Algorithmics
Correction Midterm #2 (C2)

Undergraduate 1st year S2# – Epita

novembre 2019

Solution 1 **(A little coursework. . . – *4 points*)**

1. This is the tree `B` drawn figure 1.

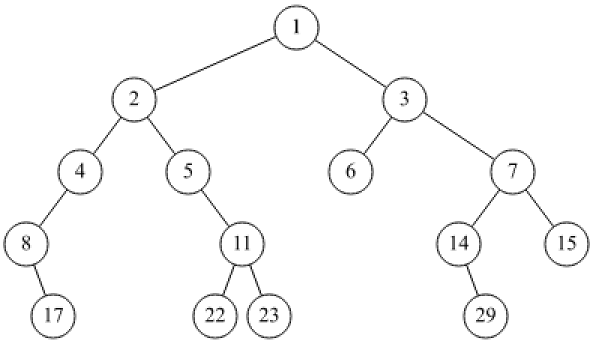

Figure 1: Binary tree

2. The interne path length of the tree `B` is: $17 = 0 + 1 + 1 + 2 + 2 + 2 + 3 + 3 + 3$
3. The external average depth of the tree `B` is: $21/6 = 3,5$ ($lce = 21 = 4 + 4 + 4 + 2 + 4 + 3$)

Solution 2 **(BST: search path – *2 points*)**

The sequences ② and ④ are impossible:

① 50 - 15 - 48 - 22 - 46 - 42
50, we go down to the left - 15, we go down to the right - 48 we go down to the left - 22, we go down to the right - 46, we go down to the left - **42**

② 48 - 15 - 45 - 22 - 47 - 42
48, we go down to the left - 15, we go down to the right - **45, we go down to the left** - 22, we go down to the right - **47 cannot be there, it is not lower than 45!**

③ 15 - 22 - 45 - 43 - 35 - 42
15, we go down to the right - 22, we go down to the right - 45, we go down to the left - 43, we go down to the left - 35, we go down to the right - **42**

④ 22 - 45 - 43 - 15 - 35 - 42
22, we go down to the right - 45, we go down to the left - 43, we go down to the left - **15 47 cannot be there, it is not higher than 22**

Solution 3 (Transpose - *3 points*)

Specifications:

The function `transpose(`A`)` builds and returns the transposed matrix of the non empty matrix A.

```
def buildTranspose(M):
    (l, c) = (len(M), len(M[0]))
    R = []
    for i in range(c):
        L = []
        for j in range(l):
            L.append(M[j][i])
        R.append(L)
    return R
```

Solution 4 (Vertical Symmetry – *5 points*)

Specifications:

The function `v_symmetric(`M`)` tests whether the matrix M has a horizontal axis of symmetry (vertical symmetry).

```
def v_symmetric(M):
    (l, c) = (len(M), len(M[0]))
    ldiv2 = l // 2
    i = 0
    test = True
    while i < ldiv2 and test:
        j = 0
        while j < c and test:
            test = M[i][j] == M[l-i-1][j]
            j += 1
        i += 1
    return test

def v_symmetric2(M):
    (l, c) = (len(M), len(M[0]))
    ldiv2 = l // 2
    (i, j) = (0, c)
    while i < ldiv2 and j == c:
        j = 0
        while j < c and M[i][j] == M[l-i-1][j]:
            j += 1
        i += 1
    return j == c
```

Solution 5 (Maximum Path Sum – *2 points*)

Specifications:

The function `maxpath(`B`)` returns the maximum value of the branches of the binary tree B (0 if the tree is empty).

```
def maxpath(B):
    if B == None:
        return 0
    else:
        return B.key + max(maxpath(B.left), maxpath(B.right))
```

Solution 6 (Full? – *3 points*)

Corrections below: Directly adapted from functions that test whether a tree is degenerate!

```python
# not the most optimized (to many test)
def full0(T):
    if T == None :  # this test might be in a call function!
        return True
    elif T.left == None or T.right == None: #single point
        return False
    else :
        return full0(T.left) and full0(T.right)

# the optimized version (only 2 tests each time)
def __full(B):
    '''
    B not empty
    '''
    if B.left == None:
        if B.right == None:
            return True
        else:
            return False
    else:
        if B.right == None:
            return False
        else:
            return __full(B.left) and full(B.right)

def full(B):
    return B == None or __full(B)

# a nice version
def __full2(B):
    '''
    B not empty
    '''
    leftEmpty = (B.left == None)
    if B.right == None:
        return leftEmpty
    else:
        return not leftEmpty and __full2(B.left) and __full2(B.right)

def full2(B):
    return B == None or __full2(B)
```

Solution 7 (Mystery – *2 points*)

```
>>> what(B)
[[5], [2, 12], [-1, 0, 4, 1], [4, 11, -2], [15]]
```