Algorithmique Correction Contrôle no 2 (C2)

Info-sup (S2) – Epita 22 février 2017 - 9 : 30

Solution 1 (Il faut oser ... - 3 points)

1. Représenter graphiquement l'arbre correspondant.

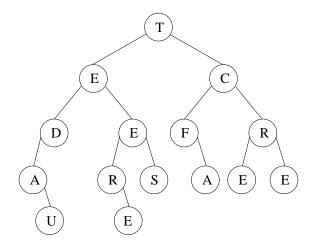


FIGURE 1 – Arbre représentant le tableau hiérarchique

2. En considérant un parcours profondeur main gauche de cet arbre, donner l'ordre infixe de rencontre des noeuds de cet arbre

Le parcours infixe de l'arbre de la figure 1 est :

AUDEREESTFACERE (Audere est facere1) "Oser c'est faire"

3. Donner sous forme d'occurrences $(B = \{\varepsilon, 0, 1, 01, 10, \ldots\})$ la représentation de cet arbre.

 $B = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 010, 011, 101, 110, 111, 0001, 0101\}$

1. Devise du club de foot de Tottenham

Solution 2 (Maximum Gap - 5 points)

Spécifications:

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

```
def gapList(L):
    valMin = L[0]
    valMax = L[0]

for i in range(1, len(L)):
    valMin = min(valMin, L[i])
    valMax = max(valMax, L[i])

return valMax - valMin

def maxGapMatrix(M):
    mgap = gapList(M[0])
    for i in range(1, len(M)):
        mgap = max(mgap, gapList(M[i]))
    return mgap
```

In one function (gapList inlined):

```
def maxGapMatrix2(M):
    mgap = 0
    (1, c) = (len(M), len(M[0]))
    for i in range(1):
        valMin = M[i][0]
        valMax = M[i][0]
        for j in range(1, c):
            valMin = min(valMin, M[i][j])
            valMax = max(valMax, M[i][j])
        mgap = max(mgap, valMax - valMin)
    return mgap
```

Solution 3 (Synergistic Dungeon – 4 points)

Spécifications:

La fonction dungeon(M) retourne le nombre minimum de points de vie que doit avoir la princesse pour sauver le chevalier dans le donjon représenté par la matrice non vide M.

```
def dungeon(M):
                   (1, c) = (len(M), len(M[0]))
                                             \#first\ row
                   for j in range(1, c):
                        M[0][j] += M[0][j-1]
                                             \#first\ column
6
                   for i in range(1, 1):
                        M[i][0] += M[i-1][0]
                                             \# rest of the grid
                   for i in range(1, 1):
                        for j in range(1, c):
11
                            M[i][j] += \min(M[i-1][j], M[i][j-1])
12
13
                   return M[1-1][c-1] + 1
14
```

Solution 4 (Tests - 8 points)

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

```
def equal(B1, B2):
              if B1 == None:
                 return B2 == None
              elif B2 == None:
                 return False
              elif B1.key == B2.key:
                 return equal(B1.left, B2.left) and equal(B1.right, B2.right)
                 return False
11
12
        def equal2(B1, B2):
            if B1 == None or B2 == None:
14
                 return B1 == B2
15
            else:
16
                 return (B1.key == B2.key) \
17
                        and equal2(B1.left, B2.left) \
18
                        and equal2(B1.right, B2.right)
```

2. Spécifications : La fonction isSubTree(S, B) vérifie si l'arbre S est un sous-arbre de l'arbre B.

```
def isSubTree(sB, B):
               if sB == None:
                    return True
               elif B == None:
                    return False
6
               else:
                    if sB.key == B.key:
                        return equal(sB, B)
                    else:
9
                        return isSubTree(sB, B.left) or isSubTree(sB, B.right)
10
12
13
           def search(x, B):
14
               if B == None:
15
                    return None
16
               elif x == B.key:
17
                    return B
18
               else:
19
                    R = search(x, B.left)
20
                    if R == None:
21
                        R = search(x, B.right)
22
                    return R
23
           def isSubTree2(sB, B):
           if sB == None:
26
               return True
27
           else:
28
               return equal(sB, search(sB.key, B))
```