Algorithmics
Midterm #2 (C2)

Undergraduate 1st year S2#
Epita

novembre 2019

Instructions (read it) :

- ☐ You must answer on **the answer sheets provided.**
  - No other sheet will be picked up. Keep your rough drafts.
  - Answer within the provided space. **Answers outside will not be marked**: Use your drafts!
  - Do not separate the sheets unless they can be re-stapled before handing in.
  - Penciled answers will not be marked.
- ☐ The presentation is negatively marked, which means that you are marked out of 20 points and the presentation points (maximum of 2) are taken off this grade.
- ☐ **Code:**
  - All code must be written in the language `Python` (no C, Caml, Algo or anything else).
  - **Any `Python` code not indented will not be marked.**
  - All that you need (functions, methods) is indicated in the **appendix** (last page)!
- ☐ Duration : 2h

WOW! WHAT A PROJECT!
COLOR THESE PICTURES! CUT AND PASTE! DRAW THOSE TREES! MORE CUTTING! MORE PASTING!
WHAT A LEARNING EXPERIENCE! YES, MA'AM, YOU'VE DONE IT AGAIN!
WHEN SHE'S HAPPY, WE'RE HAPPY..
© 1997 United Feature Syndicate, Inc.

Du cours

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

Let the tree `B = {`ε`,0,1,00,01,10,11,000,011,110,111,0001,0110,0111,1101}`.

1. Using the hierarchical order number as label for the nodes, draw the tree `B`.
2. What is the internal path length of the tree `B`?
3. What is the externe average depth of the tree `B`?

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

Which sequences among the following could be the values encountered during the research of the value 42 in a binary search tree?

① 50 - 15 - 48 - 22 - 46 - 42

② 48 - 15 - 45 - 22 - 47 - 42

③ 15 - 22 - 45 - 43 - 35 - 42

④ 22 - 45 - 43 - 15 - 35 - 42

Matrices

Exercise 3 (Transpose - *Midterm S2# 2019*)

The transpose of a matrix A is the matrix A^{T} obtained by switching the rows and columns of the matrix A.

$$A = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \text{ then } A^{\mathsf{T}} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

Write the function `transpose` that constructs and returns the transposed matrix of the non empty matrix M.

Exercise 4 (Vertical Symmetry – *Midterm S2# 2019*)

Write the function `v_symmetric` that tests whether a matrix (assumed to be non empty) has a horizontal axis of symmetry (vertical symmetry).

1	1	10	10	7
10	0	9	3	8
3	1	4	7	5
0	8	1	1	1
3	1	4	7	5
10	0	9	3	8
1	1	10	10	7

Figure 1: `Mat1`

1	1	10	10	7
10	0	9	3	8
3	1	4	7	**5**
0	8	1	1	1
3	1	4	7	**0**
10	0	9	3	8
1	1	10	10	7

Figure 2: `Mat2`

1	1	10	10	7
10	0	9	3	8
3	1	4	7	5
3	1	4	7	5
10	0	9	3	8
1	1	10	10	7

Figure 3: `Mat3`

Application examples on matrices in figures 1, 2 and 3:

```
    >>> v_symmetric(Mat1)
    True
    >>> v_symmetric(Mat2)
    False
    >>> v_symmetric(Mat3)
    True
```

Binary Trees

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

In a binary tree we define the *value of a branch* as the sum of the values of the nodes in that branch.

Write the function `maxpath`(B) that computes the maximum value of the branches of the binary tree B (whose keys are integers).

For instance, in the tree in figure 4 the returned sum is 26 = 13 + 8 + 5

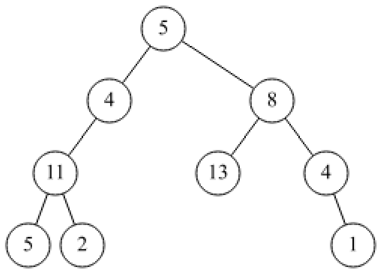

Figure 4: maximum = 26

Exercise 6 (Full? – *3 points*)

In a binary tree we define the *value of a branch* as the sum of the values of the nodes in that branch.

Exercise 7 (Mystery – *2 points*)

The function `what` below builds a list from a binary tree.

```python
def what(B):
    R = []
    if B != None:
        q = Queue()
        q.enqueue(B)
        q2 = Queue()
        L = []
        while not q.isempty():
            B = q.dequeue()
            L.append(B.key)
            if B.left != None:
                q2.enqueue(B.left)
            if B.right != None:
                q2.enqueue(B.right)
            if q.isempty():
                R.append(L)
                L = []
                (q, q2) = (q2, q)
    return R
```

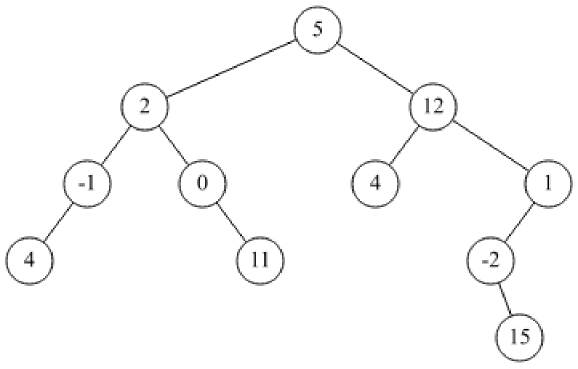

Figure 5: Arbre B

Give the result of the application, with B the tree in figure 5.

Appendix

Binary Trees

The binary trees we work on are the same as the ones in tutorials.

- `None` is the empty tree.
- The non-empty tree is an object of class `BinTree` which has 3 attributes: `key`, `left`, `right`.

```python
class BinTree:
    def __init__(self, key, left, right):
        self.key = key
        self.left = left
        self.right = right
```

Authorised functions and methods

On lists:

- `len`
- `append`

Others:

- `range`
- `abs`
- `min` and `max`, but only with two integer values!

Queues for the code to read

The methods of the class `Queue`, assumed imported:

- `Queue()` returns a new queue ;
- *q*`.enqueue(`*e*`)` enqueues *e* in *q* ;
- *q*`.dequeue()` deletes and returns the first element of *q* ;
- *q*`.isempty()` tests whether *q* is empty.

Your functions

You can write your own functions as long as they are documented (we have to known what they do).

In any case, the last written function should be the one which answers the question.