

Sujet BAC 12 : Arbres Binaires de Recherche



1. 2020, sujet 0

2020, sujet 0

Question 1

Déterminer la taille et la hauteur de l'arbre binaire suivant :

```
graph TD
    A("A") --> B("B")
    B --> C("C")
    B --> D("D")
    A --> E("E")
    C --> C1(" ")
    C --> C2(" ")
    D --> G("G")
    D --> D2(" ")
    E --> F("F")
    E --> E2(" ")
    F --> H("H")
    F --> I("I")
    E2 --> E3(" ")
    E2 --> E4(" ")
    linkStyle 4 stroke-width:0px;
    style C1 opacity:0;
    linkStyle 5 stroke-width:0px;
    style C2 opacity:0;
    linkStyle 7 stroke-width:0px;
    style D2 opacity:0;
    linkStyle 9 stroke-width:0px;
    style E2 opacity:0;
    linkStyle 12 stroke-width:0px;
    style E3 opacity:0;
    linkStyle 13 stroke-width:0px;
    style E4 opacity:0;
```

corrigé



Question 2

On décide de numérotter en binaire les nœuds d'un arbre binaire de la façon suivante :

- la racine correspond à 1 ;
- la numérotation pour un fils gauche s'obtient en ajoutant le chiffre 0 à droite au numéro de son père ;
- la numérotation pour un fils droit s'obtient en ajoutant le chiffre 1 à droite au numéro de son père ;

Par exemple, dans l'arbre ci-dessous, on a utilisé ce procédé pour numérotter les nœuds A, B, C, E et F .

```
graph TD
    A("A : 1") --> B("B : 10")
    B --> C("C : 100")
    B --> D("D : ?")
    A --> E("E : 11")
    C --> C1(" ")
    C --> C2(" ")
    D --> G("G: ?")
    D --> D2(" ")
    E --> F("F : 110")
    E --> E2(" ")
    F --> H("H : ?")
    F --> I("I : ?")
    E2 --> E3(" ")
    E2 --> E4(" ")
    linkStyle 4 stroke-width:0px;
    style C1 opacity:0;
    linkStyle 5 stroke-width:0px;
    style C2 opacity:0;
    linkStyle 7 stroke-width:0px;
    style D2 opacity:0;
    linkStyle 9 stroke-width:0px;
    style E2 opacity:0;
    linkStyle 12 stroke-width:0px;
    style E3 opacity:0;
    linkStyle 13 stroke-width:0px;
    style E4 opacity:0;
```

1. Dans l'exemple précédent, quel est le numéro en binaire associé au nœud G ?
2. Quel est le nœud dont le numéro en binaire vaut 13 en décimal ?
3. En notant h la hauteur de l'arbre, sur combien de bits seront numérotés les nœuds les plus en bas ?
4. Justifier que pour tout arbre de hauteur h et de taille $n \geq 2$, on a : $h \leq n \leq 2^h - 1$

 corrigé

>

? Question 3

Un arbre binaire est dit complet si tous les niveaux de l'arbre sont remplis.

```
graph TD
    A("A") --> B("B")
    B --> D("D")
    B --> E("E")
    A --> C("C")
    C --> F("F")
    C --> G("G")
    D --> H("H")
    D --> I("I")
    E --> J("J")
    E --> K("K")
    F --> L("L")
    F --> M("M")
    G --> N("N")
    G --> O("O")
```

On décide de représenter un arbre binaire complet par un tableau de taille $n + 1$, où n est la taille de l'arbre, de la façon suivante :

- La racine a pour indice 1 ;
- Le fils gauche du nœud d'indice i a pour indice $2 \times i$;
- Le fils droit du nœud d'indice i a pour indice $2 \times i + 1$;
- On place la taille n de l'arbre dans la case d'indice 0.

Répondre aux questions suivantes :

1. Déterminer le tableau qui représente l'arbre binaire complet de l'exemple précédent.
2. On considère le père du nœud d'indice i avec $i \geq 2$. Quel est son indice dans le tableau ?

✓ corrigé


Question 4

On se place dans le cas particulier d'un arbre binaire de recherche complet où les noeuds contiennent des entiers et pour lequel la valeur de chaque noeud est supérieure à celles des noeuds de son fils gauche, et inférieure à celles des noeuds de son fils droit.

Écrire une fonction `recherche` ayant pour paramètres un arbre `arbre` et un élément `element`. Cette fonction renvoie `True` si `element` est dans l'arbre et `False` sinon. L'arbre sera représenté par un tableau comme dans la question précédente.

corrigé



2. 2021, Métropole sujet 1

2021, Métropole sujet 1

Dans cet exercice, les arbres binaires de recherche ne peuvent pas comporter plusieurs fois la même clé. De plus, un arbre binaire de recherche limité à un noeud a une hauteur de 1. On considère l'arbre binaire de recherche représenté ci-dessous (figure 1), où `val` représente un entier :

```
graph TD
    A(18) --> B(15)
    B --> D(13)
    B --> E(val)
    A --> C(23)
    C --> F(19)
    C --> G(32)
    D --> H(12)
    D --> D1(" ")
    E --> E1(" ")
    E --> E2(" ")
    F --> F1(" ")
    F --> M(21)
    G --> G1(" ")
    G --> G2(" ")
    linkStyle 7 stroke-width:0px;
    style D1 opacity:0;
    linkStyle 8 stroke-width:0px;
    style E1 opacity:0;
    linkStyle 9 stroke-width:0px;
    style E2 opacity:0;
    linkStyle 10 stroke-width:0px;
    style F1 opacity:0;
    linkStyle 12 stroke-width:0px;
    style G1 opacity:0;
    linkStyle 13 stroke-width:0px;
    style G2 opacity:0;
```

Question 1

- a Donner le nombre de feuilles de cet arbre et préciser leur valeur (étiquette).
- b Donner le sous arbre-gauche du nœud 23.
- c Donner la hauteur et la taille de l'arbre.
- d Donner les valeurs entières possibles de `val` pour cet arbre binaire de recherche.

corrigé

On suppose, pour la suite de cet exercice, que `val` est égal à 16.

Question 2

On rappelle qu'un parcours infixé depuis un nœud consiste, dans l'ordre, à faire un parcours infixé sur le sous arbre-gauche, afficher le nœud puis faire un parcours infixé sur le sous-arbre droit.
Dans le cas d'un parcours suffixe, on fait un parcours suffixe sur le sous-arbre gauche puis un parcours suffixe sur le sous-arbre droit, avant d'afficher le nœud.

- a. Donner les valeurs d'affichage des nœuds dans le cas du parcours infixé de l'arbre.
- b. Donner les valeurs d'affichage des nœuds dans le cas du parcours suffixe de l'arbre.

corrigé

Question 3

On considère la classe `Noeud` définie de la façon suivante en Python :

Script Python

```
class Noeud():
    def __init__(self, v):
        self.ag = None
        self.ad = None
        self.v = v

    def insere(self, v):
        n = self
        est_insere = False
        while not est_insere :
            if v == n.v:
                est_insere = True | Bloc 1
            elif v < n.v:
                if n.ag != None:
                    n = n.ag
                else:
                    n.ag = Noeud(v)
                    est_insere = True | Bloc 2
            else:
                if n.ad != None:
                    n = n.ad
                else:
                    n.ad = Noeud(v)
                    est_insere = True | Bloc 3

    def insere_tout(self, vals):
        for v in vals:
            self.insere(v)
```

a. Représenter l'arbre construit suite à l'exécution de l'instruction suivante :

Script Python

```
racine = Noeud(18)
racine.insere_tout([12, 13, 15, 16, 19, 21, 32, 23])
```

b. Écrire les deux instructions permettant de construire l'arbre de la figure 1. On rappelle que le nombre `val` est égal à 16.

c. On considère l'arbre tel qu'il est présenté sur la figure 1. Déterminer l'ordre d'exécution des blocs (repérés de 1 à 3) suite à l'application de la méthode `insere(19)` au nœud racine de cet arbre.

 corrigé



Question 4

Écrire une méthode `recherche(self, v)` qui prend en argument un entier `v` et renvoie la valeur `True` si cet entier est une étiquette de l'arbre, `False` sinon.

 corrigé

> |

3. 2021, Métropole Candidats Libres 2

2021, Métropole Candidats Libres 2

On rappelle qu'un arbre binaire est composé de nœuds, chacun des nœuds possédant éventuellement un sous-arbre gauche et éventuellement un sous-arbre droit. Un nœud sans sous-arbre est appelé feuille. La taille d'un arbre est le nombre de nœuds qu'il contient ; sa hauteur est le nombre de nœuds du plus long chemin qui joint le nœud racine à l'une des feuilles. Ainsi la hauteur d'un arbre réduit à un nœud, c'est-à-dire la racine, est 1.

Dans un arbre binaire de recherche, chaque nœud contient une clé, ici un nombre entier, qui est :

- strictement supérieure à toutes les clés des nœuds du sous-arbre gauche ;
- strictement inférieure à toutes les clés des nœuds du sous-arbre droit.

Un arbre binaire de recherche est dit « bien construit » s'il n'existe pas d'arbre de hauteur inférieure qui pourrait contenir tous ses nœuds.

On considère l'arbre binaire de recherche ci-dessous.

```
graph TD
    A(12) --> B(10)
    B --> D(5)
    B --> E(" ")
    A --> C(15)
    C --> F(" ")
    C --> G(20)
    D --> H(4)
    D --> D1(8)
    E --> E1(" ")
    E --> E2(" ")
    F --> F1(" ")
    F --> M(" ")
    G --> G1(" ")
    G --> G2(" ")
    linkStyle 2 stroke-width:0px;
    style E opacity:0;
    linkStyle 4 stroke-width:0px;
    style F opacity:0;
    linkStyle 2 stroke-width:0px;
    style E opacity:0;
    linkStyle 8 stroke-width:0px;
```

```
style E1 opacity:0;  
linkStyle 9 stroke-width:0px;  
style E2 opacity:0;  
linkStyle 10 stroke-width:0px;  
style M opacity:0;  
linkStyle 11 stroke-width:0px;  
style F1 opacity:0;  
linkStyle 12 stroke-width:0px;  
style G1 opacity:0;  
linkStyle 13 stroke-width:0px;  
style G2 opacity:0;
```

?

Question 1

- a. Quelle est la taille de l'arbre ci-dessus ?
- b. Quelle est la hauteur de l'arbre ci-dessus ?

 corrigé



?

Question 2

Cet arbre binaire de recherche n'est pas « bien construit ». Proposer un arbre binaire de recherche contenant les mêmes clés et dont la hauteur est plus petite que celle de l'arbre initial.

 corrigé



Question 3

Les classes `Noeud` et `Arbre` ci-dessous permettent de mettre en œuvre en Python la structure d'arbre binaire de recherche. La méthode `insere` permet d'insérer récursivement une nouvelle clé.

Script Python

```

1 class Noeud :
2
3     def __init__(self, cle):
4         self.cle = cle
5         self.gauche = None
6         self.droit = None
7
8     def insere(self, cle):
9         if cle < self.cle :
10             if self.gauche == None :
11                 self.gauche = Noeud(cle)
12             else :
13                 self.gauche.insere(cle)
14         elif cle > self.cle :
15             if self.droit == None :
16                 self.droit = Noeud(cle)
17             else :
18                 self.droit.insere(cle)
19
20 class Arbre :
21
22     def __init__(self, cle):
23         self.racine = Noeud(cle)
24
25     def insere(self, cle):
26         self.racine.insere(cle)

```

Donner la représentation de l'arbre codé par les instructions ci-dessous.

Script Python

```

a = Arbre(10)
a.insere(20)
a.insere(15)
a.insere(12)
a.insere(8)
a.insere(4)
a.insere(5)

```

corrigé



Question 4

Pour calculer la hauteur d'un arbre non vide, on a écrit la méthode ci-dessous dans la classe `Noeud`.

Script Python

```
def hauteur(self):
    if self.gauche == None and self.droit == None:
        return 1
    if self.gauche == None:
        return 1 + self.droit.hauteur()
    elif self.droit == None:
        return 1 + self.gauche.hauteur()
    else:
        hg = self.gauche.hauteur()
        hd = self.droit.hauteur()
        if hg > hd:
            return hg + 1
        else:
            return hd + 1
```

Écrire la méthode `hauteur` de la classe `Arbre` qui renvoie la hauteur de l'arbre.

corrigé

Question 5

Écrire les méthodes `taille` des classes `Noeud` et `Arbre` permettant de calculer la taille d'un arbre.

corrigé

Question 6

On souhaite écrire une méthode `bien_construit` de la classe `Arbre` qui renvoie la valeur `True` si l'arbre est « bien construit » et `False` sinon.

On rappelle que la taille maximale d'un arbre binaire de recherche de hauteur h est $2^h - 1$.

- Quelle est la taille minimale, notée `min` d'un arbre binaire de recherche « bien construit » de hauteur h ?
- Écrire la méthode `bien_construit` demandée.

 corrigé

> |

4. 2021, Polynésie

2021, Polynésie

Cet exercice traite principalement du thème « algorithmique, langages et programmation » et en particulier les arbres binaires de recherche. La première partie aborde les arbres en mode débranché via l'application d'un algorithme sur un exemple.

La suivante porte sur la programmation orientée objet. La dernière partie fait le lien avec les algorithmes de tri.

Partie A : Étude d'un exemple

Considérons l'arbre binaire de recherche ci-dessous :

```
graph TD
    A(5) --> B(2)
    B --> D(" ")
    B --> E(3)
    A --> C(7)
    C --> F(" ")
    C --> G(8)
    linkStyle 1 stroke-width:0px;
    style D opacity:0;
    linkStyle 4 stroke-width:0px;
    style F opacity:0;
```

Question A.1

Indiquer quelle valeur a le nœud racine et quels sont les fils de ce nœud.

Réponse

> |

Question A.2

Indiquer quels sont les nœuds de la branche qui se termine par la feuille qui a pour valeur 3.

Réponse

> |

?

Question A.3

Dessiner l'arbre obtenu après l'ajout de la valeur 6.

✓ Réponse



Partie B : Implémentation en Python

Voici un extrait d'une implémentation en Python d'une classe modélisant un arbre binaire de recherche.

Script Python

```

1 class ABR:
2     """Implémentation d'un arbre binaire de recherche (ABR)"""
3     def __init__(self, valeur=None):
4         self.valeur = valeur
5         self.fg = None
6         self.fd = None
7
8     def estVide(self):
9         return self.valeur == None
10
11    def insererElement(self, e):
12        if self.estVide():
13            self.valeur = e
14        else:
15            if e < self.valeur:
16                if self.fg:
17                    self.fg.insererElement(e)
18                else:
19                    self.fg = ABR(e)
20            if e > self.valeur:
21                if self.fd:
22                    self.fd.insererElement(e)
23                else:
24                    self.fd = ABR(e)

```

?

Question B.1

Expliquer le rôle de la fonction `__init__`.

✓ Réponse



?

Question B.2

Dans cette implémentation, expliquer ce qui se passe si on ajoute un élément déjà présent dans l'arbre.

Réponse



Question B.3

Recopier et compléter les pointillés ci-dessous permettant de créer l'arbre de la partie A.

Script Python

```
arbre = ABR(.....)
arbre.insererElement(2)
arbre.insererElement(.....)
arbre.insererElement(7)
arbre.insererElement(.....)
```

Réponse

**Partie C : Tri par arbre binaire de recherche**

On souhaite trier un ensemble de valeurs entières distinctes grâce à un arbre binaire de recherche. Pour cela, on ajoute un à un les éléments de l'ensemble dans un arbre initialement vide. Il ne reste plus qu'à parcourir l'arbre afin de lire et de stocker dans un tableau résultat les valeurs dans l'ordre croissant.

Question C.1

Donner le nom du parcours qui permet de visiter les valeurs d'un arbre binaire de recherche dans l'ordre croissant.

Réponse



Question C.2

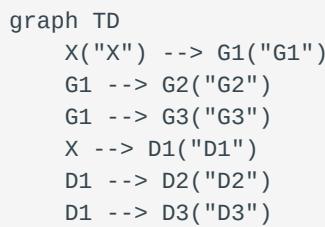
Comparer la complexité de cette méthode de tri avec celle du tri par insertion ou du tri par sélection.

Réponse

**5. 2021, Centres Étrangers, sujet 1**

 2021, Centres Etrangers, sujet 1

Un arbre binaire est soit vide, soit un nœud qui a une valeur et au plus deux fils (le sous-arbre gauche et le sous-arbre droit).



- X est un nœud, sa valeur est X.valeur
- G1 est le fils gauche de X, noté X.fils_gauche
- D1 est le fils droit de X, noté X.fils_droit

Un arbre binaire de recherche est ordonné de la manière suivante :

Pour chaque nœud X,

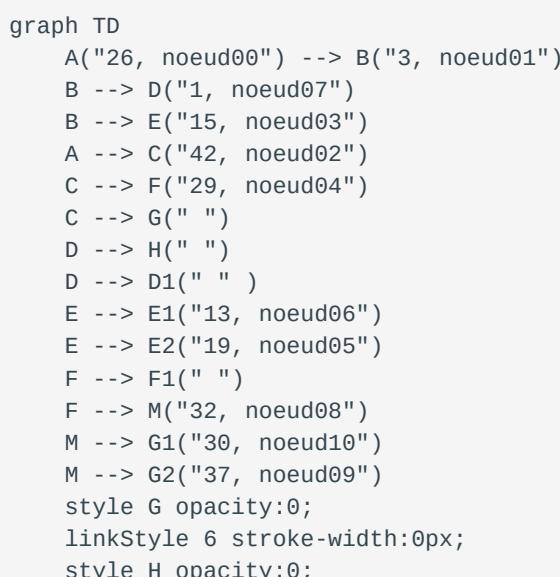
- les valeurs de tous les nœuds du sous-arbre gauche sont **strictement inférieures** à la valeur du nœud X
- les valeurs de tous les nœuds du sous-arbre droit sont **supérieures ou égales** à la valeur du nœud X.

Ainsi, par exemple, toutes les valeurs des nœuds G1, G2 et G3 sont strictement inférieures à la valeur du nœud X et toutes les valeurs des nœuds D1, D2 et D3 sont supérieures ou égales à la valeur du nœud X.

Voici un exemple d'arbre binaire de recherche dans lequel on a stocké dans cet ordre les valeurs : [26, 3, 42, 15, 29, 19, 13, 1, 32, 37, 30]

L'étiquette d'un nœud indique la valeur du nœud suivie du nom du nœud. Les nœuds ont été nommés dans l'ordre de leur insertion dans l'arbre ci-dessous.

'29, noeud04' signifie que le nœud nommé `noeud04` possède la valeur 29.



```
linkStyle 7 stroke-width:0px;  
style D1 opacity:0;  
style F1 opacity:0;
```

?

Question 1

On insère la valeur 25 dans l'arbre, dans un nouveau nœud nommé nœud11.

Recopier l'arbre binaire de recherche étudié et placer la valeur 25 sur cet arbre en coloriant en rouge le chemin parcouru.

Préciser sous quel nœud la valeur 25 sera insérée et si elle est insérée en fils gauche ou en fils droit, et expliquer toutes les étapes de la décision.

✓ Réponse

?

Question 2

Préciser toutes les valeurs entières que l'on peut stocker dans le nœud fils gauche du nœud04 (vide pour l'instant), en respectant les règles sur les arbres binaires de recherche ?

✓ Réponse

?

Question 3

Voici un algorithme récursif permettant de parcourir et d'afficher les valeurs de l'arbre :

duck Script Python

```
Parcours(A) # A est un arbre binaire de recherche  
    Afficher(A.valeur)  
    Parcours(A.fils_gauche)  
    Parcours(A.fils_droit)
```

✓ Réponse

? Question 3

- a. Écrire la liste de toutes les valeurs dans l'ordre où elles seront affichées.
- b. Choisir le type de parcours d'arbres binaires de recherche réalisé parmi les propositions suivantes : Préfixe, Suffixe ou Infixe.

✓ Réponse

> |

? Question 4

En vous inspirant de l'algorithme précédent, écrire un algorithme Parcours2 permettant de parcourir et d'afficher les valeurs de l'arbre A dans l'ordre croissant.

✓ Réponse

> |