Algorithmique
Contrôle n° 2 (C2)

Info-sup S2#
Epita

6 novembre 2017 - 8 : 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 (Représentations et questions ... – *4 points*)

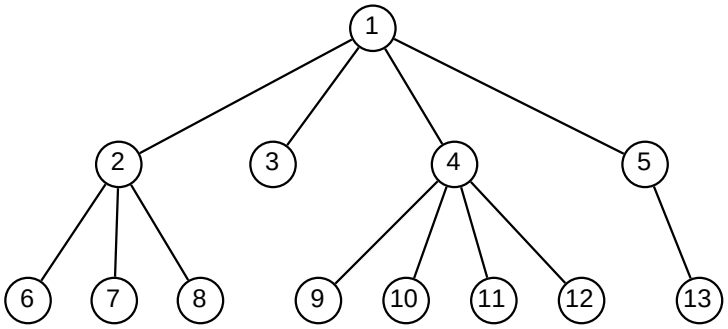


FIGURE 1 – Arbre général

1. Comment s'appelle la représentation d'un arbre général par un arbre binaire ?
2. Représenter l'arbre général de la figure 1 selon cette représentation binaire.
3. Comment calcule t'on la taille de l'arbre général à l'aide de l'arbre binaire le représentant ?
4. Comment calcule t'on la hauteur de l'arbre général à l'aide de l'arbre binaire le représentant ?
5. Lorsqu'on recherche une clé x qui est présente dans une collection de données, on parle de recherche ?

Exercice 2 (ABR : insertions – *3 points*)

On veut créer un arbre binaire de recherche par insertions successives, à partir d'un arbre vide, des valeurs `U, N, J, O, L, I, A, B, R`.

Donner le résultat (dessiner l'arbre final) lorsque ces valeurs sont ajoutées :

1. en feuille ;
2. en racine.

Matrices

Exercice 3 (Recherche – *4 points*)

Écrire la fonction `searchMatrix(M, x)` qui retourne la position (i, j) de la première valeur `x` trouvée dans la matrice `M` (que l'on supposera non vide). Si `x` n'est pas présent, la fonction retourne $(-1, -1)$.

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

```
>>>  searchMatrix(Mat1, -5)
(2, 5)
>>> searchMatrix(Mat1, 5)
(1, 4)
>>> searchMatrix(Mat1, 15)
(-1, -1)
```

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`

Arbres binaires

Exercice 4 (Symmetric – *4 points*)

Un arbre binaire est dit *symétrique* s'il est identique à son miroir (arbre renversé selon un axe vertical).

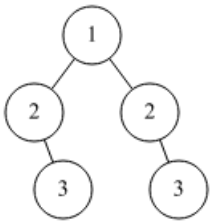


FIGURE 2 – Arbre non symétrique

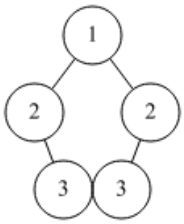


FIGURE 3 – Arbre symétrique

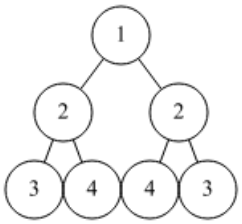


FIGURE 4 – Arbre symétrique

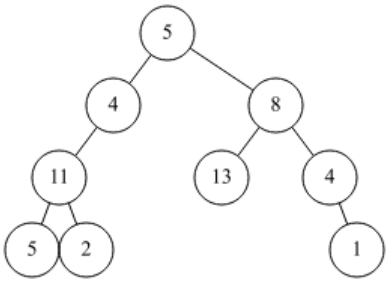
Écrire la fonction `symmetric` qui vérifie si un arbre binaire est symétrique. La méthode est laissée à votre imagination, mais bien entendu, plus elle sera optimisée, mieux ce sera !

Exercice 5 (Maximum Path Sum – *3 points*)

Dans un arbre binaire, nous définissons la *valeur d'une branche* comme étant la somme des valeurs de nœuds sur cette branche.

Écrire la fonction `maxpath(`B`)` qui détermine la plus grande valeur des branches de l'arbre binaire B (dont les clés sont des entiers).

Par exemple, dans l'arbre ci-contre, la somme retournée sera 26.
$5 + 8 + 13 = 26$



Exercice 6 (Mystery – *2 points*)

La fonction `mystery` ci-dessous construit un arbre binaire à partir d'une liste (on suppose ici, la classe `BinTree` importée).

```
def build(L, a, b):
    if a > b:
        return None
    else:
        c = (b - a) // 2 + a
        return BinTree(L[c], build(L, c+1, b), build(L, a, c-1))

def mystery(L):
    return build(L, 0, len(L)-1)
```

1. Dessiner l'arbre, résultat de l'application de la fonction `mystery` à la liste suivante `L = [4, 8, 2, 9, 5, 10, 1, 6, 11, 3, 12, 7, 13]`.
2. Quelles propriétés doit avoir la liste pour que le résultat soit :
   (a) un arbre binaire de recherche ?
   (b) un arbre complet ?

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.