Algorithmique
Correction Contrôle n° 3 (C3)

INFO-SPÉ - S3 – EPITA

9 novembre 2021 - 9 : 30

Solution 1 **(Graphes et composantes... –** ***5 points*****)**

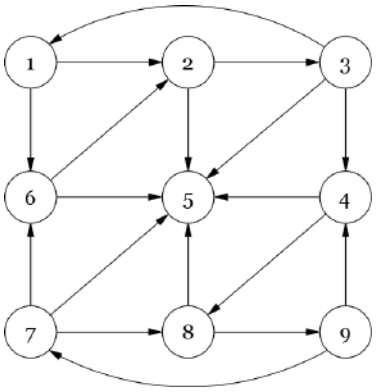

FIGURE 1 – Graphe Orienté G

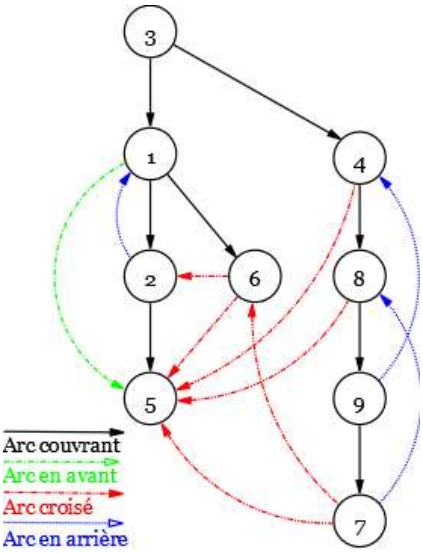

FIGURE 2 – Forêt couvrante associée au parcours profondeur de G (base 3 et ordre croissant des sommets)

1. Le tableau des demi-degrés intérieurs des sommets de G est le suivant :

	1	2	3	4	5	6	7	8	9
demi-degrés intérieurs	1	2	1	2	6	2	1	2	1

2. Les sommets du graphe G dans l'ordre de rencontre *préfixe* en partant du sommet 3 sont :
   `3, 1, 2, 5, 6, 4, 8, 9, 7`

3. Non, le graphe G n'est pas fortement connexe.

4. Le graphe possède 2 composantes fortement connexes.

5. Il n'y a pas de sommets de degré égal à 0.

Solution 2 **(Famille nombreuse – *4 points*)**

Spécifications :

La fonction `morechildren(`T`)` vérifie si chaque nœud interne de l'arbre T (`TreeAsBin`) a strictement plus de fils que son père.

```
    def morechildren(B, nbc=0): # nbc = child number of B's parent
        k = 0
        C = B.child
        while C:
            k += 1
            C = C.sibling
        if B.child and k <= nbc:
            return False
        else:
            C = B.child
            while C and morechildren(C, k):
                C = C.sibling
            return C == None
```

Solution 3 **(Décroissant – *4 points*)**

Spécifications :

`decrease(`B`)` retourne la liste des clés du B-arbre B en ordre décroissant.

```
    def __decrease(B, L):
        if B.children == []:
            for i in range(B.nbkeys-1, -1, -1):
                L.append(B.keys[i])
        else:
            for i in range(B.nbkeys, 0, -1):
                __decrease(B.children[i], L)
                L.append(B.keys[i-1])
            __decrease(B.children[0], L)

    def decrease(B):
        L = []
        if B:
            __decrease(B, L)
        return L
```

Solution 4 **(B-arbre : insertion et supression –** ***3 points*****)**

1. Arbre `B1` après insertion des valeurs 11, 32, 20, avec le principe "à la descente" :

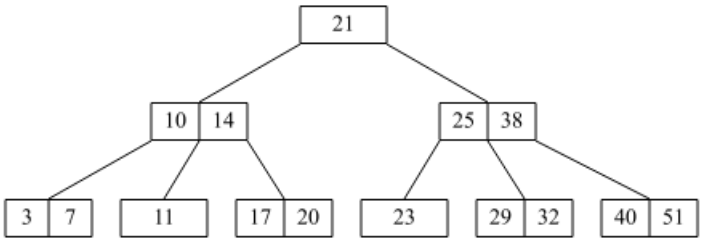

FIGURE 3 – Après insertions

2. Arbre `B2` après suppression de la valeur 15, avec le principe "à la descente" :

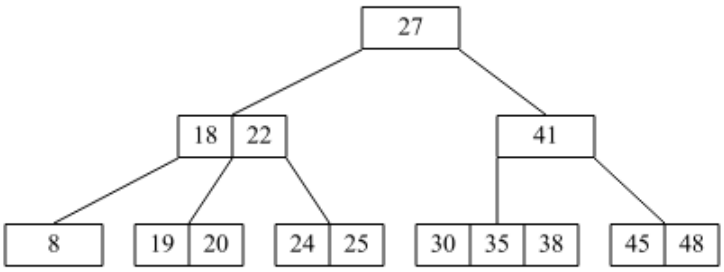

FIGURE 4 – Après suppression

Solution 5 **(What ? –** ***4 points*****)**

1.

	Résultat retourné	*Nombre d'appels*
(a) `mystery(B2, 0, 92)`	`True`	8
(b) `mystery(B3, 0, 20)`	`False`	6
(c) `mystery(B3, 1, 99)`	`False`	8

2. La fonction `mystery(B, a, b)` vérifie si l'arbre B est bien "ordonné" i.e. est un arbre de recherche, avec ses valeurs dans l'intervalle $[a, b]$.