

Algorithmique

Correction Contrôle n° 2 (C2)

INFO-SUP S2 – EPITA

4 mars 2019 - 9 : 00

Solution 1 (Un peu de cours... – 4 points)

1. Mesures :

- (a) La taille de l'arbre B est : 11
- (b) La hauteur de l'arbre B est : 4
- (c) La longueur de cheminement de l'arbre B est : 27
- (d) La profondeur moyenne externe de l'arbre B est : $15/4 = 3.75$

2. En utilisant la numérotation hiérarchique, les nœuds de l'arbre B sont :
1, 2, 3, 5, 7, 10, 11, 14, 21, 28, 29

Solution 2 (Maximum Gap – 4 points)

Spécifications :

La fonction `maxgap(M)` retourne le gap maximum des lignes de la matrice non vide M .

```
1 def gaplist(L):
2     """
3     returns the gap of the list L (not empty)
4     """
5     valMin = L[0]
6     valMax = L[0]
7     for i in range(1, len(L)):
8         valMin = min(valMin, L[i])
9         valMax = max(valMax, L[i])
10    return valMax - valMin
11
12 def maxgap(M):
13     mgap = gaplist(M[0])
14     for i in range(1, len(M)):
15         mgap = max(mgap, gaplist(M[i]))
16    return mgap
```

In one function (gaplist inlined) :

```
1     def maxgap2(M):
2         mgap = 0
3         (l, c) = (len(M), len(M[0]))
4         for i in range(l):
5             valMin = M[i][0]
6             valMax = M[i][0]
7             for j in range(1, c):
8                 valMin = min(valMin, M[i][j])
9                 valMax = max(valMax, M[i][j])
10            mgap = max(mgap, valMax - valMin)
11    return mgap
```

Solution 3 (Recherche – 4 points)

Spécifications :

La fonction `searchMatrix(M, x)` retourne la position (i, j) de la première valeur x trouvée dans la matrice M (non vide) ou $(-1, -1)$ si $x \notin M$.

```
1         def searchMatrix(M, x):
2
3             (i, lin, col) = (0, len(M), len(M[0]))
4             found = -1
5
6             while i < lin and found == -1:
7                 j = 0
8                 while j < col and M[i][j] != x:
9                     j += 1
10                if j != col:
11                    found = j
12                i += 1
13
14            if found != -1:
15                return (i-1, found)
16            else:
17                return (-1, -1)
```

Solution 4 (Tests – 4 points)

Spécifications : La fonction `equal(B1, B2)` vérifie si les arbres $B1$ et $B2$ sont indentiques.

```
1         def equal(B1, B2):
2             if B1 == None:
3                 return B2 == None
4             elif B2 == None:
5                 return False
6             elif B1.key == B2.key:
7                 return equal(B1.left, B2.left) and equal(B1.right, B2.right)
8             else:
9                 return False
10
11         #
12
13         def equal2(B1, B2):
14             if B1 == None or B2 == None:
15                 return B1 == B2
16             else:
17                 return (B1.key == B2.key) \
18                     and equal2(B1.left, B2.left) \
19                     and equal2(B1.right, B2.right)
```

Solution 5 (Feuilles – 2 points)

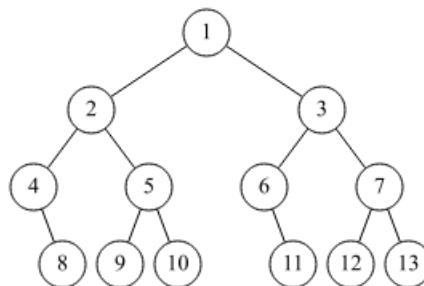
Spécifications :

La fonction `leaves(B)` calcule le nombre de feuilles de l'arbre binaire B .

```
1  def leaves(B):  
2      if B == None:  
3          return 0  
4      else:  
5          if B.left == B.right:    # B.left == None and B.right == None  
6              return 1  
7          else:  
8              return leaves(B.left) + leaves(B.right)
```

Solution 6 (Mystery – 3 points)

1. Arbre binaire résultat de l'application `mystery([4, 8, 2, 9, 5, 10, 1, 6, 11, 3, 12, 7, 13])` :



2. (a) L'arbre est un ABR si la liste est triée en ordre croissant.
(b) L'arbre est complet si la liste est de taille $= 2^h - 1$ avec h un entier naturel.