



Questionnaire examen final

16,25/20

INF2010

Sigle du cours

<i>Identification de l'étudiant(e)</i>			<i>Réervé</i>
Nom	Prénom :		
Signature :	Matricule	Groupe :	
<i>Sigle et titre du cours</i>			
INF2010 – Structures de données et algorithmes			
Professeur	Groupe	Trimestre	
Tarek Ould-Bachir	Tous	20203	
Jour	Date	Durée	Heures
Samedi	19 décembre	2h30	13h30 – 16h00
Documentation		Calculatrice	Outils électroniques
<input type="checkbox"/> Aucune <input checked="" type="checkbox"/> Toute <input type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input checked="" type="checkbox"/> Toutes	Les appareils électroniques personnels sont interdits.
Directives particulières			
<ul style="list-style-type: none"> Le professeur ne répondra à aucune question durant cet examen. Si vous estimez que vous ne pouvez pas répondre à une question pour diverses raisons, veuillez le justifier puis passer à la question suivante. Il est strictement interdit de débrocher l'examen. IMPORTANT : inscrire votre matricule sur toutes les pages numérotées. $2 + (11 - \cancel{1} \% \cancel{1}) = 5$ 			
Cet examen contient 6 questions sur un total de 20 pages (inclus cette page).			

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 : Tables de dispersement**(4.5/20 points)**

Soit une table de dispersement double avec un facteur de compression maximal de 75%. En admettant que $\text{Hash}(x) = (\text{H}_1(x) + i \text{H}_2(x)) \% N$ où

$$x \geq 0 ;$$

$$\text{H}_1(x) = x \% N ;$$

$$\text{H}_2(x) = R - (x \% R) ;$$

N est la taille de la table, un nombre premier ;

R est un nombre premier tel que $0 < R < N$.

i est le nombre de collisions survenues à date lors de l'insertion de x

1.1) (3 points) En vous servant du tableau ci-dessous, donnez l'état de la mémoire d'une table de dispersement de taille $N = 13$ après l'insertion, dans l'ordre, des clés suivantes (on prendra $R = 11$):

15, 41, 122, 38, 180, 357.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Entrées			15		357	41	180					38	122

Pour chacune des clés, donnez ci-après le détail de vos calculs (non noté) et indiquez le nombre de collisions :

$x = 15 :$ $15 \% 13 = 2$

Nombre total de collisions : <u>0</u>
$x = 41 :$ $41 \% 13 = 2$ occupé $2 + (11 - (41 \% 11)) = 5$
Nombre total de collisions : <u>1</u>

$x = 122 :$

$$122\%13=5 \text{ occupé} \quad (5+2(10))\%13 = 12$$

$$11-(122\%11) = 10$$

$$5+10=15$$

$$15\%13 = 2 \text{ occupé}$$

Nombre total de collisions : 2

$x = 38 :$

$$38\%13=12 \text{ occupé}$$

$$11-(38\%11) = 6$$

$$(12+6)\%13=5 \text{ occupé}$$

$$(12+(2*6))\%13 = 11$$

Nombre total de collisions : 2

$x = 180 :$

$$180\%13=11 \text{ occupé}$$

$$11-(180\%11) = 7$$

$$(11+7)\%13 = 5 \text{ occupé}$$

$$(11+14)\%13 = 12 \text{ occupé}$$

$$(11+21)\%13=6$$

Nombre total de collisions : 3

$x = 357 :$

$$357\%13=6 \text{ occupé} \quad (6+18)\%13 = 11 \text{ occupé}$$

$$11-(357\%11)=6 \quad (6+24)\%13 = 4$$

$$6+6 = 12 \text{ occupé}$$

$$(6+12)\%13 = 5 \text{ occupé}$$

Nombre total de collisions : 4

1.2) (**0.75 point**) Existe-t-il une 7^e clé $x_7 \geq 0$ telle que l'insertion de x_7 ne produise aucune collision ? Justifiez votre réponse.

Oui, si $x_7 \% 13 = (0 \text{ ou } 1 \text{ ou } 4 \text{ ou } 7 \text{ ou } 8 \text{ ou } 9 \text{ ou } 10)$ ->(une case vide du tableau)

Par exemple, on pourrait placer la valeur 1 sans collision parce que $1 \% 13 = 1$ et que la case 1 est vide.

1.3) (**0.75 point**) Existe-t-il une 7^e clé $x_7 \geq 0$ telle que l'insertion de x_7 produise un cycle infini de collisions sans qu'il ne soit jamais possible d'insérer cette 7^e clé ? Justifiez votre réponse.

Oui, pour éviter que cela se produise, deux conditions doivent être respectées:

- 1) la taille de la table est un nombre premier : respectée
- 2) le facteur de compression est inférieur ou égale à 50% : non respectée

Avec une 7e valeur on arrive à un 50% de la table remplie, mais on ne rehash pas, un cycle infini pourrait donc se produire

0/1

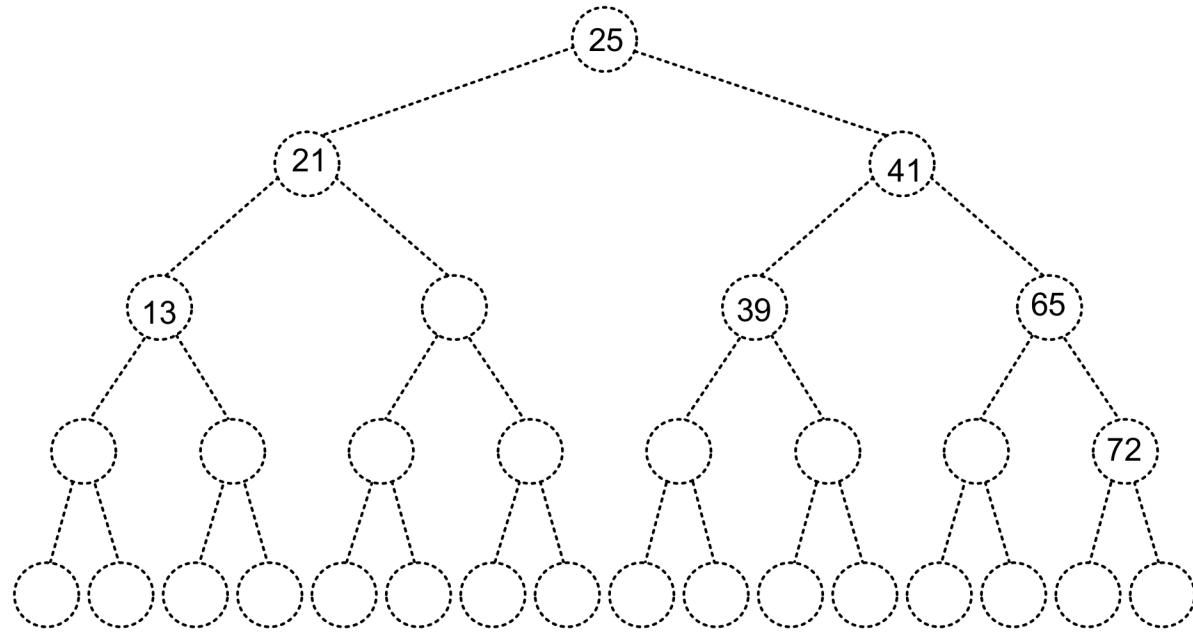
3,75/4,5

Question 2 : Parcours d'arbre**(1.5/20 points)**

2.1) **(0.25 point)** Si l'affichage pré-ordre d'un arbre binaire de recherche donne :

25, 21, 13, 41, 39, 65, 72

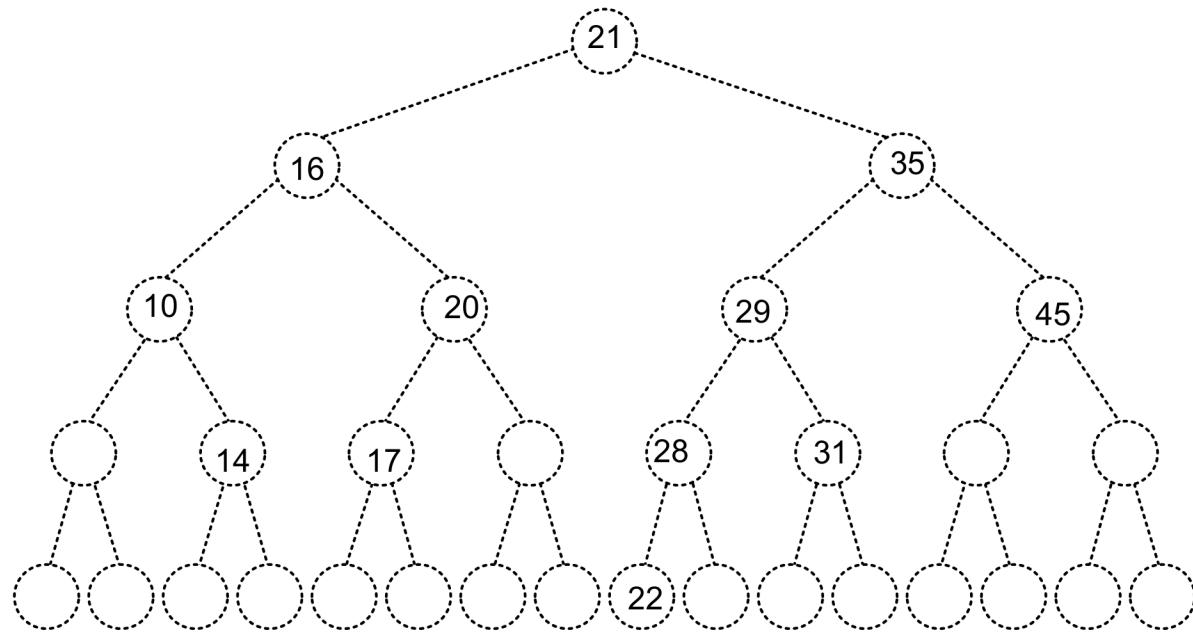
Donnez sa représentation graphique en complétant l'arbre ci-après :



2.2) **(0.25 point)** Si l'affichage post-ordre d'un arbre binaire de recherche donne :

14, 10, 17, 20, 16, 22, 28, 31, 29, 45, 35, 21

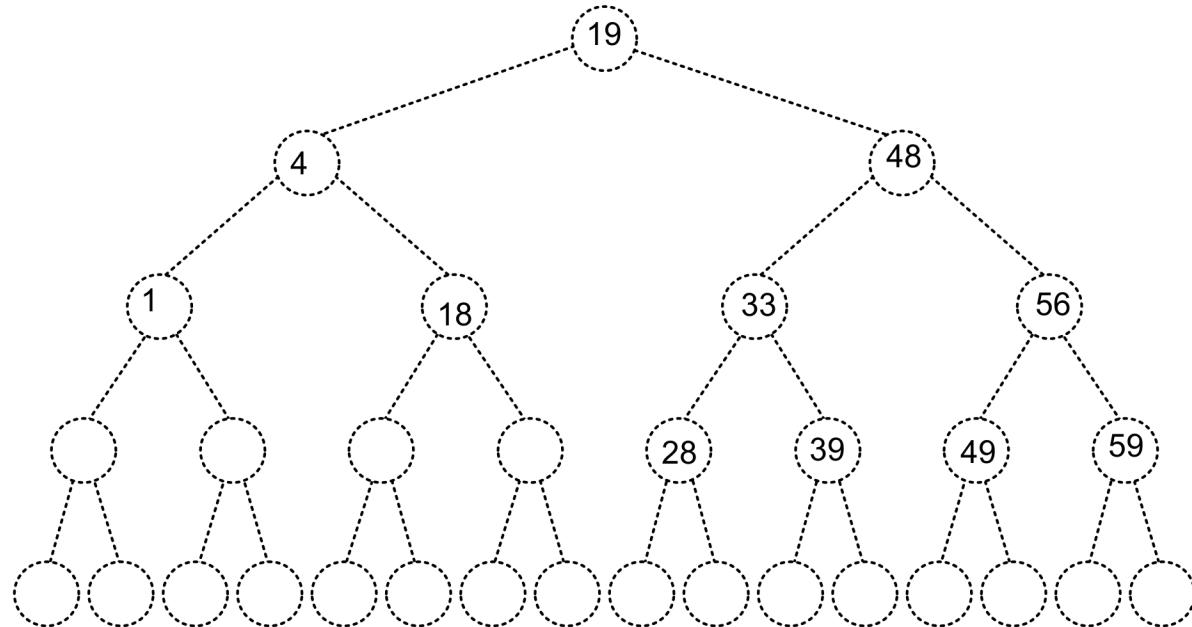
Donnez sa représentation graphique en complétant l'arbre ci-après :



2.3) (0.5 point) Si l'affichage en ordre d'un arbre AVL dont la racine est 19 donne :

1, 4, 18, 19, 28, 33, 39, 48, 49, 56, 59

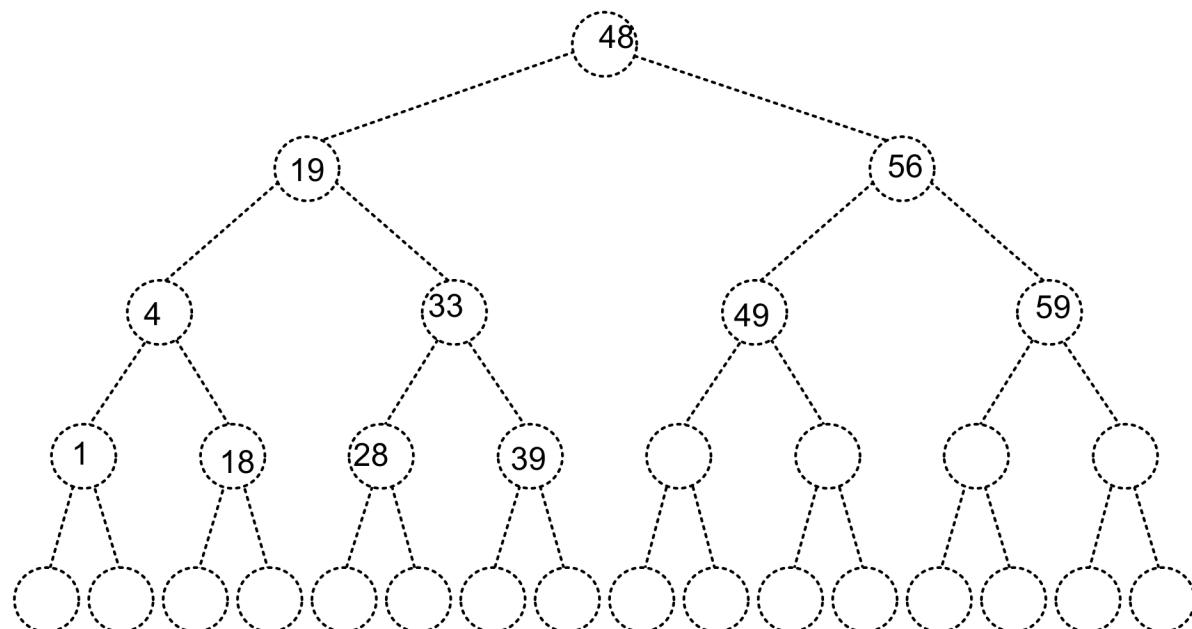
Donnez sa représentation graphique en complétant l'arbre ci-après :



2.4) (0.5 point) Si l'affichage en ordre d'un arbre binaire complet donne :

1, 4, 18, 19, 28, 33, 39, 48, 49, 56, 59

Donnez sa représentation graphique en complétant l'arbre ci-après :

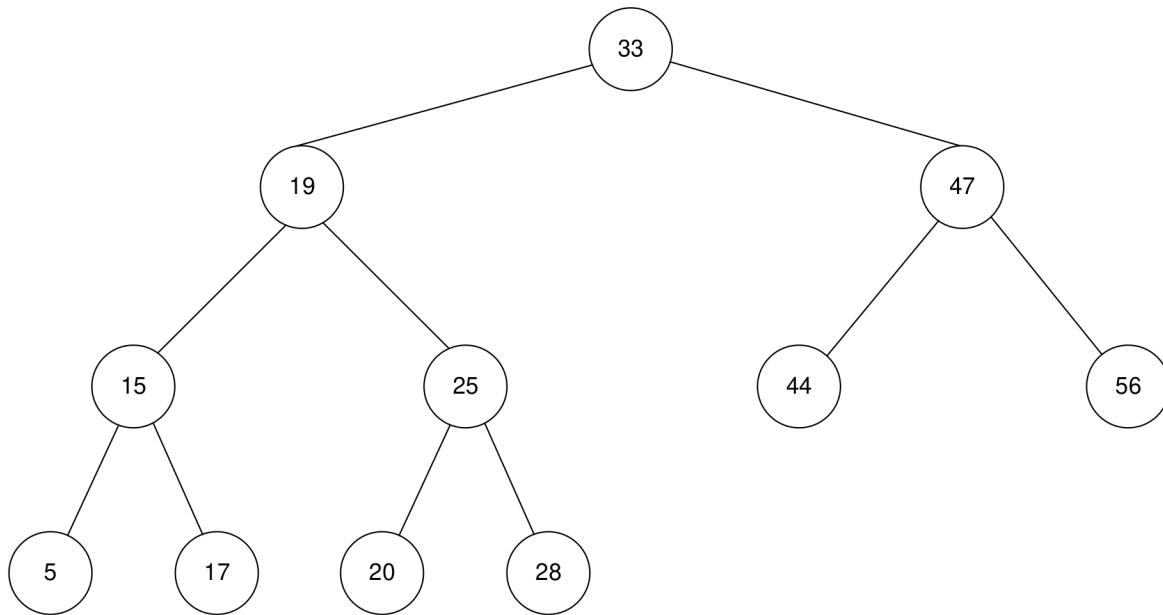


1,5/1,5

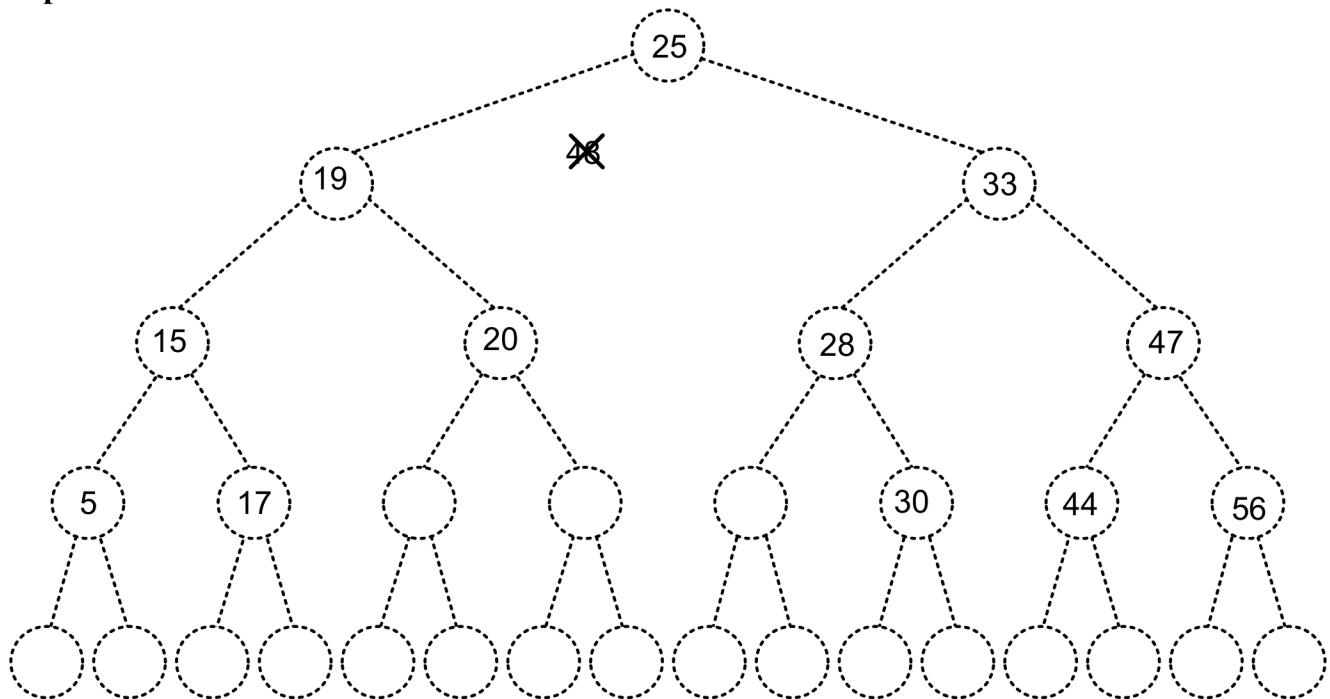
Question 3 : Opérations sur les arbres AVL**(4/20 points)**

En partant de chacun des arbres AVL suivants, effectuer les opérations demandées. Référez-vous au besoin à l'Annexe 1.

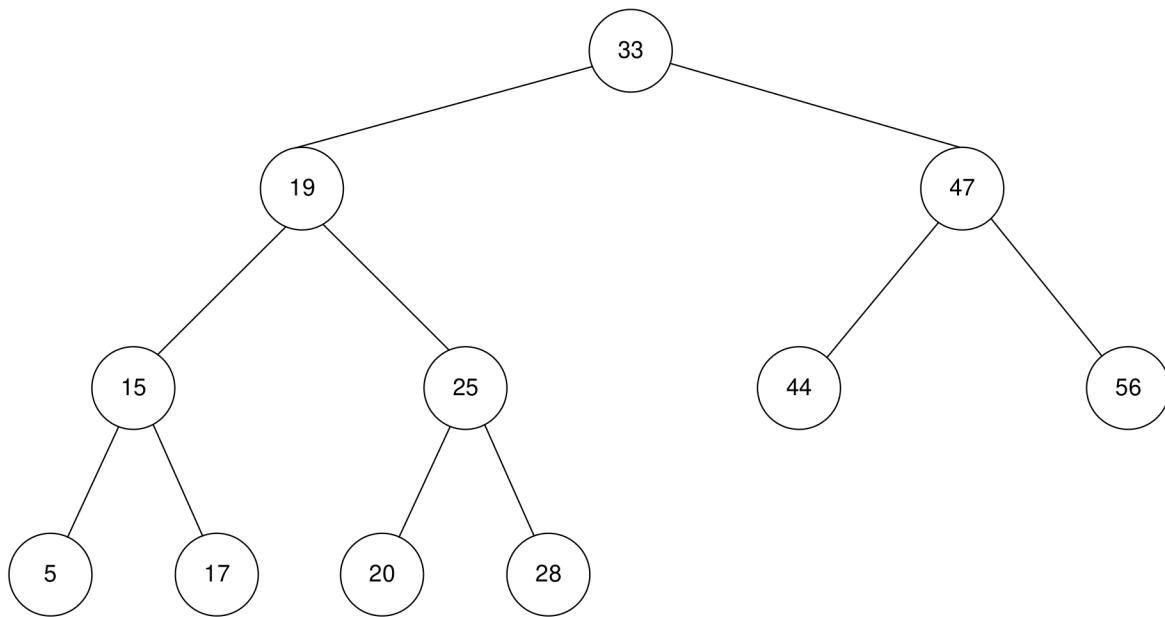
3.1) (1 point) Insérez 30.



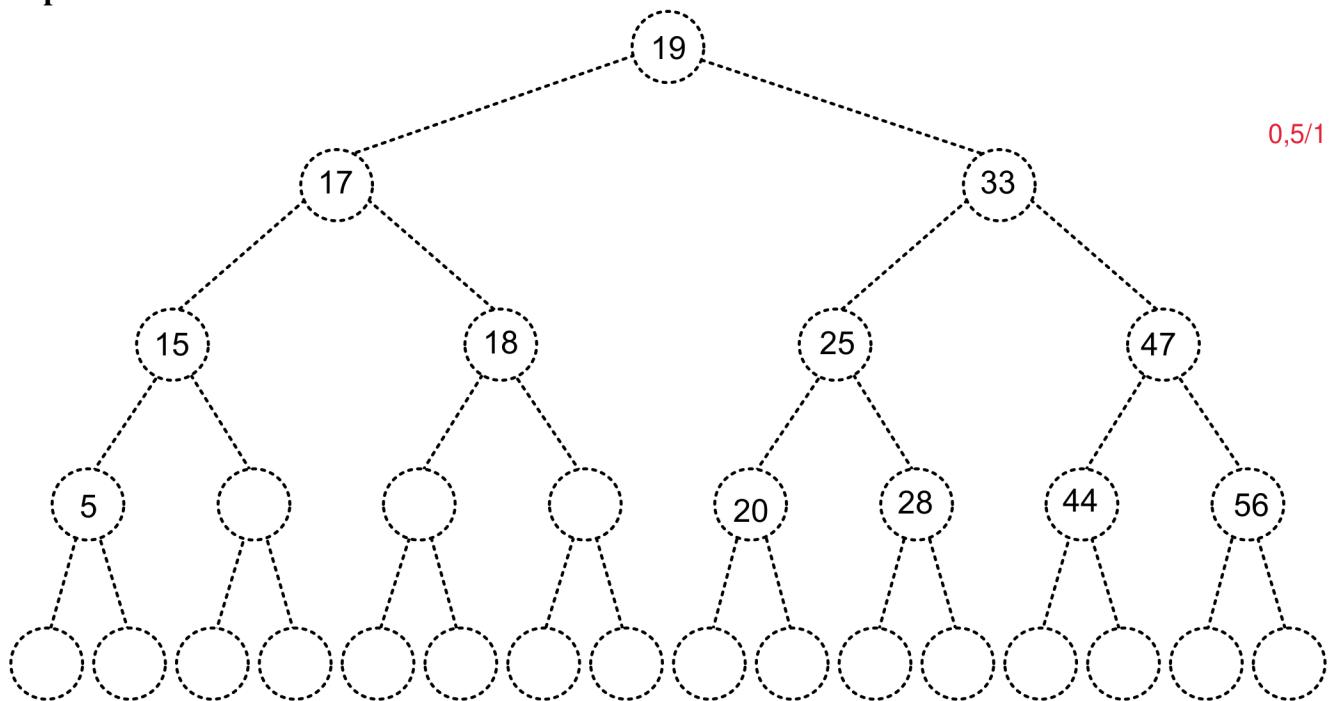
Réponse :



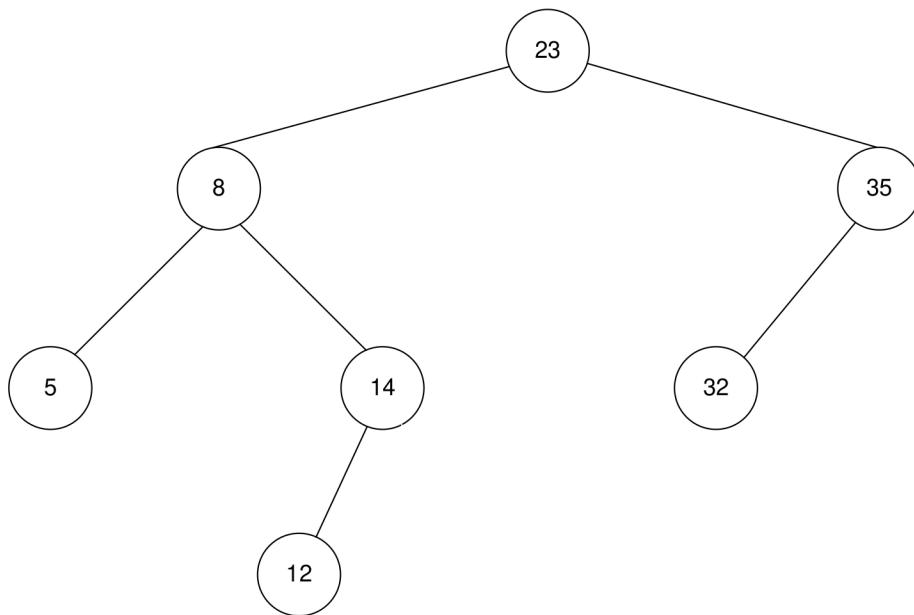
3.2) (1 point) Insérez 18.



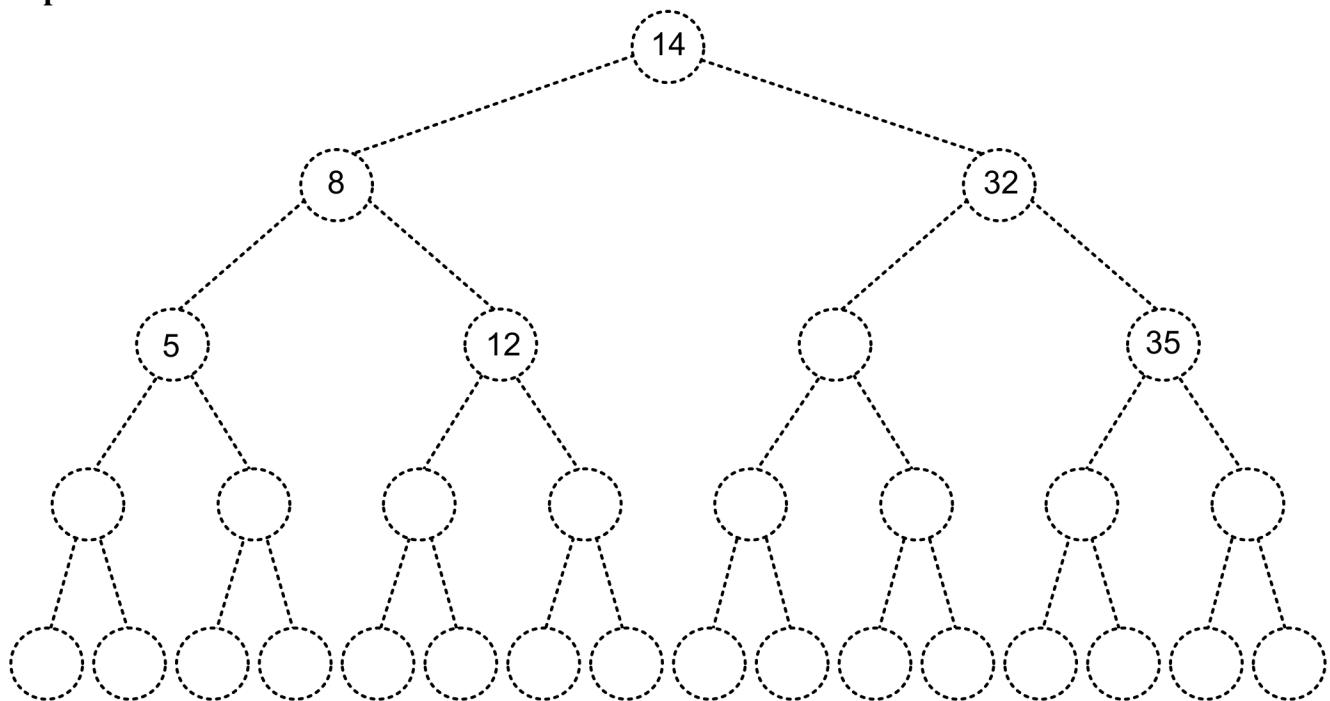
Réponse :



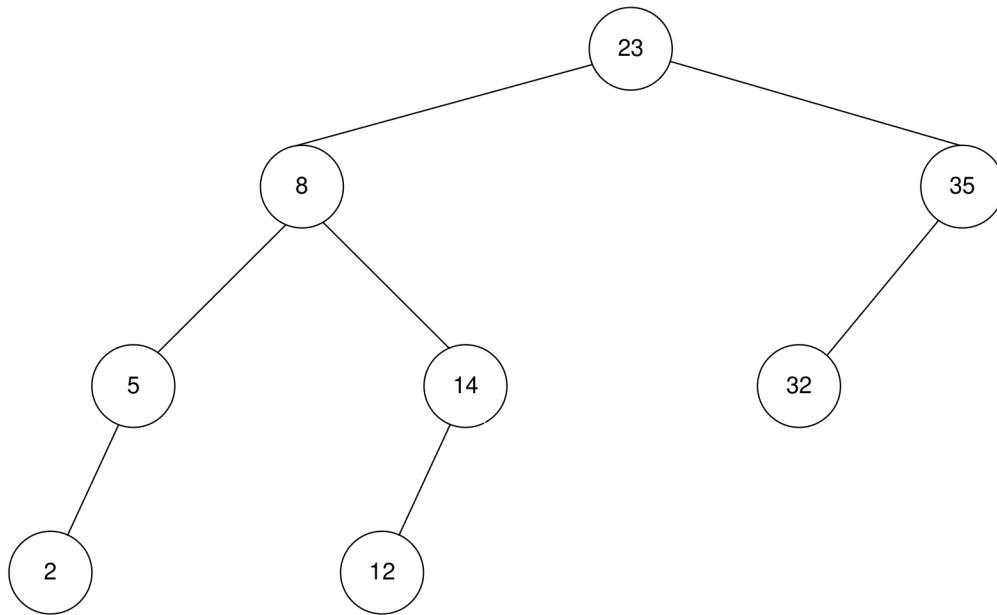
3.3) (1 point) Supprimez la racine.



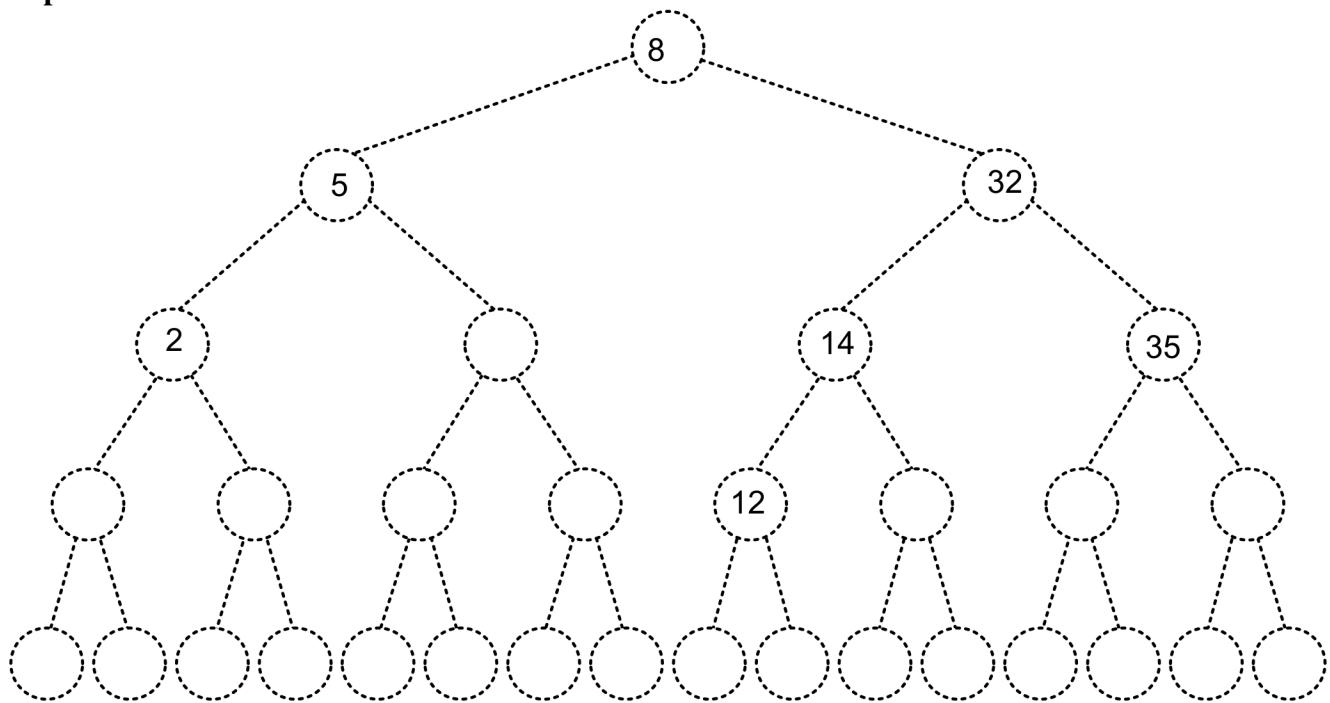
Réponse :



3.4) (1 point) Supprimez la racine. Référez-vous au code de l'Annexe 4 pour ce faire.



Réponse :



3,5/4

Question 4 : Graphes dirigés**(4/20 points)**

Pour chacun des graphes dirigés donnés ci-après, donnez un ordre topologique du graphe s'il en admet un, et identifiez ses composantes fortement connexes.

4.1) Soit le graphe dirigé suivant :

$$V = \{A, B, C, D, E, F, G, H, J, K\}$$

$$E = \{(A, B), (A, C), (B, H), (C, D), (C, E), (D, F), (E, A), (E, B), (E, F), (F, K), (G, B), (G, F), (H, J), (J, G), (J, K), (K, D)\}$$

4.1.1) (1 point) Admet-il un ordre topologique? non

Oui : _____ Non :

Si votre réponse est oui, proposez-en un (débutez la numérotation par 1) :

Nœud	A	B	C	D	E	F	G	H	J	K
Ordre :	—	—	—	—	—	—	—	—	—	—

Si votre réponse est non, justifiez votre réponse :

Il y a présence de cycle. Le graphe est orienté, mais pas acyclique, ce qui implique qu'il n'admet pas d'ordre topologique. J'ai encerclé 3 arêtes qui forment un cycle entre les sommet A, C et E.

4.1.2) (1 point) Identifiez les composantes fortement connexes du graphe dirigé en vous aidant de ce tableau. Chaque colonne représente une composante. Un sommet X est associé à une composante Y en noircissant la case correspondant à l'intersection de la ligne X et de la colonne Y. La numérotation des composantes débute à 1. On admettra que le sommet A appartient à la composante 1. Laissez les colonnes non utilisées vides.

Nœud	Composante									
	1	2	3	4	5	6	7	8	9	10
A	●									
B			●							
C	●									
D		●								
E	●									
F		●								
G				●						
H					●					
J						●				
K		●								

0,5

4.2) Soit le graphe dirigé suivant :

$$V = \{A, B, C, D, E, F, G, H, J, K\}$$

$$E = \{(A, C), (A, E), (B, A), (B, E), (B, G), (B, H), (C, D), (C, E), (D, F), (D, K), (E, D), (E, G), (F, E), (F, G), (F, K), (G, J), (J, H), (J, K)\}$$

4.2.1) (1 point) Admet-il un ordre topologique? non

Oui : _____ Non :

Si votre réponse est oui, proposez-en un (débutez la numérotation par 1) :

Nœud	A	B	C	D	E	F	G	H	J	K
Ordre :	—	—	—	—	—	—	—	—	—	—

Si votre réponse est non, justifiez votre réponse :

Il y a présence de cycle. Le graphe est orienté, mais pas acyclique, ce qui implique qu'il n'admet pas d'ordre topologique. J'ai encerclé 3 arêtes qui forment un cycle entre les sommet D, E et F.

4.2.2) (1 point) Identifiez les composantes fortement connexes du graphe dirigé en vous aidant de ce tableau. Chaque colonne représente une composante. Un sommet X est associé à une composante Y en noircissant la case correspondant à l'intersection de la ligne X et de la colonne Y. La numérotation des composantes débute à 1. On admettra que le sommet A appartient à la composante 1. Laissez les colonnes non utilisées vides.

Nœud	Composante									
	1	2	3	4	5	6	7	8	9	10
A	●									
B			●							
C				●						
D		●								
E		●								
F		●								
G					●					
H						●				
J							●			
K								●		

0,5

3/4

Question 5 : Automate de reconnaissance de motifs**(3/20 points)**

5.1) **(2 points)** En utilisant l'algorithme de construction des automates de reconnaissance de motifs, construisez l'automate capable de reconnaître la séquence suivante:

« abacaba »

Donnez l'automate ainsi obtenu dans la table suivante. Laissez les lignes inusitées vides.

Symboles États	a	b	c	d
0	1	0	0	0
1	1	2	0	0
2	3	0	0	0
3	1	2	4	0
4	5	0	0	0
5	1	6	0	0
6	7	0	0	0
7	1	2	4	0
8				
9				

État initial : 0

État final : 7

5.2) **(1 point)** Exécutez l'automate en reconnaissance sur le texte suivant:

« aabacabacabacabadacababadacabacabadabacaba »

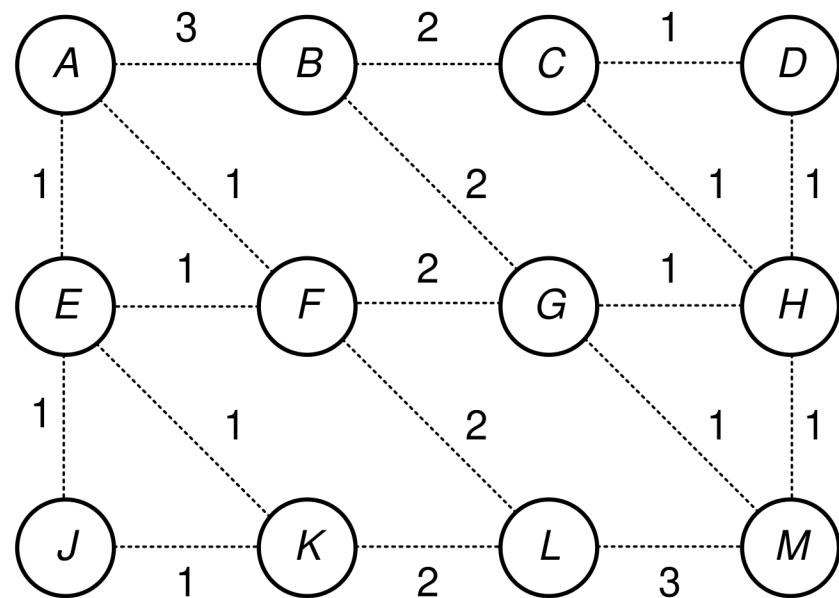
et donnez l'ensemble des décalages retournés par l'algorithme :

1, 5, 9, 25, 33 sont les indices des débuts séquence abacaba (le premier caractère est à l'indice 0). Il y a donc 5 fois la chaîne recherchée dans le texte.

3/3

Question 6 : Arbre sous-tendant minimum**(3/20 points)**

Soit le graphe suivant :



6.1) **(1,5 point)** Donnez l'arbre sous-tendant minimum obtenu par l'algorithme de Kruskal du graphe. Indiquez dans le tableau suivant les arêtes retenues. ✓ : oui ✗ : non

Arête	Coût	Retenue?
(A, B)	3	✗
(A, E)	1	✓
(A, F)	1	✓
(B, C)	2	✓
(B, G)	2	✗
(C, D)	1	✓
(C, H)	1	✓
(D, H)	1	✗
(E, F)	1	✗
(E, J)	1	✓
(E, K)	1	✓
(F, G)	2	✗
(F, L)	2	✓
(G, H)	1	✓
(G, M)	1	✓
(H, M)	1	✗
(J, K)	1	✗
(K, L)	2	✗
(L, M)	3	✗

1/1,5

6.2) Donnez le coût de l'arbre ainsi obtenu. Justifiez votre réponse par un calcul.

Le coût de l'arbre est la somme des valeurs des arêtes.

$$1+1+2+1+1+1+1+2+1+1 = 12$$

6.3) **(0.5 point)** Le graphe admet d'autres arbres sous-tendant minimum. Proposez en un second qui soit différent de celui donné en 6.1.

Arête	Coût	Retenue?
(A, B)	3	✗
(A, E)	1	✓
(A, F)	1	✓
(B, C)	2	✓
(B, G)	2	✗
(C, D)	1	✓
(C, H)	1	✓
(D, H)	1	✗
(E, F)	1	✗
(E, J)	1	✓
(E, K)	1	✓
(F, G)	2	✗
(F, L)	2	✗
(G, H)	1	✓
(G, M)	1	✓
(H, M)	1	✗
(J, K)	1	✗
(K, L)	2	✓
(L, M)	3	✗

6.4) **(1 point)** Combien d'arbres sous-tendant minimum différents le graphe admet-il au total ? Justifiez votre réponse par un calcul.

Beaucoup, plus de 10.

324 arbres

0/1

1,5/3

Annexe 1

```

public class AvlTree<AnyType extends Comparable<? super AnyType>> {

    public AvlTree( ) { root = null; }

    public void insert( AnyType x ) {
        root = insert( x, root );
    }
    public void remove( AnyType x ) {
        root = remove( x, root );
    }
    public boolean contains( AnyType x ) {
        return contains( x, root );
    }
    public void makeEmpty( ) {
        root = null;
    }
    public boolean isEmpty( ) {
        return root == null;
    }

    private static final int ALLOWED_IMBALANCE = 1;

    // Assume t is either balanced or within one of being balanced
    private AvlNode<AnyType> balance( AvlNode<AnyType> t ) {
        if( t == null )
            return t;

        if( height( t.left ) - height( t.right ) > ALLOWED_IMBALANCE )
            if( height( t.left.left ) >= height( t.left.right ) )
                t = rotateWithLeftChild( t );
            else
                t = doubleWithLeftChild( t );
        else
            if( height( t.right ) - height( t.left ) > ALLOWED_IMBALANCE )
                if( height( t.right.right ) >= height( t.right.left ) )
                    t = rotateWithRightChild( t );
                else
                    t = doubleWithRightChild( t );
        t.height = Math.max( height( t.left ), height( t.right ) ) + 1;
        return t;
    }

    private boolean checkBalance( AvlNode<AnyType> t ) {
        if( t == null )
            return true;
        if( Math.abs( height( t.left ) - height( t.right ) ) > 1 )
            return false;

        boolean lb = checkBalance( t.left );
        boolean rb = checkBalance( t.right );

        return lb && rb;
    }
}

```

```

private AvlNode<AnyType> insert( AnyType x, AvlNode<AnyType> t ) {
    if( t == null )
        return new AvlNode<>( x, null, null );

    int compareResult = x.compareTo( t.element );
    if( compareResult < 0 )
        t.left = insert( x, t.left );
    else if( compareResult > 0 )
        t.right = insert( x, t.right );
    else
        ; // Duplicate; do nothing
    return balance( t );
}

private AvlNode<AnyType> remove( AnyType x, AvlNode<AnyType> t ) {
    if( t == null )
        return t; // Item not found; do nothing

    int compareResult = x.compareTo( t.element );

    if( compareResult < 0 )
        t.left = remove( x, t.left );
    else if( compareResult > 0 )
        t.right = remove( x, t.right );
    else if( t.left != null && t.right != null ) { // Two children
        t.element = findMin( t.right ).element;
        t.right = remove( t.element, t.right );
    }
    else
        t = ( t.left != null ) ? t.left : t.right;
    return balance( t );
}

private AvlNode<AnyType> findMin( AvlNode<AnyType> t ) {
    if( t == null )
        return t;

    while( t.left != null )
        t = t.left;
    return t;
}

private boolean contains( AnyType x, AvlNode<AnyType> t ) {
    while( t != null ) {
        int compareResult = x.compareTo( t.element );
        if( compareResult < 0 )
            t = t.left;
        else if( compareResult > 0 )
            t = t.right;
        else
            return true; // Match
    }

    return false; // No match
}

```

```
private int height( AvlNode<AnyType> t ) {
    return t == null ? -1 : t.height;
}

private AvlNode<AnyType> rotateWithLeftChild( AvlNode<AnyType> k2 ) {
    AvlNode<AnyType> k1 = k2.left;
    k2.left = k1.right;
    k1.right = k2;
    k2.height = Math.max( height( k2.left ), height( k2.right ) ) + 1;
    k1.height = Math.max( height( k1.left ), k2.height ) + 1;
    return k1;
}

private AvlNode<AnyType> rotateWithRightChild( AvlNode<AnyType> k1 ) {
    AvlNode<AnyType> k2 = k1.right;
    k1.right = k2.left;
    k2.left = k1;
    k1.height = Math.max( height( k1.left ), height( k1.right ) ) + 1;
    k2.height = Math.max( height( k2.right ), k1.height ) + 1;
    return k2;
}

private AvlNode<AnyType> doubleWithLeftChild( AvlNode<AnyType> k3 ) {
    k3.left = rotateWithRightChild( k3.left );
    return rotateWithLeftChild( k3 );
}

private AvlNode<AnyType> doubleWithRightChild( AvlNode<AnyType> k1 ) {
    k1.right = rotateWithLeftChild( k1.right );
    return rotateWithRightChild( k1 );
}

private static class AvlNode<AnyType>{

    AvlNode( AnyType theElement, AvlNode<AnyType> lt, AvlNode<AnyType> rt ){
        element = theElement;
        left = lt;
        right = rt;
        height = 0;
    }

    AnyType element; // The data in the node
    AvlNode<AnyType> left; // Left child
    AvlNode<AnyType> right; // Right child
    int height; // Height
}

/** The tree root. */
private AvlNode<AnyType> root;
}
```

Page supplémentaire :