Algorithmique
Contrôle nº 2 (C2)

Info-sup (S2)
Epita

22 février 2017 - 9 : 30

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - — Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - — Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - — Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - — Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Le code :**
  - — Tout code doit être écrit dans le langage `Python` (pas de C, Caml, Algo ou autre).
  - — **Tout code `Python` non indenté ne sera pas corrigé.**
  - — Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué en **annexe** !
- ☐ Durée : 2h00

Cours

Exercice 1 (Il faut oser ... - *3 points*)

Soit l'arbre binaire suivant, représenté (statiquement) sous forme hiérarchique :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
T	E	C	D	E	F	R	A		R	S		A	E	E		U				E	

1. Représenter graphiquement l'arbre correspondant.
2. En considérant un parcours profondeur main gauche de cet arbre, donner la liste des nœuds dans l'ordre infixe de rencontre.
3. Donner sous forme d'occurrences ($B = \{\varepsilon, 0, 1, 01, 10, \ldots\}$) la représentation de cet arbre.

Matrices

Exercice 2 (Maximum Gap – *5 points*)

Pour cet exercice, on définit le *gap* (écart) d'une liste comme étant l'écart maximum entre deux valeurs de la liste. Par exemple, dans la matrice ci-dessous le *gap* de la première ligne est 13.

Écrire la fonction qui retourne le gap maximum des lignes d'une matrice (que l'on supposera non vide).

Exemple d'application avec la matrice `Mat1` *ci-contre :*

```
>>> maxGapMatrix(Mat1)
19
```

En effet le gap maximum est celui de la ligne du milieu ($19 = 14 - (-5)$).

1	10	3	0	-3	2	8
-1	0	1	8	5	0	-4
10	9	14	1	4	-5	1
10	-3	7	11	6	3	0
7	8	-5	1	5	4	10

`Mat1`

Exercice 3 (Synergistic Dungeon – *4 points*)

Les démons ont capturé le chevalier **(K)** et l'ont emprisonné dans la salle en bas à droite d'un donjon. Le donjon est composé de $m \times n$ salles disposées sur une grille 2D. Notre courageuse princesse **(P)** entre dans le donjon par la salle en haut à gauche. Elle doit se frayer un chemin à travers le donjon pour sauver le chevalier.

La princesse a un nombre de points de vie représenté par un entier positif. Si à un moment quelconque ses points de vie tombent à 0 ou moins, elle meurt immédiatement.

Certaines salles sont gardées par des démons (entiers strictement positifs), la princesse perd des points de vie en entrant dans ces salles. Les autres salles sont vides (0).

La princesse décide de se déplacer uniquement vers la droite ou vers le bas à chaque étape, afin de rejoindre le chevalire plus rapidement.

Par exemple, dans le donjon ci-dessous, la princesse doit avoir au moins 7 points de vie si elle emprunte le chemin optimal *droite* → *droite* → *bas* → *bas*.

(P) 2	3	0
2	10	0
3	0	1 **(K)**

Écrire la fonction `dungeon(`M`)`, avec M la matrice représentant le donjon (non vide), qui détermine le nombre minimum de points de vie que doit avoir la princesse pour sauver le chevalier (si elle suit le chemin optimal).

En plus / indice : il n'est pas nécessaire de construire une nouvelle matrice, celle en paramètre peut être modifiée.

Arbres binaires

Exercice 4 (Tests – *8 points*)

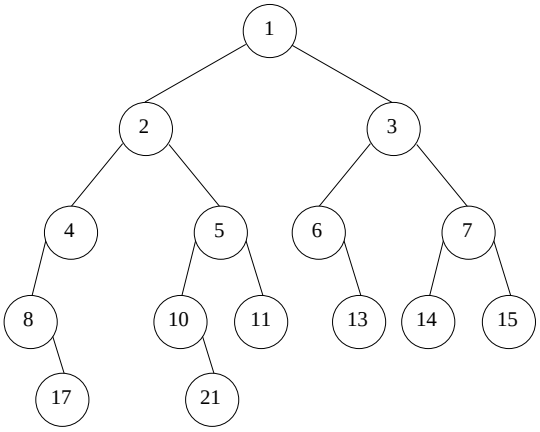

FIGURE 1 – Arbre binaire B

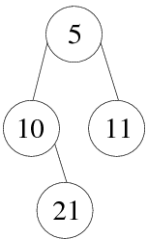

FIGURE 2 – Un sous-arbre de B

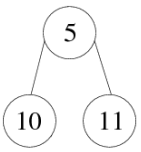

FIGURE 3 – Pas un sous-arbre de B

1. Écrire la fonction `equal` qui vérifie si deux arbres binaires sont identiques : ils contiennent les mêmes valeurs aux mêmes places.
2. Écrire la fonction `isSubTree(`S`,` B`)` qui vérifie si l'arbre binaire S est un sous-arbre de l'arbre binaire B. Il n'y a pas deux clés identiques dans B.
   Vous pouvez utiliser la fonction `equal`, qu'elle soit écrite ou non.

Annexes

Les arbres binaires

Les arbres binaires manipulés ici sont les mêmes qu'en td.

— `None` est l'arbre vide.

— L'arbre non vide a 3 attributs : `key`, `left`, `right`.

Fonctions et méthodes autorisées

Sur les listes :

— `len`

— `append`

Autres :

— `range`

— `abs`

— `min` et `max`, mais uniquement avec deux valeurs entières !

Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font).

Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.