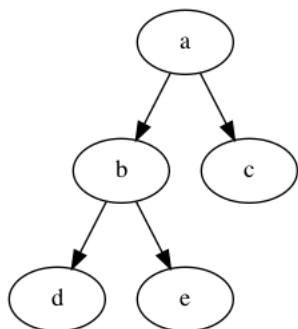


Exercice 1 : Arbres binaires (60min)

Un arbre binaire comporte des noeuds ayant au maximum deux enfants. On dira qu'il est *complet* si tous les noeuds de tous les niveaux ont deux enfants et qu'il est *équilibré* si l'arbre est complet sauf pour le dernier niveau. Ces arbres (binaires ou non) peuvent être stockés avec une structure de donnée *explicite* en POO¹ (comme ci-dessous à droite):



Une structure POO possible pour un arbre binaire :

```

Class Noeud:
    def __init__(self, v, d = None, g = None):
        self.v = v
        self.g = g
        self.d = d

```

Exercice 1.1 (20min) – Utilisez la structure de donnée POO afin d'implémenter l'arbre donné en exemple ci-dessus (à gauche). Quel type de parcours renvoie les lettres dans l'ordre alphabétique ? Implémentez le.

Exercice 1.2 (10min) - Écrire une fonction de calcul de la hauteur d'un arbre.

Exercice 1.3 (15min) – Automatisez la création de l'arbre pour les n premières lettres de l'alphabet (depuis a). Testez les méthodes des questions précédentes dessus (parcours et visualisation). Conseil : générez d'abord une liste de n lettres et utilisez là comme structure *implicite* d'arbre pour le créer en POO.

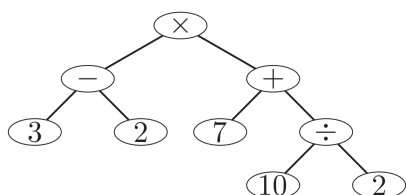
Exercice 1.4 (15min) - Nous souhaitons désormais valider si l'arbre est bien complet. Conseil : proposez une méthode basée sur le nombre théorique de noeuds contenus dans un arbre complet en fonction de la hauteur.

Exercice 1.5 (Bonus) - Valider si l'arbre est bien équilibré. Une approche possible : comparez la différence de hauteur entre les feuilles.

Exercice 1.5 (Bonus) – Visualisez l'arbre en utilisant la bibliothèque Graphviz (décrite en annexe) en réalisant un parcours en largeur .

Exercice 2 : Arbres syntaxiques (60min)

Un arbre *syntactique* permet le stockage d'une expression structurée, par exemple une équation. Dans cet exercice nous allons utiliser un tel arbre syntaxique (exemple ci-dessous à gauche) afin de réaliser un calcul arithmétique simple à partir de l'expression fournie de manière textuelle (ci-dessous à droite). Le but sera de calculer la valeur de cette expression :



Expression : $(3-2) * (7+(10/2))$

Résultat : 12.0

Nous ferons l'hypothèse que les opérations sont limitées à $+$ $-$ $/$ et $*$, seront toujours binaires et porteront sur des valeurs numériques entières (mais le résultat peut ne pas être un entier).

Exercice 2.1 (10min) - Proposez une structure de données d'arbre permettant de stocker l'arbre syntaxique ci-dessus (à gauche). Appliquez la structure sur l'expression ci-dessus (à droite).

Exercice 2.2 (20min) - Implémentez deux méthodes afin de 1) valider et 2) d'évaluer l'expression stockée

¹Programmation Orientée Objet

dans l'arbre. Vous pouvez vous inspirer de la version récursive vue lors du cours 2 (page 21²).

Exercice 2.3 (30min) - Écrivez une méthode permettant de construire l'arbre à partir d'une expression sous forme de chaîne de caractère en entrée " $((3 - 2) * (7 + (10 / 2)))$ ". Les espaces permettent de faire un `.split(" ")`. Suggestion : parcourez caractère par caractère l'expression textuelle, et utilisez une Pile permettant la bonne construction de l'arbre au fur et à mesure de son parcours.

Exercice 2.4 (Bonus) - Proposez une optimisation afin de ne pas effectuer des calculs déjà réalisés.

Annexe : Visualiser un arbre avec GraphViz

IMPORTANT : il vous est nécessaire d'installer le module Graphviz dans Conda comme suit (dans le prompt Spyder):

Voici un exemple de code permettant la création des noeuds et arrêtes d'un arbre :

```
conda install python-graphviz
```

```
import graphviz as gv
```

```
# cr ation d'un arbre
```

```
dot = gv.Digraph(format="png")
```

```
# ajout des noeuds
```

```
dot.node('1')
```

```
dot.node('3')
```

```
dot.node('2')
```

```
dot.node('5')
```

```
# ajout des arr tes
```

```
dot.edge('1', '3')
```

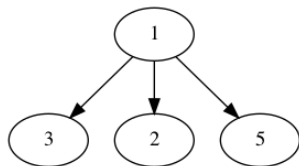
```
dot.edge('1', '2')
```

```
dot.edge('1', '5')
```

```
# visualiser
```

```
dot.view()
```

Voici l'arbre résultat :



²<https://gitlab.ec-lyon.fr/rvuillemin/inf-tc1/-/blob/master/cours/INF-TC1-cours-chap2.pdf>