TD5 : structures de données arborescentes

Exercice 1 : Des arbres

Pour chacun des arbres donnés ci-dessous sous forme de leur écriture en triplets, dessinez-le, donnez sa taille et sa hauteur, le nombre de feuilles, le nombre de nœuds à chaque profondeur.

1. $(1, \Delta, \Delta)$
2. $(3, (1, \Delta, (4, (1, \Delta, (5, \Delta, \Delta)), \Delta)), \Delta)$
3. $(3, (1, (1, \Delta, \Delta), \Delta), (4, (5, \Delta, \Delta), (9, \Delta, \Delta)))$
4. $(3, (1, (1, \Delta, \Delta), (5, \Delta, \Delta)), (4, (9, \Delta, \Delta), (2, \Delta, \Delta)))$

Exercice 2 : Nombre d'arbres binaires

On appelle squelette ou forme d'arbres binaires tout arbre binaire dans lequel on ne tient pas compte des étiquettes (ou dont on a effacé les étiquettes).

Q 2.1 Pour n compris entre 0 et 4, dessinez tous les squelettes d'arbres binaires de taille n, et donnez-en le nombre.

Q 2.2 Si on désigne par c_n le nombre de squelettes d'arbres binaires de taille n, expliquez la relation de récurrence vue en TP permettant d'exprimer le nombre c_n en fonction des nombres c_k avec $k < n$.

$$c_0 = 1$$
$$c_n = \sum_{k=0}^{n-1} c_k \cdot c_{n-k-1}$$

Exercice 3 : Noeuds d'une profondeur donnée

Réalisez une fonction qui à partir d'un arbre binaire construit la liste de tous ses nœuds situés à une profondeur donnée en paramètre.

Exercice 4 : Égalité d'arbres

Deux arbres a_1 et a_2 sont égaux s'ils ont même forme ou squelette, et chacun des nœuds de ce squelette est étiqueté par la même valeur dans les deux arbres.

Réalisez une fonction pour tester l'égalité de deux arbres binaires.

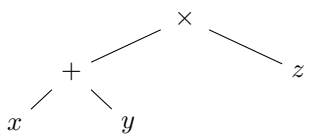
Exercice 5 : Arbre équilibré

Réalisez une fonction pour tester qu'un arbre binaire est équilibré, c'est-à-dire soit un arbre vide, soit un arbre dont les deux sous-arbres sont équilibrés et ont une taille qui ne diffère que d'au plus un.

Exercice 6 : Expressions arithmétiques

Les expressions arithmétiques opérant avec des opérateurs binaires (addition, soustraction, multiplication et division) peuvent toutes être représentées par des arbres binaires dont les nœuds internes sont étiquetés par des opérateurs et les nœuds externes (les feuilles) par des variables.

Par exemple l'expression $(x + y) \times z$ peut être représentée par l'arbre



Q 6.1 Quel type de parcours de l'arbre permet de lire l'expression dans le bon ordre ?

Q 6.2 Réalisez une procédure qui imprime l'expression arithmétique (sous forme arborescente) passée en paramètre dans son écriture infixée complètement parenthésée. L'expression arithmétique donnée en exemple plus haut sera imprimée `((x + y) * z)`.

Q 6.3 Réalisez une fonction qui évalue une expression arithmétique (sous forme arborescente) passée en paramètre accompagnée d'un dictionnaire qui associe à chaque littéral une valeur (par exemple pour `{x: 5, y:3, z:4}`, le résultat sera 32). On peut supposer qu'il existe déjà une fonction `AppliqueOperateur` qui prend trois paramètres : le premier un opérateur (`+`, `-`, `*` ou `/`) et les deux suivants sont les deux opérandes de l'opération, la fonction renvoie le résultat de l'opération.

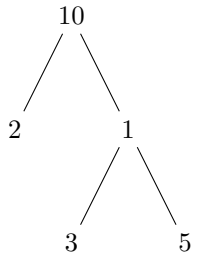
Exercice 7 : Parcours d'arbre

Représenter les états successifs de la pile utilisée lors d'un affichage préfixe selon la procédure suivante :

Procédure *AffichagePrefixe(a)*
- p : une pile d'arbres;
- empiler a dans p;
- **tant que** p *n'est pas vide* **faire**
  - `// on traite l'arbre au sommet de la pile`
  - $s \leftarrow$ sommet de p;
  - dépiler p;
  - **si** s *n'est pas vide* **alors**
    - afficher la valeur de la racine de s;
    - empiler le fils droit de s dans p;
    - empiler le fils gauche de s dans p;
  - **fin**
- **fin**

fin

pour l'arbre suivant :



Exercice 8 : Contruction d'un tas max

Q 8.1 Construire le tas max associé aux valeurs : 5,13,2,25,7,17,20,8,4.

Q 8.2 Identifier un pire des cas pour la construction d'un tas. Quel est le nombre d'échanges réalisé ?

Q 8.3 Identifier un meilleur des cas pour la construction d'un tas. Quel est le nombre d'échanges réalisé ?

Exercice 9 : Maximum d'un ensemble qui évolue

Un utilisateur souhaite disposer d'une structure de donnée lui permettant :
- d'ajouter **au fur et à mesure** de nouveaux entiers à cet ensemble,
- d'être capable d'en extraire le **maximum** (après quoi, la valeur est retirée de la structure).

Un exemple d'utilisation pourrait être le suivant (`s` est la structure de données) :

```
> ajouter(s,2)
> ajouter(s,5)
> ajouter(s,3)
> maximum(s)
5
> ajouter(s,7)
> maximum(s)
7
> maximum(s)
3
```

Pour réaliser cela, on utilisera un tas max implanté dans une classe, composée de deux attributs : `tas` qui représente le tas sous forme d'un tableau, et `last` la position du dernier élément du tas dans le tableau `tas` (ou -1) si le tas est vide. On supposera qu'on n'ajoutera pas deux fois le même entier.

On dispose d'une constructeur d'un tas vide :

```
class Tas:
  def __init__():
    self.tas = []
    self.last = -1
```

Dans cet exercice on s'intéresse au décompte du nombre de comparaisons et d'affectations

Q 9.1 Donner le contenu de la structure de données (c'est-à-dire les valeurs des attributs) après chacune des instructions de l'exemple ci-dessus.

Q 9.2 Ecrire l'algorithme de la fonction `maximum` (ou le code Python). On supposera le tas non vide.

Q 9.3 Donner le comportement asymptotique en temps de la fonction `maximum`. Justifier.

Q 9.4 Ecrire l'algorithme ou le code Python de la procédure `ajouter`.

Q 9.5 Donner le comportement asymptotique en temps de la procédure `ajouter` (on suppose que la structure contient déjà n valeurs). Justifier.

Exercice 10 : Construction d'AVL

Construire l'arbre AVL pour la série de valeurs suivante : 5,13,2,25,7,17,20,8,4.

Exercice 11 : Expression des rotations et représentation compacte des nœuds

Nous avons vu dans le cours un algorithme pour exécuter des rotations gauche et droite quand les nœuds stockaient la hauteur de leur sous-arbre. On suppose maintenant qu'au lieu de la hauteur du sous-arbre, c'est la valeur du déséquilibre du nœud qui sera stockée.

Q 11.1 Ecrire l'algorithme de l'opération de rotation droite avec ce nouveau type pour les nœuds (la mise à jour des valeurs de déséquilibre peut-être obtenue grâce aux formules de l'exercice suivant).

Q 11.2 Combien de bits d sont nécessaires pour stocker la valeur du déséquilibre ?

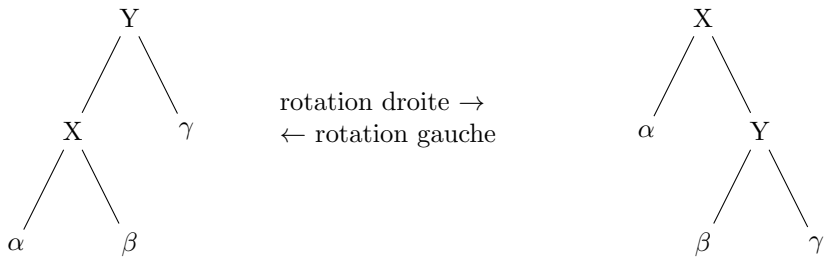
On suppose maintenant que les entiers sont stockés sur w bits et que les valeurs associées aux nœuds seront des entiers tels qu'il faut au plus $w - d$ bits pour être stockés.

Q 11.3 Proposer un stockage intelligent de la valeur du nœud et du déséquilibre.

Q 11.4 Ecrire le code de la fonction `valeur` qui étant donné un nœud retourne la valeur associée.

Q 11.5 Ecrire le code de la fonction `desequilibre` qui étant donné un nœud retourne le déséquilibre associé.

Exercice 12 : Calcul de l'équilibre des nœuds dans les AVL



On note $d(n)$ le déséquilibre du nœud n et $d'(n)$ le déséquilibre du nœud n après une rotation. Démontrer les propriétés suivantes :

- Après une rotation droite autour du sommet Y, on a :
  - $d'(X) = d(X) - 1 + \min(d'(Y), 0)$
  - $d'(Y) = d(Y) - 1 - \max(d(X), 0)$
- Après une rotation gauche autour du sommet X, on a :
  - $d'(X) = d(X) + 1 - \min(d(Y), 0)$
  - $d'(Y) = d(Y) + 1 + \max(d'(X), 0)$