

Questionnaire examen intra

INF2010

Sigle du cours

Identification de l'étudiant(e)		
Nom :	Prénom :	
Signature :	Matricule :	Groupe :

Sigle et titre du cours		Groupe	Trimestre
INF2010 – Structures de données et algorithmes		Tous	20101
Professeur		Local	Téléphone
Ettore Merlo, responsable – Tarek Ould Bachir, chargé de cours		B-505/B-506	7128
Jour	Date	Durée	Heure
Mercredi	17 février 2009	2h00	19h00
Documentation		Calculatrice	
<input type="checkbox"/> Toute <input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Voir directives particulières		<input type="checkbox"/> Aucune <input type="checkbox"/> Programmable <input checked="" type="checkbox"/> Non programmable Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.	

Directives particulières
<div></div>

Bonne chance à tous!

Important	Cet examen contient 5 questions sur un total de 12 pages (excluant cette page)
	La pondération de cet examen est de 30 %
	Vous devez répondre sur : <input checked="" type="checkbox"/> le questionnaire <input type="checkbox"/> le cahier <input type="checkbox"/> les deux
	Vous devez remettre le questionnaire : <input checked="" type="checkbox"/> oui <input type="checkbox"/> non

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

Question 1 : Tables de dispersion**(16 points)**

Considérant une table de dispersion par débordement progressif avec sondage linéaire où $\text{Hash}(\text{clé}) = \text{clé} \% N + 2 \cdot i$:

1.1) **(2 point)** Donnez la complexité asymptotique en cas moyen de l'insertion pour cette structure de données :

1.2) **(10 points)** En vous servant du tableau ci-dessous, donnez l'état de la mémoire d'une table de dispersion par débordement progressif avec sondage linéaire ($\text{Hash}(\text{clé}) = \text{clé} \% N + 2 \cdot i$ et $N = 7$) dans laquelle on insère dans l'ordre les paires (objet, clé) suivantes : (patate, 7), (tomate, 7), (chou-rave, 7), (brocoli, 7), (vache, 7).

0	
1	
2	
3	
4	
5	
6	

1.3) **(4 points)** Quelle est la complexité asymptotique en cas moyen de l'insertion pour cette structure de données dans le cas où toutes les entrées ont la même clé. Justifiez votre réponse.

Question 2 : Complexité algorithmique**(12 points)**

Considérez le bout de code suivant et la fonction `quelleComplexite` qui y est présentée :

```
public class complexite
{
    public static void quelleComplexite(int[] a)
    {
        int i, j, k = 1, n = a.length;
        for(i = 1; i <= n; ++i)
        {
            for(j=1; j*k <= n; ++j)
                a[j-1] += i;

            k = 2*k;
        }
    }

    public static void main(String[] args)
    {
        int[] a = {0, 0, 0, 0, 0, 0, 0, 0, 0};
        quelleComplexite( a );
        for(int i : a)
        {
            System.out.print( i + " " );
        }
    }
}
```

2.1) **(4 points)** Donnez l’affichage produit par l’exécution de la fonction principale (`main`) :

2.2) **(8 points)** Quelle est la complexité asymptotique $O(f(n))$ de `quelleComplexite(int[] a)`, fonction de la taille n du tableau `int[] a`. Justifiez votre réponse par un calcul.

Rappel : $\lim_{n \rightarrow \infty} \sum_{i=1}^n \frac{1}{2^i} = 1$

Question 3 : Tris en $n \log(n)$ **(24 points)**

Exécuter l'algorithme « QuickSort » donné à l'Annexe I pour trier le vecteur suivant. La valeur *cut-off* pour l'algorithme « QuickSort » est 10.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vecteur	10	85	23	29	32	56	54	13	49	31	12	33	51	47	9	18

Réponses :

3.1) **(2 points)** Donnez :

Les trois valeurs de « Median3 » à la première récursion :

La valeur de la médiane (pivot) :

3.2) **(2 points)** Donnez l'état du vecteur après l'exécution de Median 3 de la première récursion :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vecteur																

3.3) **(6 points)** Donnez l'état du vecteur après l'exécution du partitionnement de la première récursion :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vecteur																

3.4) **(3 points)** La fonction récursive QuickSort sera-telle appelée après cette première récursion? Justifiez brièvement.

3.5) **(3 points)** Si la fonction récursive QuickSort devait entrer dans son second niveau de récursion, il n'y aurait au total qu'un seul des deux appels récursifs QuickSort qui lancerait de nouveau Median 3 et le code de partitionnement. Expliquez pourquoi.

3.6) **(2 points)** Median 3 est effectivement appelée une seconde fois dans notre exemple. Donnez :

Les trois valeurs de « Median3 » à cette seconde récursion :

La valeur de la médiane (pivot) :

3.7) **(6 points)** Donnez l'état du vecteur après l'exécution de Median 3 à cette seconde récursion (faites bien attention à l'ordre des appels de récursion! Regardez le code à l'annexe 1 au besoin) :

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vecteur																

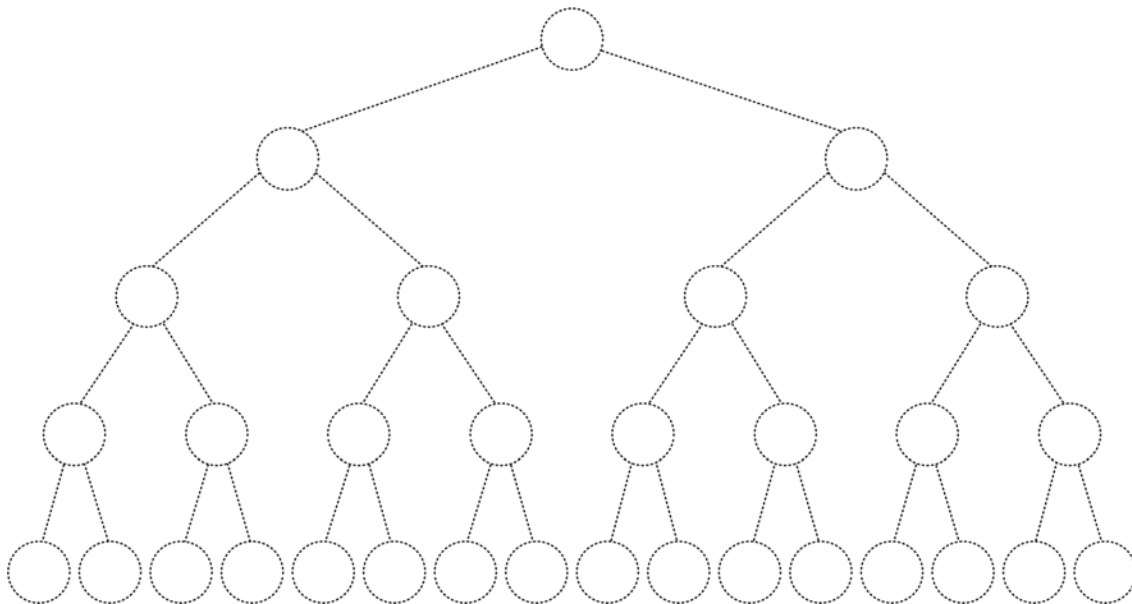
Question 4 : Arbres binaire de recherche**(24 points)**

Considérez les affichages des arbres binaires de recherche suivants. Pour chacun d'eux, donnez la représentation graphique de l'arbre. Dans chaque cas, dites si l'arbre binaire est un AVL.

4.1) **(4 points)** L'affichage pré-ordre de l'arbre binaire de recherche donne :

53, 24, 10, 35, 42, 78, 65, 85, 81, 90, 96

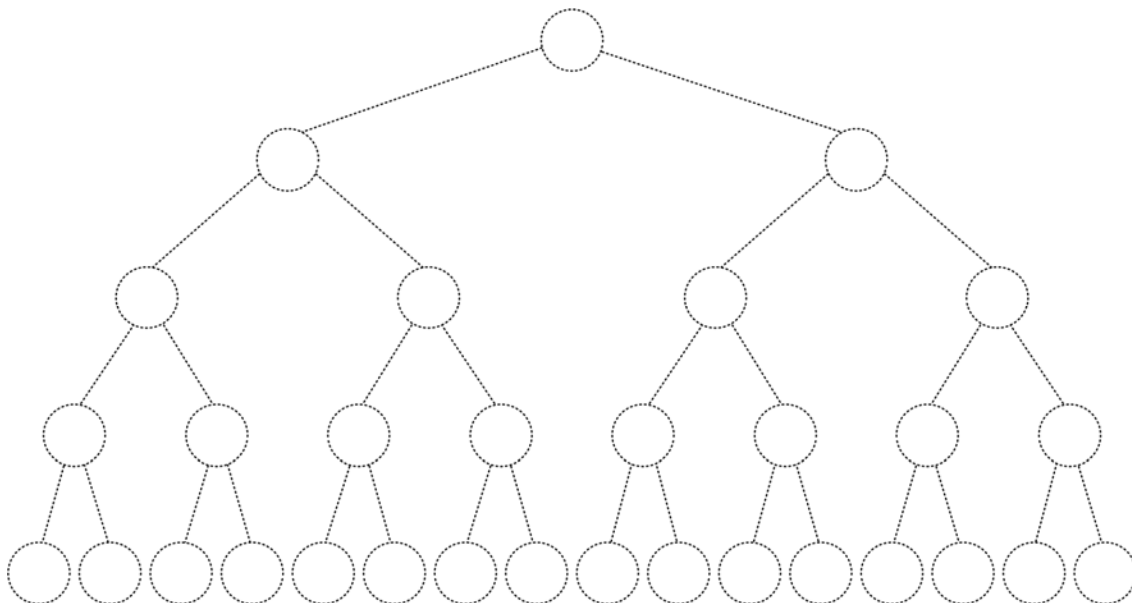
AVL? : _____



4.2) **(4 points)** L'affichage post-ordre de l'arbre binaire de recherche donne :

24, 12, 45, 31, 52, 54, 59, 71, 78, 66, 49

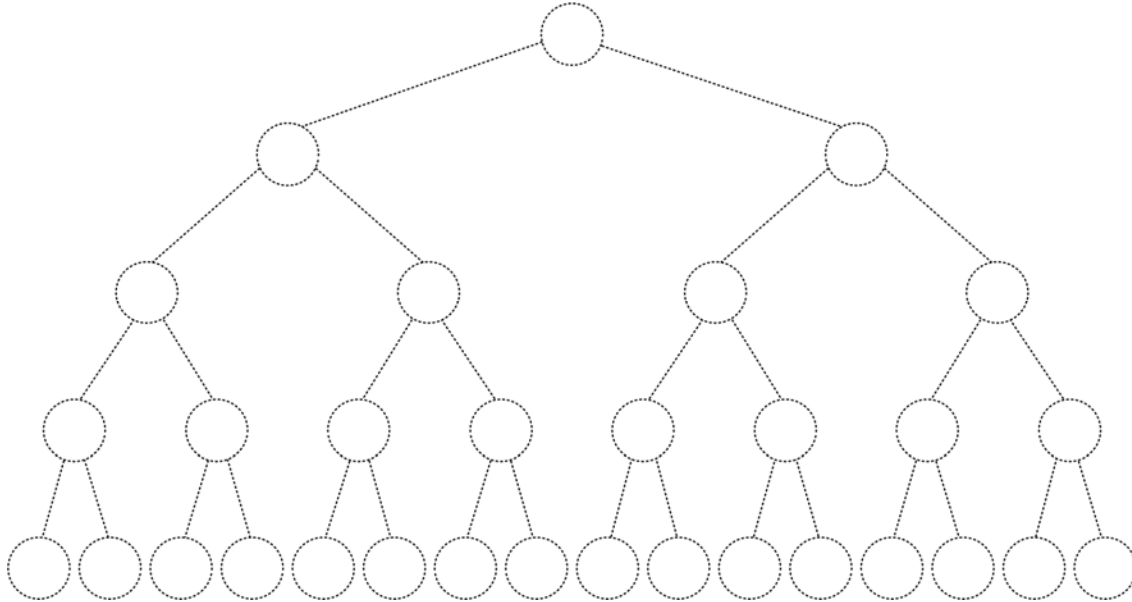
AVL? : _____



4.3) (4 points) L'affichage post-ordre de l'arbre binaire de recherche donne :

24, 10, 42, 35, 31, 54, 59, 71, 96, 90, 85, 66, 53

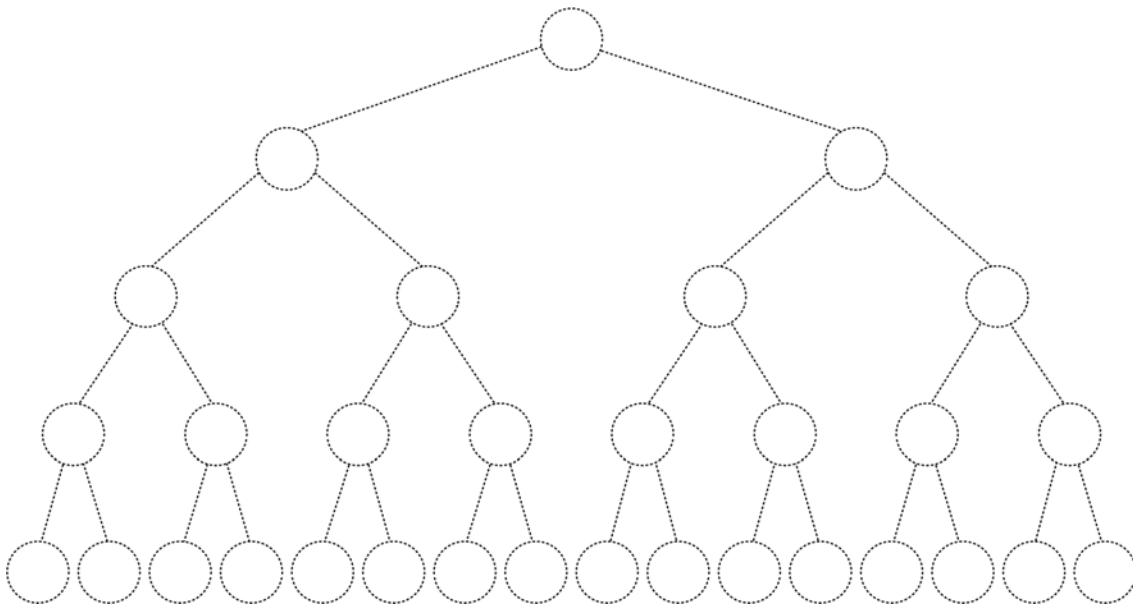
AVL? : _____



4.4) (4 points) L'affichage par niveau de l'arbre binaire de recherche donne :

49, 32, 66, 10, 59, 70, 52, 68, 72, 71, 76

AVL? : _____

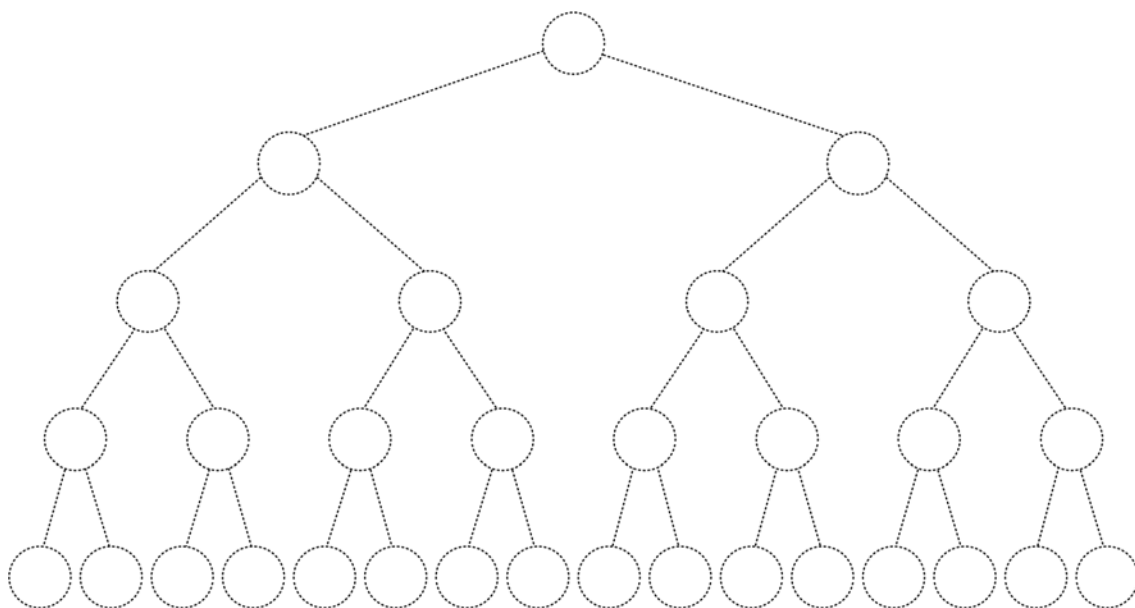


4.5) (8 points) L'affichage en ordre d'un arbre binaire de recherche ne permet pas d'en déduire la constitution. Néanmoins, sachant que :

- Le sous-arbre de gauche de chaque nœud de l'arbre binaire de recherche que nous considérons ici, est plus haut que le sous-arbre de droite;
- L'arbre binaire de recherche considéré est un AVL;
- Que la fonction $S(h)$ donnant le nombre minimal contenus dans un AVL respecte la relation $S(h) = S(h-1) + S(h-2) + 1$; $S(0) = 1$, $S(1) = 2$;

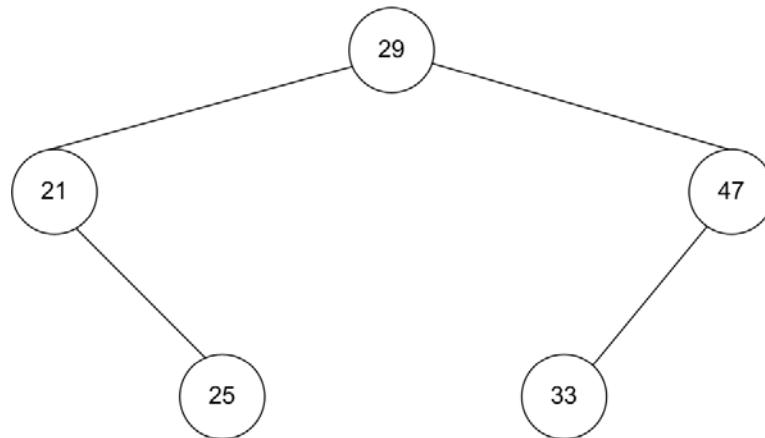
Donnez la représentation graphique de notre arbre AVL dont l'affiche en ordre est :

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12



Question 5 : Arbre binaire de recherche de type AVL**(24 points)**

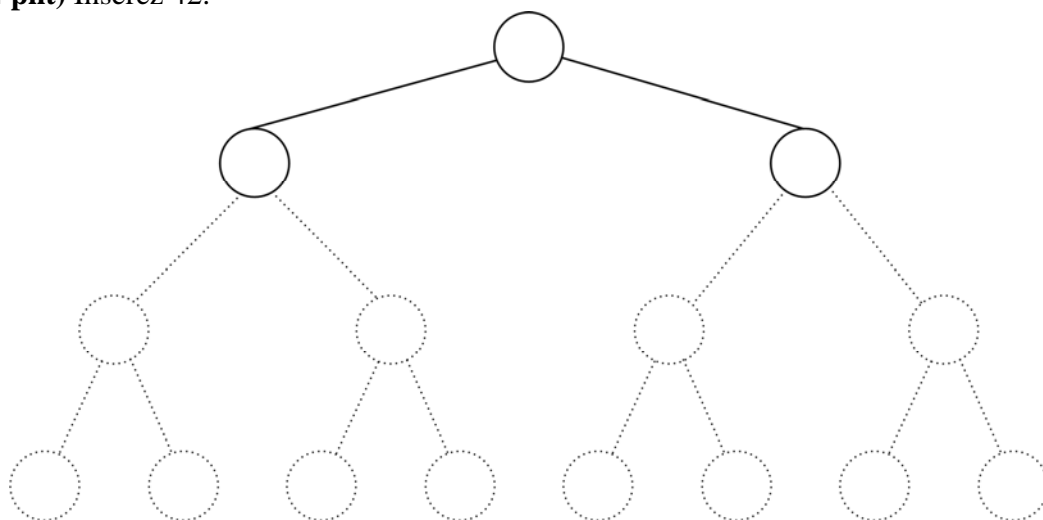
En considérant l'arbre AVL suivant :



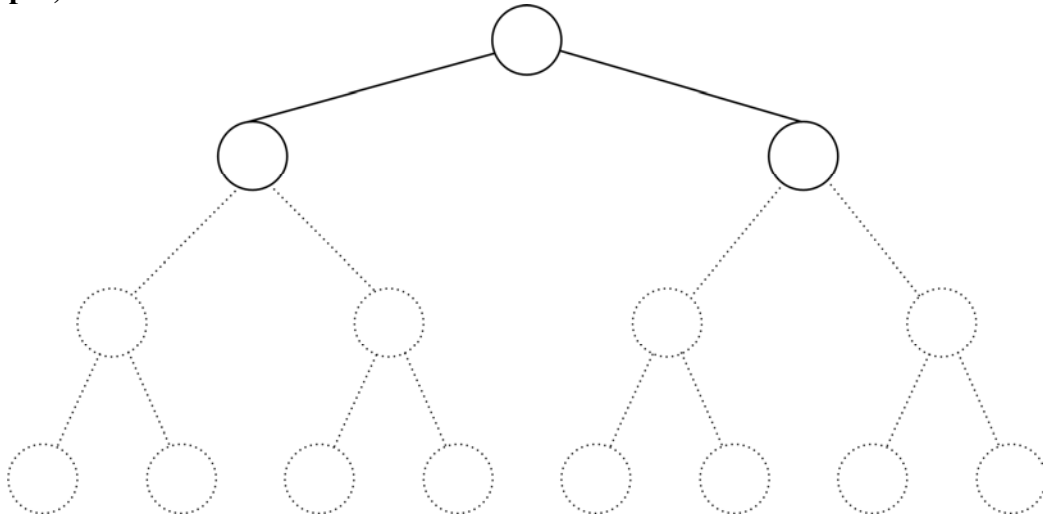
Effectuez l'ensemble des opérations suivantes dans l'ordre en vous servant des arbres ci-dessous :

Insérez 42
Insérez 23
Insérez 51
Insérez 45
Insérez 60
Insérez 30

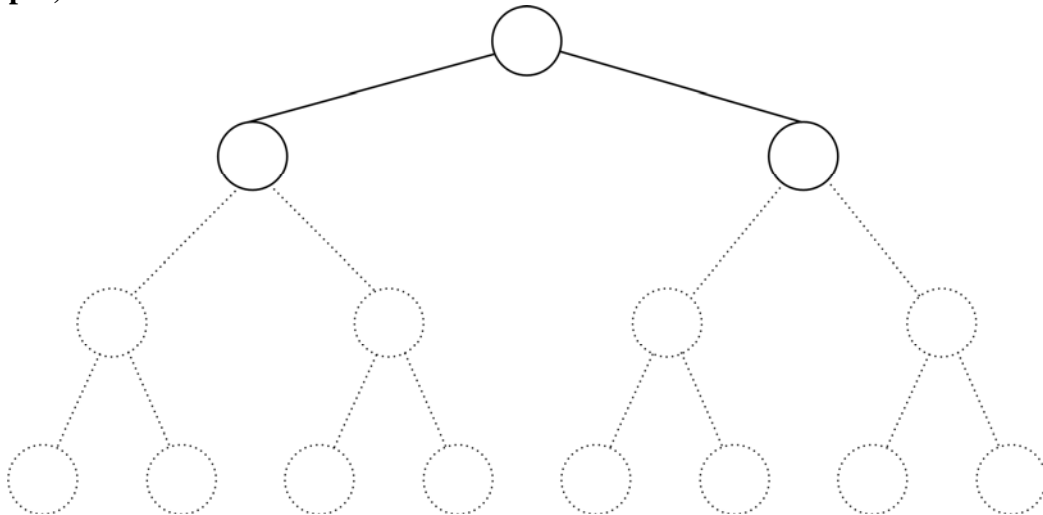
5.1) (4 pnt) Insérez 42.



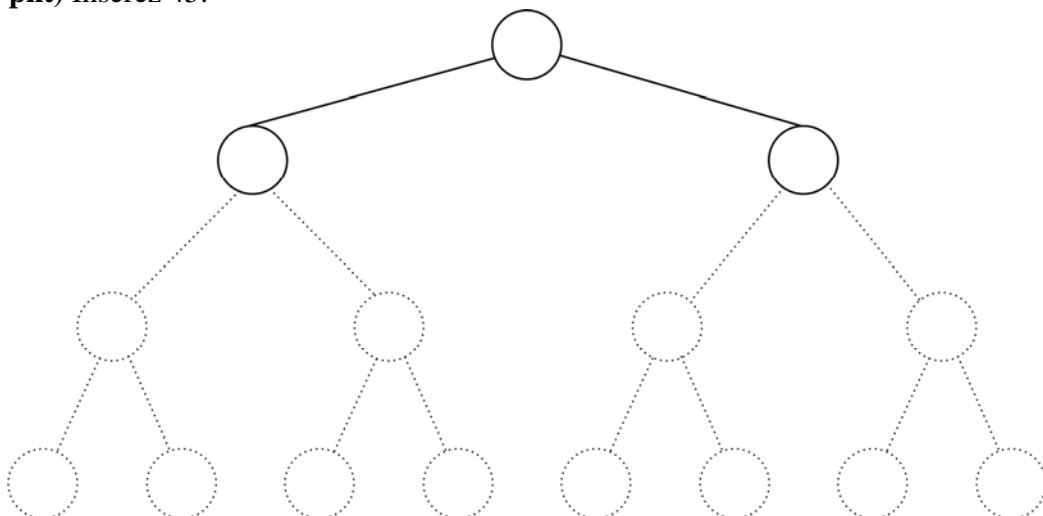
5.2) (4 pnt) Insérez 23.



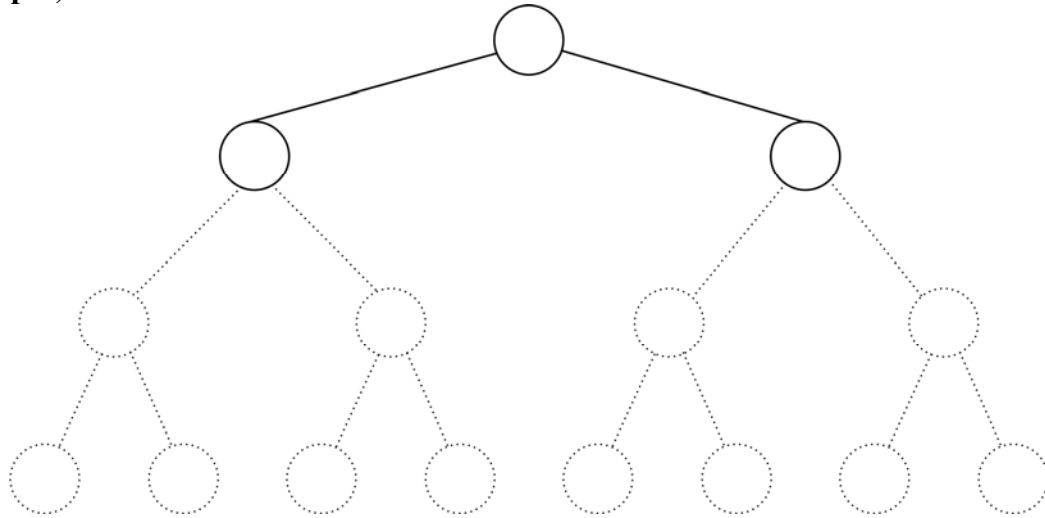
5.3) (4 pnt) Insérez 51.



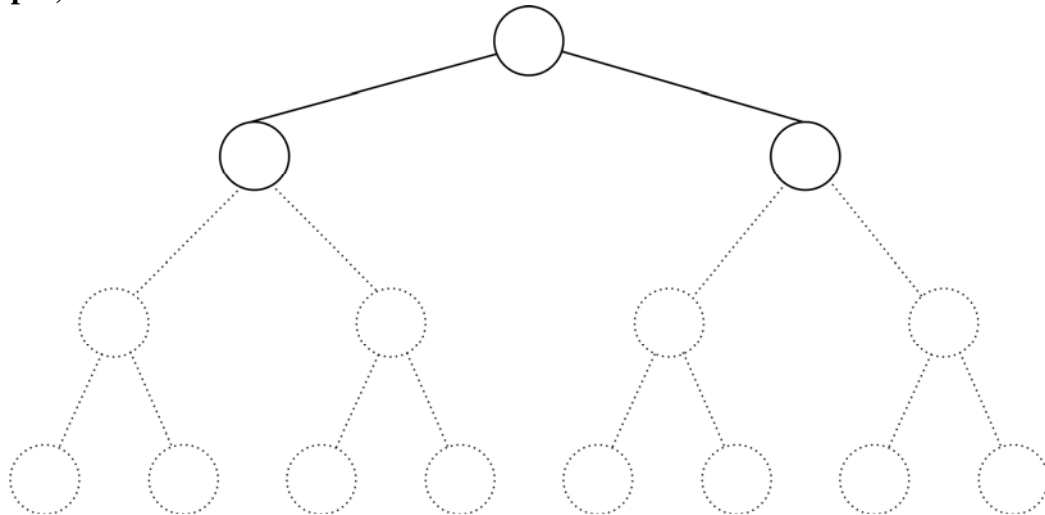
5.4) (4 pnt) Insérez 45.



5.5) (4 pnt) Insérez 60.



5.6) (4 pnt) Insérez 30.



Annexe 1

```
public final class SortIntra
{
    private static final int CUTOFF = 10;

    /**
     * Quicksort
     */
    public static <AnyType extends Comparable<? super AnyType>>
    void quicksort( AnyType [ ] a )
    {
        quicksort( a, 0, a.length - 1 );
    }

    /**
     * Appel interne à quicksort
     * Utilise Median 3 et une valeur limite (cutoff) de 10.
     */
    private static <AnyType extends Comparable<? super AnyType>>
    void quicksort( AnyType [ ] a, int left, int right )
    {
        if( left + CUTOFF <= right )
        {
            AnyType pivot = median3( a, left, right );

            // partitionnement
            int i = left, j = right - 1;
            for( ; ; )
            {
                while( a[ ++i ].compareTo( pivot ) < 0 ) { }
                while( a[ --j ].compareTo( pivot ) > 0 ) { }
                if( i < j )
                    swapReferences( a, i, j );
                else
                    break;
            }

            swapReferences( a, i, right - 1 );
            // fin du partitionnement

            // recursion
            quicksort( a, left, i - 1 );
            quicksort( a, i + 1, right );
        }
        else
            insertionSort( a, left, right );
    }
}
```

```

/**
 * Interchange (swap) deux valeurs
 */
public static <AnyType> void
swapReferences( AnyType [ ] a, int index1, int index2 )
{
    AnyType tmp = a[ index1 ];
    a[ index1 ] = a[ index2 ];
    a[ index2 ] = tmp;
}

/**
 * Median 3
 */
private static <AnyType extends Comparable<? super AnyType>>
AnyType median3( AnyType [ ] a, int left, int right )
{
    int center = ( left + right ) / 2;
    if( a[ center ].compareTo( a[ left ] ) < 0 )
        swapReferences( a, left, center );
    if( a[ right ].compareTo( a[ left ] ) < 0 )
        swapReferences( a, left, right );
    if( a[ right ].compareTo( a[ center ] ) < 0 )
        swapReferences( a, center, right );

    swapReferences( a, center, right - 1 );
    return a[ right - 1 ];
}

/**
 * insertionSort interne.
 * Utilisé par by quicksort.
 */
private static <AnyType extends Comparable<? super AnyType>>
void insertionSort( AnyType [ ] a, int left, int right )
{
    for( int p = left + 1; p <= right; p++ )
    {
        AnyType tmp = a[ p ]; int j;

        for( j = p; j > left && tmp.compareTo( a[ j - 1 ] ) < 0; j-- )
            a[ j ] = a[ j - 1 ];
        a[ j ] = tmp;
    }
}

public static void main( String [ ] args )
{
    Integer [ ] a = {10, 85, 23, 29, 32, 56, 54, 13,
                     49, 31, 12, 33, 51, 47, 9, 18};

    quicksort( a );

    for(Integer valeur : a) System.out.print(valeur + " ");
}
}

```