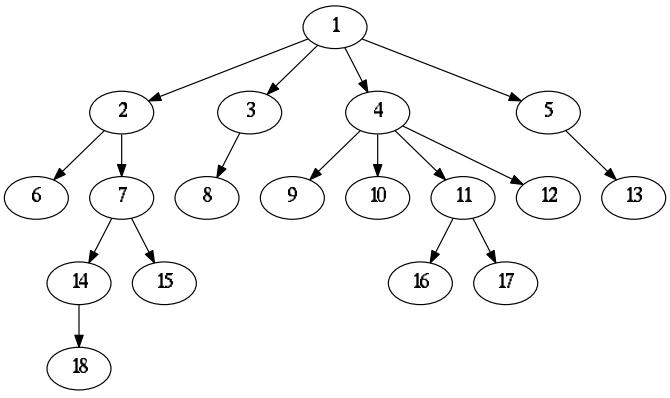

FIGURE 2 – A General Tree

Exercise 3.6 (Arité moyenne d'un arbre général – *cont. 01 nov. 2012*)

On va s'intéresser à l'arité (nombre de fils d'un nœud) moyenne dans un arbre général. On définit l'arité moyenne comme la somme des nombres de fils par nœud divisée par le nombre de nœuds *internes* (nœuds qui ne sont pas des feuilles).

Par exemple, dans l'arbre de la figure 2, il y a 8 nœuds internes (non feuilles), et la somme des nombres de fils par nœud est 17 (compter les flèches pour vérifier), l'arité moyenne est donc de $17/8 = 2.125$

Les deux fonctions demandées seront appelées par la fonction d'appel suivante (avec le bon type).

```
algorithme fonction arity_tuple : real
  parametres locaux
    t_tree_tuples     T
  variables
    entier            nodes, children
    real              r
debut
  children ← 0
  nodes ← 0
  rec_arity_tuple(T, nodes, children)
  r ← children
  retourne (r / nodes)
fin algorithme fonction arity_tuple
```

1. Écrire la procédure `rec_arity_tuple(T,nodes,children)` qui accumule dans le paramètre global `nodes` le nombre de nœuds internes de l'arbre général `T` (en représentation n-uplet de pointeurs) et dans `children` la somme des nombres de fils par nœuds de `T`.

2. Écrire la procédure `rec_arity_dyn(T,nodes,children)` qui accumule dans le paramètre global `nodes` le nombre de nœuds internes de l'arbre général `T` (en représentation *premier-fils-frère-droit*) et dans `children` la somme des nombres de fils par nœuds de `T`.

Exercise 3.7 (Dynamique ↔ Statique)

1. Écrire un algorithme qui transforme un arbre général depuis la forme dynamique (premier fils - frère droit, un arbre binaire, donc) vers la forme statique ("*n*-uplets" de pointeurs.)

2. Écrire la transformation dans l'autre sens.

Exercise 3.8 (Représentation par listes)

Soit un arbre général A défini par A=$< o, A_1, A_2, ..., A_N >$. Nous appellerons *liste* la représentation linéaire suivante de A : (o A_1 A_2 ... A_N). *Cette liste sera représentée par une chaîne de caractères.*

1. Donner la représentation linéaire de l'arbre de la figure 1.

2. Soit la *liste* `(12(2(25)(6)(-7))(0(18(1)(8))(9))(4(3)(11)))`, dessiner l'arbre général correspondant.

3. Écrire l'algorithme qui construit à partir d'un arbre sa représentation linéaire (sous forme de chaîne de caractères), dans le cas de la représentation *n*-uplets.

4. Écrire le même algorithme dans le cas de la représentation premier fils-frère droit.

5. Écrire l'algorithme réciproque (qui construit l'arbre à partir de la liste) pour la représentation premier fils-frère droit.
