Structures de Données Simples 2018 (CSV'18)

TD 4 : Arbres

Pour chaque exercice programmer une ou plusieurs fonctions et plusieurs tests unitaires.

Exercice 1 Utiliser des structures de données natives de Python pour représenter des familles des arbres suivantes :

- arbre binaire ordonnée noeuds-étiquettes.
- arbre d'arité non-bornée non-ordonné arêtes-étiquetées (l'ensemble d'étiquettes des arêtes sortantes d'un noeud n'ayant pas de répétitions)
- arbre d'arité non-bornée ordonnée feuilles-étiquetés

Proposer un interface pour gérer tous ces types d'arbres d'une manière uniforme.

Exercice 2 Pour tout famille d'arbres proposer les constructeurs ainsi que les fonctions suivantes :

- `depth(t)` qui calcule la profondeur de l'arbre `t`,
- `size(t)` qui calcule le nombre de noeuds de l'arbre `t`,
- `find(t,l)` qui vérifie s'il l'arbre `t` contient l'étiquette `l`
- `labels(t)` qui calcule l'ensemble de tous les étiquettes de l'arbre `t`

Exercice 3 Implémenter des fonctions de parcours des arbres suivantes :

- parcours en largeur (BFS) d'un arbre d'arité non-bornée non-ordonné arêtes-étiquetées
- parcours en profondeur (DFS) d'un arbre binaire ordonnée noeuds-étiquettes en 3 variantes :
  - ▷ prefixe,
  - ▷ postfixe (notation polonaise inversée) et
  - ▷ infixe.

Implémenter le parcours d'un arbre binaire ordonné noeud-étiquets avec l'interface (protocole) *Iterator* en utilisant *Generator* i.e., l'instruction `yield`.

Exercice 4 Implémenter les arbres de recherche en utilisant les arbres binaire ordonnées noeuds-étiquettes. Plus spécifiquement, implémenter les fonctions suivantes

- `empty()` qui retourne l'arbre vide
- `insert(t,x)` qui insère un élément `x` à l'arbre `t`
- `find(t,x)` qui vérifie si l'arbre `t` contient l'élément `x`

Exercice 5 Pour les arbres de l'exercice précédent implémenter deux fonctions `rotate_left(t)` et `rotate_right(t)` qui font la rotation à gauche et à droite respectivement. Ensuite utiliser ces fonction pour assurer que l'insertion produit un arbre AVL c.à.d., un arbre bien équilibré. Dans un arbre AVL, les hauteurs des deux sous-arbres d'un même noeud diffèrent au plus de un.

Exercice 6 Implementer les arbres de préfixe avec les arbres d'arité non-bornée non-ordonné arêtes-étiquetées. Plus spécifiquement, implémenter une fonction `create_prefix_tree(words)` qui construit un arbre de préfixe pour un ensemble de mots `words` et implémenter une fonction `expansions(t,word)` qui produit une liste de complementations possibles du mot `word` selon l'arbre `t`.