Algorithmique
Partiel n° 2 (P2)

Info-sup S2#
Epita

18 janvier 2021 - 8h30-10h30

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Le code :**
  - Tout code doit être écrit dans le langage `Python` (pas de C, Caml, Algo ou autre).
  - **Tout code `Python` non indenté ne sera pas corrigé.**
  - Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué en **annexe** !
- ☐ Durée : 2h00

Exercice 1 (AVL. : Construction – *3 points*)

À partir d'un arbre vide, construire l'AVL en insérant successivement les valeurs

20, 62, 37, 3, 2, 6, 74, 73, 14

Vous dessinerez l'arbre à deux étapes :
– après l'insertion de 2 ;
– l'arbre final.
Indiquez quelles rotations ont été effectuées, dans l'ordre (par exemple si une rotation gauche a été effectuée sur l'arbre de racine 42, indiquez *rg(42)*).

Exercice 2 (Arbre 2-3-4 → Arbre bicolore – *2 points*)

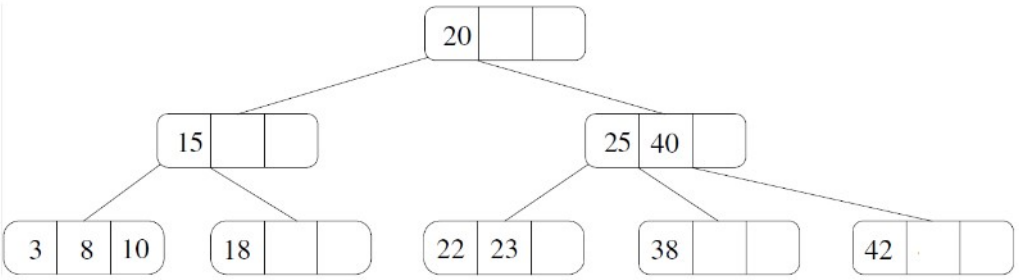
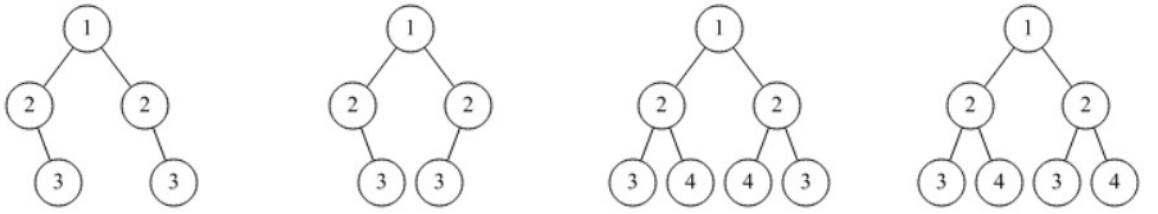


FIGURE 1 – Arbre 2-3-4 à transformer

1. Dessiner l'arbre bicolore correspondant à l'arbre 2-3-4 de la figure 1. Les 3-nœuds devront être représentés penchés à gauche.
2. L'arbre obtenu est-il un AVL ? Justifiez (avec précision et concision) votre réponse.

Exercice 3 (Symmetric – *4 points*)

Un arbre binaire est dit *symétrique* s'il est identique à son miroir (arbre renversé selon un axe vertical).



Arbre non symétrique *Arbre symétrique* *Arbre symétrique* *Arbre non symétrique*

Écrire la fonction `symmetric(`B`)` qui vérifie si l'arbre binaire B est symétrique.

Exercice 4 (ABR : Insertion en feuille – *4 points*)

Écrire la fonction récursive `leaf_insert(`B`, `x`)` qui ajoute en feuille dans l'arbre binaire de recherche B la valeur x sauf si celle-ci est déjà présente. On veut également savoir si l'insertion a eu lieu. La fonction retourne donc un couple : avec comme premier élément l'arbre après éventuelle insertion, et comme deuxième élément un booléen indiquant si l'insertion a eu lieu.

Exercice 5 (Average Balance Factor – *5 points*)

Écrire la fonction `average_balances(B)` qui calcule la moyenne des déséquilibres de tous les nœuds de l'arbre binaire (class `BinTree`) B. Si l'arbre est vide la fonction retourne 0.

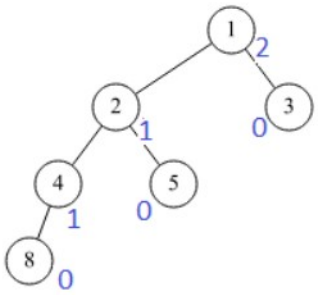


FIGURE 2 – Arbre `B4`
somme déséquilibres = 4
taille = 6

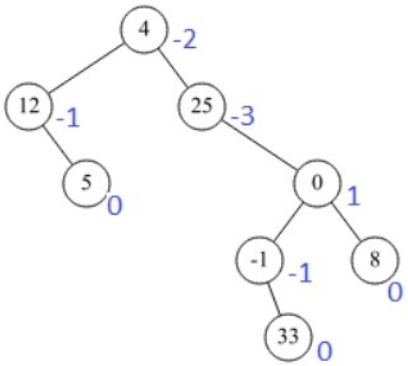


FIGURE 3 – Arbre `B5`
somme déséquilibres = -6
taille = 8

```
>>> average_balances(B4)
0.6666666666666666
>>> average_balances(B5)
-0.75
```

Exercice 6 (What is this ? – *2 points*)

Soit la fonction suivante :

```
def New(B, x = None):
    """
    B != None
    """
    if B.left == None:
        if x != None:
            B.left = BinTree(B.key, None, None)
    else:
        B.left = New(B.left, B.key)
    if B.right == None:
        if x != None:
            B.right = BinTree(x, None, None)
    else:
        B.right = New(B.right, x)
    return B
```

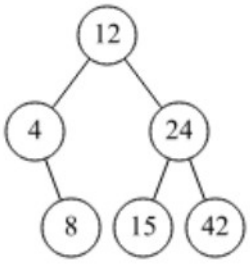


FIGURE 4 – Arbre B

Dessiner l'arbre retourné par `New(B)` avec B l'arbre ci-dessus.

Annexes

Les arbres binaires

On suppose la class `BinTree` importée.

```
class BinTree:
    def __init__(self, key, left, right):
        self.key = key
        self.left = left
        self.right = right
```

Fonctions et méthodes autorisées

- `abs`
- `min`, `max`

Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas vous devez donner leurs spécifications : on doit savoir ce qu'elles font.

Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.