

Sujet BAC 10 : Les arbres - Partie 1



1. 2022, Polynésie, J1, Ex. 5

BAC : Construction d'arbres binaires

On manipule ici les arbres binaires avec trois fonctions :

- `est_vide(A)` renvoie `True` si l'arbre binaire `A` est vide, `False` s'il ne l'est pas ;
- Pour un arbre binaire `A` **non vide** :
 - `sous_arbre_gauche(A)` renvoie le sous-arbre à gauche de `A` ;
 - `sous_arbre_droite(A)` renvoie le sous-arbre à droite de `A`.

L'arbre binaire renvoyé par les fonctions `sous_arbre_gauche` et `sous_arbre_droite` peut éventuellement être l'arbre vide.

On définit la hauteur d'un arbre binaire de la façon suivante :

- la hauteur de l'arbre vide est 0 ;
- sinon, la hauteur est égale à $1 + M$, où M est la plus grande des hauteurs de ses sous-arbres (à gauche et à droite).

1.a. Donner la hauteur de l'arbre ci-dessous.

```
graph TD
    N0( ) --> N2( )
    N0 --> N1( )
    N2 --> N5( )
    N2 --> N6( )
    linkStyle 3 stroke-width:0px;
    style N6 opacity:0;
```

Réponse



1.b. Dessiner sur la copie un arbre binaire de hauteur 5.

Réponse



La hauteur d'un arbre est calculée par l'algorithme récursif suivant :

Pseudo Code

```

1 Algorithme hauteur(A) :
2     si A vide :
3         renvoyer ...
4     sinon:
5         renvoyer 1 + max(
6             hauteur(sous_arbre_gauche(A)),
7             ...
8         )

```

2. Recopier sur la copie les lignes 3 et 7 en complétant les points de suspension.

✓ Réponse > |

On considère un arbre binaire R dont on note G le sous-arbre à gauche et D le sous-arbre à droite. On suppose que R est de hauteur 5 et G de hauteur 3.

3.a. Justifier le fait que D n'est pas l'arbre vide et déterminer sa hauteur.

✓ Réponse > |

3.b. Illustrer cette situation par un dessin.

✓ Réponse > |

Soit un arbre binaire non vide de hauteur h . On note n le nombre de noeuds de cet arbre. On admet que $h \leq n \leq 2^h - 1$.

4.a. Vérifier ces inégalités sur l'arbre binaire de la question 1.a..

✓ Réponse > |

4.b. Expliquer comment construire un arbre binaire de hauteur h quelconque ayant h noeuds.

✓ Réponse > |

4.c. Expliquer comment construire un arbre binaire de hauteur h quelconque ayant $2^h - 1$ noeuds.

Indication : $2^h - 1 = 1 + 2 + 4 + \dots + 2^{h-1}$.

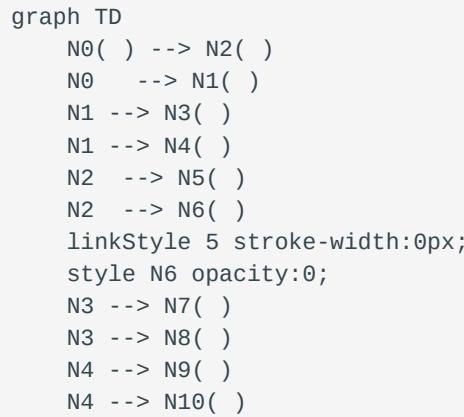
✓ Réponse > |

L'objectif de la fin de l'exercice est d'écrire le code d'une fonction `fabrique(h, n)` qui prend comme paramètres deux nombres entiers positifs `h` et `n` tels que $h < n < 2^h - 1$, et qui renvoie un arbre binaire de hauteur `h` à `n` noeuds.

Pour cela, on utilise les deux fonctions suivantes :

- `arbre_vide()`, qui renvoie un arbre vide ;
- `arbre(gauche, droite)` qui renvoie l'arbre fils à `gauche` et le fils à `droite`.

5. Recopier sur la copie l'arbre binaire ci-dessous et numérotter ses noeuds de 1 en 1 en commençant à 1, en effectuant un parcours en profondeur préfixe.



✓ Réponse

La fonction `fabrique` ci-dessous a pour but de répondre au problème posé.

Script Python

```
def fabrique(h, n):
    if n == 0:
        return ...
    else:
        reste = n - 1
        n_gauche = min(reste, ...)
        n_droite = reste - n_gauche
        return arbre(
            fabrique(..., n_gauche),
            ...
        )
```

6. Recopier sur la copie les lignes 3, 6 et 9 en complétant les points de suspension.

✓ Réponse

2. 2022, Métropole, J1, Ex. 4

 **BAC : Somme des valeurs d'un arbre binaire**

Cet exercice traite du calcul de la somme d'un arbre binaire. Cette somme consiste à additionner toutes les valeurs numériques contenues dans les nœuds de l'arbre.

L'arbre utilisé dans les parties A et B est le suivant :

```
graph TD
    N0(3) --> N1(6)
    N0 --> N2(2)
    N1 --> N4(7)
    N1 --> N5(4)
    N2 --> N6(9)
    N2 --> N7(1)
```

2.1. Partie A : Parcours d'un arbre

1. Donner la somme de l'arbre précédent. Justifier la réponse en explicitant le calcul qui a permis de l'obtenir.

 Réponse



2. Indiquer la lettre correspondante aux noms « racine », « feuille », « nœud », « SAG » (Sous Arbre à Gauche) et « SAD » (Sous Arbre à Droite). Chaque lettre A, B, C, D et E ne devra être utilisée qu'une seule fois.

```
flowchart TD
    subgraph G0 ["Arbre avec des lettres à associer"]
        N0(3) --> N1(6)
        N0 --> N2(2)
        subgraph G2 [ ]
            N2 --> N6(9)
            N2 --> N7(1)
        end
        subgraph G3 [ ]
            N1 --> N4(7)
            N1 --> N5(4)
        end
    end
    A>A] -.- N0
    B>B] -.- N1
    C>C] -.- N4
    G3 -.- D>D]
    G2 -.- E>E]
```

 Réponse



3. Parmi les quatre propositions A, B, C et D ci-dessous, donnant un parcours en largeur de l'arbre, une seule est correcte. Indiquer laquelle.

- **Proposition A** : 7 - 6 - 4 - 3 - 9 - 2 - 1
- **Proposition B** : 3 - 6 - 7 - 4 - 2 - 9 - 1
- **Proposition C** : 3 - 6 - 2 - 7 - 4 - 9 - 1
- **Proposition D** : 7 - 4 - 6 - 9 - 1 - 2 - 3

Réponse



4. Écrire en langage Python la fonction `somme` qui prend en paramètre une liste de nombres et qui renvoie la somme de ses éléments.

Exemple : `somme([1, 2, 3, 4])` est égale à 10.

Réponse



5. La fonction `parcourir(arbre)` pourrait se traduire en langage naturel par :

Pseudo-code

```
parcourir(A):
    L = liste_vide
    F = file_vide
    enfiler A dans F
    Tant que F n'est pas vide
        défiler S de F
        ajouter la valeur de la racine de S dans L
        Pour chaque sous arbre SA non vide de S
            enfiler SA dans F
    renvoyer L
```

Donner le type de parcours obtenu grâce à la fonction `parcourir`.

Réponse



2.2. Partie B : Méthode « diviser pour régner »

6. Parmi les quatre propositions A,B, C et D ci-dessous, indiquer la seule proposition correcte. En informatique, le principe diviser pour régner est associé à :

- **Proposition A** : diviser une fonction en deux fonctions de plus petit code.
- **Proposition B** : utiliser plusieurs modules
- **Proposition C** : séparer les informations en fonction de leur type
- **Proposition D** : découper un problème initial en sous-problèmes, à résoudre, puis combiner leurs solutions

Réponse

7. L'arbre présenté dans le problème peut être décomposé en racine et sous arbres :

```
graph TD
    N0(3) --> N1(6)
    N0 --> N2(2)
    subgraph "SAD"
        N2 --> N6(9)
        N2 --> N7(1)
    end
    subgraph "SAG"
        N1 --> N4(7)
        N1 --> N5(4)
    end
```

Indiquer dans l'esprit de « diviser pour régner » l'égalité donnant la somme d'un arbre en fonction de la somme des sous arbres et de la valeur numérique de la racine.

Réponse

8. Écrire en langage Python une fonction récursive `somme(arbre)`. Cette fonction renvoie la somme de l'arbre passé en paramètre.

Les fonctions suivantes sont disponibles :

- `est_vide(arbre)` : détermine si `arbre` est vide et renvoie un booléen `True` ou `False` .
- `valeur_racine(arbre)` : renvoie la valeur numérique de la racine de `arbre` ;
- `arbre_gauche(arbre)` : renvoie le sous arbre à gauche de `arbre` ;
- `arbre_droite(arbre)` : renvoie le sous arbre à droite de `arbre` .

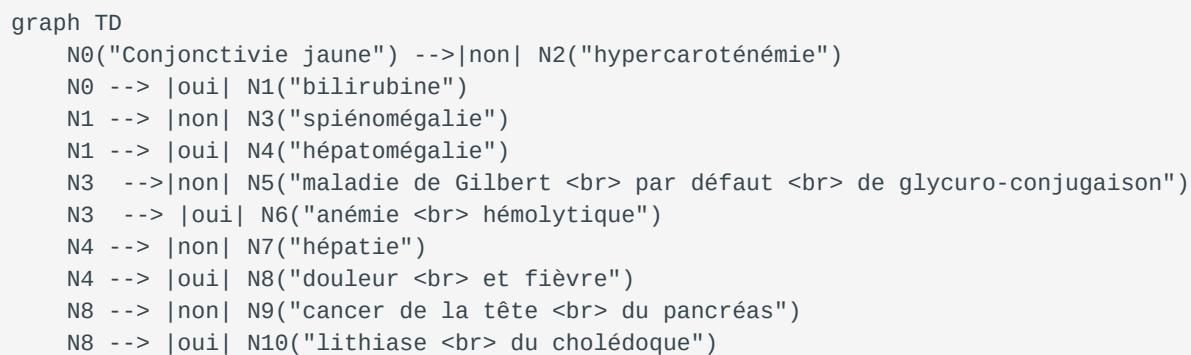
Réponse

3. 2022 Asie J1 - Ex3

BAC : Arbres binaires

Les premiers travaux concernant l'aide à la décision médicale se sont développés pendant les années soixante-dix parallèlement à l'avènement de l'informatique dans le secteur médical. L'arbre de décision est une technique

décisionnelle fréquemment employée pour rechercher la meilleure stratégie thérapeutique. L'arbre de décision de cet exercice, présenté ci-dessous, est un arbre binaire que l'on nommera `arb_decision`.



Arbre de décision en présence d'une jaunisse (peau anormalement jaune) chez un patient.

Rappels :

- Un arbre binaire est une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément, appelé nœud, porte une étiquette.
- Le nœud initial est appelé racine.
- Chaque nœud d'un arbre binaire possède au plus deux sous-arbres.
- Chacun de ces sous-arbres est un arbre binaire, appelés sous-arbre gauche et sous-arbre droit.
- Un nœud dont les sous-arbres sont vides est appelé une feuille.
- Dans cet exercice, on utilisera la convention suivante : la hauteur d'un arbre binaire ne comportant qu'un nœud est égale à 1.

Dans l'arbre de décision en présence d'une jaunisse chez un patient,

- un nœud représente un symptôme dont le médecin doit étudier la présence ou l'absence ; la réponse ne peut être que oui ou non ;
- le sous-arbre gauche d'un nœud donné décrit la démarche à adopter si le symptôme est absent ;
- le sous-arbre droit d'un nœud donné décrit la démarche à adopter si le symptôme est présent ;
- l'étiquette d'une feuille est la maladie induite par le chemin parcouru.

Questions :

1. Déterminer la taille et la hauteur de l'arbre donné en exemple en introduction (arbre de décision en présence d'une jaunisse).

 **Réponse**



2. On choisit d'implémenter un arbre binaire à l'aide d'un dictionnaire.

 **Script Python**

```
arbre_vide = []
arbre = {'etiquette': 'valeur' ,
         'sag': sous_arbre_gauche ,
         'sad': sous_arbre_droit }
```

Le code ci-dessous représente un arbre selon le modèle précédent.

Script Python

```
{'etiquette' : 'a',
 'sag': {'etiquette' : 'b',
          'sag': {},
          'sad' : {'etiquette' : 'd',
                    'sag' : {},
                    'sad' : {}}},
 'sad': {'etiquette' : 'f',
          'sag' : {'etiquette' : 'g',
                    'sag' : {},
                    'sad' : {}},
          'sad' : {}}}
```

a. À quelle représentation graphique correspond la structure implémentée ci-dessus ?

- arbre 1 :

```
graph TD
    A("a") --> B("b")
    B --> d("d")
    B --> d1(" ")
    A --> F("f")
    F --> G("g")
    F --> G1(" ")
    linkStyle 2 stroke-width:0px;
    style d1 opacity:0;
    linkStyle 5 stroke-width:0px;
    style G1 opacity:0;
```

- arbre 2 :

```
graph TD
    A("a") --> B("b")
    B --> d(" ")
    B --> d1("d")
    A --> F("f")
    F --> G("g")
    F --> G1(" ")
    linkStyle 1 stroke-width:0px;
    style d opacity:0;
    linkStyle 5 stroke-width:0px;
    style G1 opacity:0;
```

- arbre 3 :

```
graph TD
    A("a") --> B("b")
    B --> d(" ")
```

```

B --> d1("d")
A --> F("f")
F --> G(" ")
F --> G1("g")
linkStyle 1 stroke-width:0px;
style d opacity:0;
linkStyle 4 stroke-width:0px;
style G opacity:0;

```

✓ Réponse


- b. Représenter graphiquement l'arbre correspondant au code ci-dessous.

duck Script Python

```

{'etiquette' : 'H',
'sag':{ 'etiquette' : 'G',
'sag': { 'etiquette' : 'E',
'sag' : {},
'sad' : {}},
'sad' : { 'etiquette' : 'D',
'sag' : {},
'sad' : { 'etiquette' : 'B',
'sag' : {},
'sad' : {} } },
'sad': { 'etiquette' : 'F',
'sag' : { 'etiquette' : 'C',
'sag' : {},
'sad' : { 'etiquette' : 'A',
'sag' : {} },
'sad' : {} } },
'sad' : {} }

```

✓ Réponse


3. La fonction parcours(arb) ci-dessous permet de réaliser le parcours des nœuds d'un arbre binaire arb donné en argument.

duck Script Python

```

def parcours(arb):
    if arb == {}:
        return None
    parcours(arb['sag'])
    parcours(arb['sad'])
    print(arb['etiquette'])

```

- a. Donner l'affichage après l'appel de la fonction parcours avec l'arbre dont une représentation graphique est ci-dessous.

```

graph TD
    A("a") --> B("b")
    B --> d("d")
    B --> d1(" ")
    A --> F("f")
    F --> G("g")
    F --> G1(" ")
    linkStyle 2 stroke-width:0px;
    style d1 opacity:0;
    linkStyle 5 stroke-width:0px;
    style G1 opacity:0;

```

✓ Réponse > |

b. Écrire une fonction `parcours_maladies(arb)` qui n'affiche que les feuilles de l'arbre binaire non vide arb passé en argument, ce qui correspond aux maladies possiblement induites par l'arbre de décision.

✓ Réponse > |

4. On souhaite maintenant afficher l'ensemble des symptômes relatifs à une maladie. On considère la fonction `symptomes(arbre, mal)` avec comme argument arbre un arbre de décision binaire et mal le nom d'une maladie.

L'appel de cette fonction sur l'arbre de décision `arb_decision` de l'introduction fournit les affichages suivants.

🐍 Script Python

```

>>> symptomes(arb_decision, "anémie hémolytique")
symptômes de anémie hémolytique
splénomégalie
pas de bilirubine
conjonctive jaune

```

Pour cela, on modifie la structure précédente en ajoutant une clé `surChemin` qui sera un booléen indiquant si le nœud est sur le chemin de la maladie.

La clé `surChemin` est initialisée à False pour tous les nœuds.

🐍 Script Python

```

arbre = {'etiquette': 'valeur' ,
         'surChemin': False ,
         'sag': 'sous-arbre gauche' ,
         'sad': 'sous-arbre droit' }

```

Recopier et compléter les lignes 6, 8, 14 et 18 du code suivant sur votre copie.

🐍 Script Python

```

1 def symptomes(arb, mal):
2     if arb['sag'] != {} :

```

```
3     symptomes(arb['sag'],mal)
4
5     if arb['sad'] != {} :
6         symptomes(...)

7
8     if ... ... ... ... :
9         arb['surChemin'] = True
10        print('symptômes de', arb['etiquette'],':')
11
12 else :
13     if arb['sad'] != {} and arb['sad']['surChemin'] :
14         print(...)
15         arb['surChemin'] = True
16
17     if arb['sag'] != {} and arb['sag']['surChemin'] :
18         print(...)
19         arb['surChemin'] = True
```

✓ Réponse

