Algorithmique
Correction Partiel n° 2 (P2)

INFO-SUP S2# – EPITA

8 janvier 2018 - 11 : 00

Solution 1 **(Combien ? –** ***3,5 points*****)**

Spécifications :

La fonction `nb_inter(B, a, b)` calcule le nombre de valeurs de l'arbre binaire de recherche B dans l'intervalle $[a, b[$.

```
    def nb_inter(B, a, b):
        if B == None:
            return 0
        elif B.key < a:
            return nb_inter(B.right, a, b)
        elif B.key > b:
            return nb_inter(B.left, a, b)
        else:
            return 1 + nb_inter(B.left, a, b) + nb_inter(B.right, a, b)
```

Solution 2 **(ABR → AVL –** ***4,5 points*****)**

Spécifications :

La fonction `makeAVL(`B`)` contruit une copie de l'arbre binaire B avec les déséquilibres renseignés en chaque nœud.

```
        def __makeAVL(B):
            if B == None:
                return(None, -1)

            else:
                A = AVL(B.key)

                (A.left, hl) = __makeAVL(B.left)
                (A.right, hr) = __makeAVL(B.right)

                A.bal = hl - hr
                return (A, 1 + max(hl, hr))


        def makeAVL(B):
            (A, h) = __makeAVL(B)
            return A
```

***Solution 3* (AVL - Ajout 0 – *5 points*)**

Spécifications :

La fonction `add0(A)` ajoute la valeur 0 dans l'AVL A (ne contenant que des entiers naturels non nuls) et retourne un couple : l'arbre résultat et un booléen indiquant si la hauteur de A a changé.

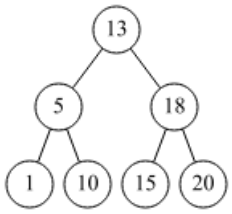
```python
def __add0_AVL(A):
    if A == None:
        return (avl.AVL(0, None, None, 0), True)
    else:
        (A.left, dh) = __add0_AVL(A.left)
        if not dh:
            return (A, False)
        else:
            A.bal += 1
            if A.bal == 0:
                return (A, False)
            elif A.bal == 1:
                return (A, True)
            else: # A.bal == 2
                A = rr(A)
                return (A, False)

def add0_AVL(A):
    (A, _) = __add0_AVL(A)
    return A
```

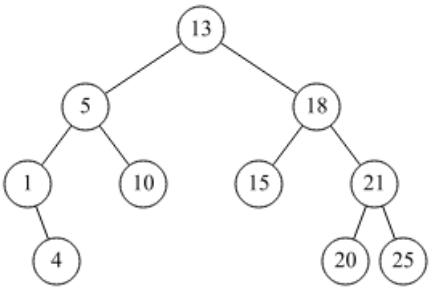
***Solution 4* (AVL – *3 points*)**

AVL résultat depuis la liste $[13, 20, 5, 1, 15, 10, 18, 25, 4, 21, 7, 12, 23]$.

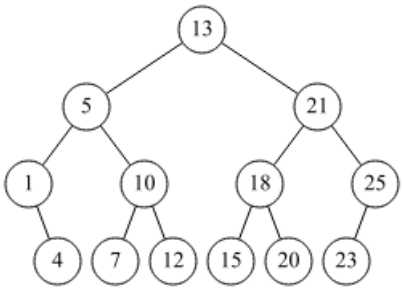
13, 20, 5, 1, 15, 10, 18 :



ajouts de 25, 4 et 21 :



ajouts de 7, 12 et 23 :



Solution 5 **(Arbre 2.3.4 → Arbre bicolore –** ***2 points*****)**

1. Transformation de l'arbre 2.3.4 du sujet :

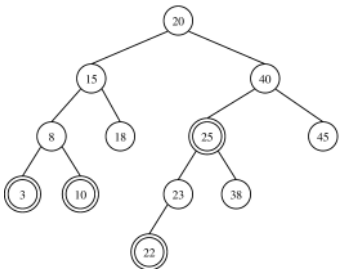
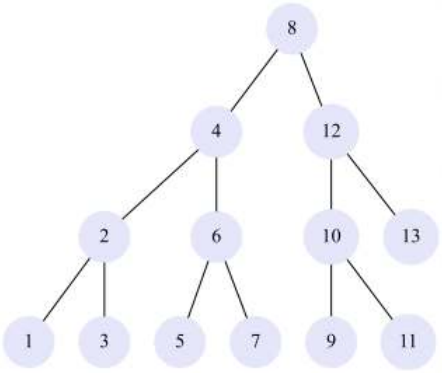


FIGURE 1 – Arbre bicolore

2. L'arbre obtenu n'est pas un AVL.
   Le nœud contenant 40 a un déséquilibre de 2 (l'arbre n'est pas h-équilibré).

Solution 6 **(Arbres et mystère –** ***3 points*****)**

1. *Arbre construit par* `makeTree(13)` *:*



2. *Propriétés de l'arbre construit par* `makeTree(n)` *(*$n > 0$*)* :

   (a) Arbre parfait

   (b) Arbre binaire de recherche
