Algorithmique
Correction Contrôle n° 2 (C2)

Info-sup S2 – Epita

mars 2021

Solution 1 **(Arbre Binaire : Ordres –** ***2 points*****)**

1. L'arbre B correspondant à ces deux séquences est le suivant :

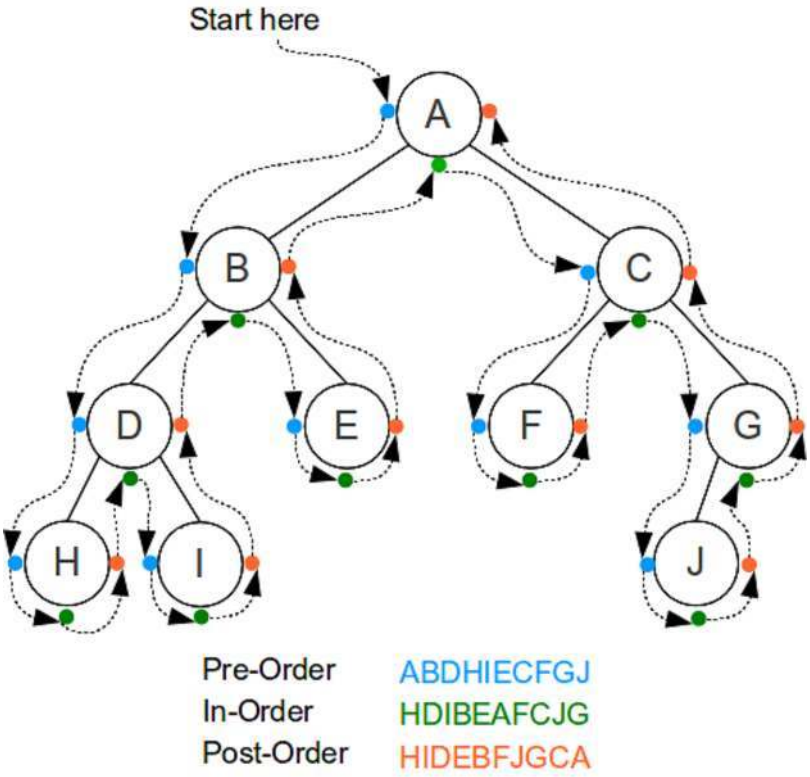

Figure 1 – Arbre binaire B

2. Les valeurs *suffixe* de rencontre de l'arbre B sont : H I D E B F J G C A

Solution 2 **(ABR : insertions – *3 points*)**

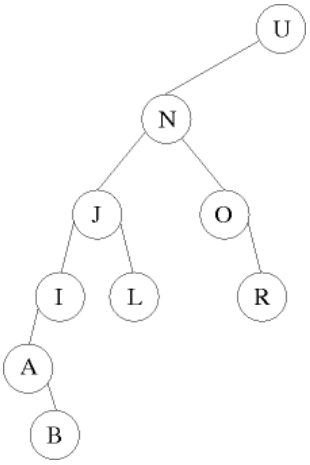

FIGURE 2 – Un joli ABR, construit en feuille

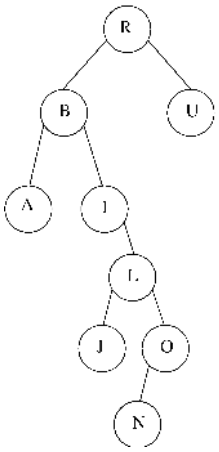

FIGURE 3 – Un joli ABR, construit en racine

Solution 3 **(Matrice triée – *5 points*)**

Spécifications :

La fonction `list_sorted(L, n)` vérifie si la liste non vide `L` de longueur `n` est triée en ordre croissant.

```
    def list_sorted(L, n):
        i = 1
        while i<n and L[i-1]<=L[i]:
            i = i+1
        return i==n

```

Spécifications :

La fonction `matrix_sorted(M)` vérifie si la matrice `M` est *triée* en ordre croissant. La matrice `M` est supposée non vide.

```
    def matrix_sorted(M):
        c = len(M[0])
        if not list_sorted(M[0], c):
            return False
        else :
            l = len(M)
            i = 1
            while i<l and M[i-1][c-1]<M[i][0] and list_sorted(M[i], c):
                i = i+1
            return i==l

```

Solution 4 **(Largeur moyenne pondérée – *6 points*)**

Spécifications :

La fonction `get_average(B)` construit la liste `L`.

— `len(L)` = nombre de niveaux de l'arbre `B`

— `L[i]` = somme des clés des noeuds du niveau `i` divisée par le nombre de noeuds du niveau `i`

```
def get_average(B):
    if not B:
        return []
    else:
        q = queue.Queue()
        q.enqueue(B)
```

```
        q.enqueue(None)
        s = 0
        cpt = 0
        res = []
        while not q.isempty():
            B = q.dequeue()
            if B == None:
                res.append(s/cpt)
                if not q.isempty():
                    q.enqueue(None)
                    s = 0
                    cpt = 0
            else:
                s = s+B.key
                cpt = cpt+1
                if B.left != None:
                    q.enqueue(B.left)
                if B.right != None:
                    q.enqueue(B.right)
        return res
```

Solution 5 **(Recherche du numéro hiérarchique –** ***4 points*****)**

Spécifications :

La fonction `search_hier(B, x)` retourne le numéro en ordre hiérarchique de `x` s'il est présent dans `B`, `None` sinon.

```
def search_hier(B, x, num=1):
    if B == None:
        return None
    else:
        if B.key == x:
            return num
        else:
            v = search_hier(B.left, x, 2*num)
            if v != None:
                return v
            else :
                return search_hier(B.right, x, 2*num+1)
```