Algorithmique
Correction Partiel n° 2 (P2)

Info-sup S2# – Epita

7 janvier 2020 - 13h-15h

Solution 1 **(Arbres de Léonard –** ***3 points*****)**

1. L'arbre A_5 de Fibonacci est celui de la figure 1 dont les noeuds contiennent leur propre valeur de déséquilibre

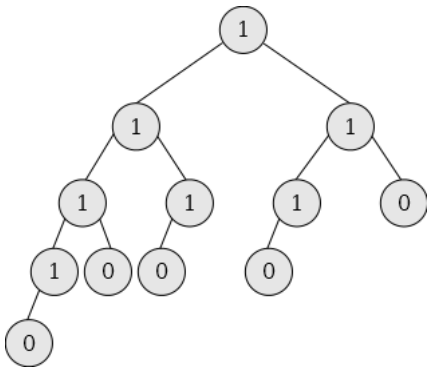


Figure 1 – A_5 de Fibonacci

2. (a) $h_n = n - 1$
   (b) A_0 est réduit à une feuille, donc un arbre h-équilibré.
   La racine de A_1 a pour déséquilibre 1 (une feuille à gauche, rien à droite).
   Pour $n \geq 2$, A_n est un arbre de hauteur $n-1$. Ses 2 sous-arbres sont A_{n-1} de hauteur $n-2$ et A_{n-2} de hauteur $n-3$. Le déséquilibre de la racine de A_n est donc 1 ($n-2-(n-3)$).
   Bref, tous les noeuds internes d'un arbre de Fibonacci ont un déséquilibre de 1 : c'est donc un arbre h-équilibré.

***Solution 2* (Arbres de Léonard, encore – *4 points*)**

Spécifications :
La fonction `leonard_tree(`n`)` construit l'arbre de Fibonacci A_n.

```
def leonard_tree(n):
    if n == 0:
        return None
    elif n == 1:
        return BinTree(1, None, None)
    else:
        G = leonard_tree(n-1)
        D = leonard_tree(n-2)
        key = G.key
        if D != None:
            key += D.key

        return BinTree(key, G, D)
```

***Solution 3* (Suppression)**

1. **Spécifications :**
   La fonction `maxBST(`B`)` retourne la valeur maximale de l'arbre binaire de recherche non vide B.

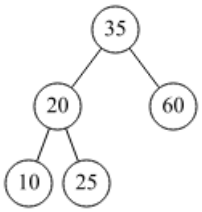
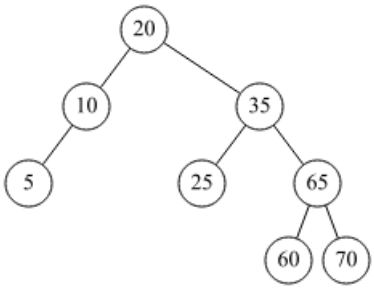
```
def maxBST(B):
    while B.right != None:
        B = B.right
    return B.key
```

2. **Spécifications :**
   La fonction `delBST(`B`, `x`)` supprime l'élément x de l'arbre binaire de recherche B et renvoie l'arbre résultat.

```
def delBST(B, x):
    if B == None:
        return None
    else:
        if x == B.key:
            if B.left == None:
                return B.right
            elif B.right == None:
                return B.left
            else:
                B.key = maxBST(B.left)
                B.left = del_bst(B.left, B.key)
                return B
        else:
            if x < B.key:
                B.left = delBST(B.left, x)
            else:
                B.right = delBST(B.right, x)
            return B
```

***Solution 4* (AVL – *4 points*)**

AVL résultat depuis la liste $[25, 60, 35, 10, 20, 5, 70, 65]$.

Arbre créé par insertions de 25, 60, 35, 10, 20 :	*Rotations :*
	`rlr(25)` `rdg(25)` `lrr(25)` `rgd(25)`
Arbre après ajout de 5, 70, 65 :	*Rotations :*
	`rr(35)` `rd(35)` `rlr(60)` `rdg(60)`

***Solution 5* (What is this ? – *3 points*)**

1. *Résultats pour*
   (a) `test(`B_2`)` : True
   (b) `test(`B_3`)` : False

2. `test(`B`)` vérifie si l'arbre binaire B est h-équilibré.

3. Pour optimiser cette fonction : si le booléen du premier appel est faux, il est possible d'éviter le deuxième en retournant directement `(?, False)`.