Commencé le	lundi 24 avril 2023, 13:35
État	Terminé
Terminé le	lundi 24 avril 2023, 14:35
Temps mis	1 heure
En retard	59 min
Points	20,50/20,50
Note	**20,00** sur 20,00 (**100**%)

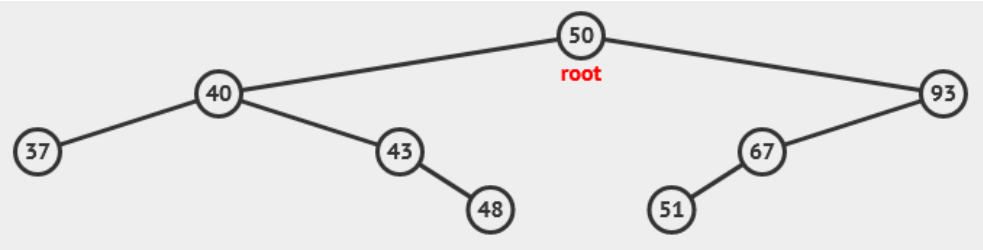
Question 1

Correct

Note de 1,00 sur 1,00

Donner la liste des sommets dans le **parcours pré-ordre** du Binary Search Tree ci-dessous :

List the nodes in the BST **pre-order path** below:



✔ 50 ✔ 40 ✔ 37 ✔ 43 ✔ 48 ✔ 93 ✔ 67 ✔ 51

Votre réponse est correcte.

Question 2

Correct

Note de 1,50 sur 1,50

Pour trouver le nœud 62 dans un BST ***B***, l'ensemble des sommets traversés est :
56, 60, 62, 68, 70 et 85 (ces sommets sont donnés dans l'ordre croissant qui n'est pas l'ordre de traversée des sommets pour trouver 62).

Parmi ces nœuds, quels sont ceux qui peuvent être racine de ***B***.

To find node 62 in a BST B, the set of traversed vertices is :

56, 60, 62, 68, 70 and 85 (these nodes are given in ascending order which is not the order of traversal of the vertices to find 62).

Among these nodes, which ones can be a root of B?

Veuillez choisir au moins une réponse.

- [x] 85 ✔
- [ ] 60
- [ ] 68
- [ ] 62
- [ ] 70
- [x] 56 ✔

Votre réponse est correcte.

56, 85, 70, 68, 56, 60, 62

Si on place **85** en racine on part à gauche sur 56 puis à droite sur 70 etc.

Si on place 70 on ne peut pas passer dessous par 85 et 56 pour chercher 62 (à gauche forcément et donc plus petit que 70)

etc.

Les réponses correctes sont :
56,

85

Question 3

Correct

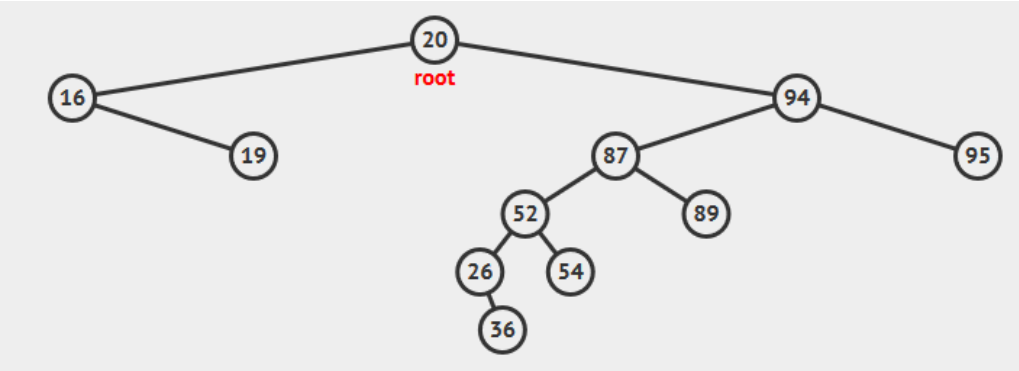
Note de 1,00 sur 1,00

Quelle est la valeur de l'élément de ***rang 8*** dans le BST ci-dessous ?

Le ***rang*** d'un élément est sa place dans la liste ordonnée par ordre croissant des valeurs du BST, le plus petit élément a le rang 1.

What is the value of the element whose rank is 8 in the BST below?

The rank of an element is its place in the ordered list of BST values; the smallest element has rank 1.



Réponse : 87

La réponse correcte est : 87

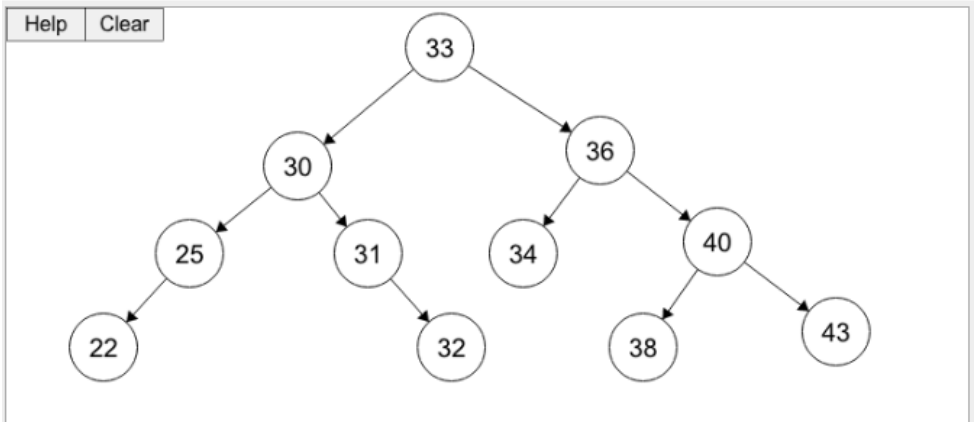
Question 4

Correct

Note de 1,00 sur 1,00

L'arbre binaire ci-dessous est-il un AVL ?

Is the binary tree below an AVL?



Veuillez choisir une réponse.

- Vrai ✔
- Faux

Votre réponse est correcte.

La réponse correcte est : Vrai

Question 5

Correct

Note de 2,00 sur 2,00

Dans tout tas (min-heap) de taille n :

In any heap (min-heap) of size n :

Veuillez choisir au moins une réponse.

- [ ] Le nombre de feuilles est inférieur ou égal à n/2
  The number of leaves is less than or equal to n/2
- [x] On doit regarder toutes les feuilles pour trouver le maximum ✔
  We must look at all the leaves to find the maximum
- [ ] Les 2 plus grandes valeurs sont nécessairement sur des feuilles
  The 2 largest values are necessarily on leaves
- [x] La plus grande valeur est forcément sur une feuille ✔
  The largest value is necessarily on a leaf
- [ ] Tout nœud non feuille a nécessairement 2 sous-arbres non-vides
  Any non-leaf node necessarily has 2 non-empty subtrees
- [x] Le nombre de feuilles est supérieur ou égal à n/2 ✔
  The number of leaves is greater than or equal to n/2

Votre réponse est correcte.

Les réponses correctes sont : La plus grande valeur est forcément sur une feuille
The largest value is necessarily on a leaf, On doit regarder toutes les feuilles pour trouver le maximum
We must look at all the leaves to find the maximum, Le nombre de feuilles est supérieur ou égal à n/2
The number of leaves is greater than or equal to n/2

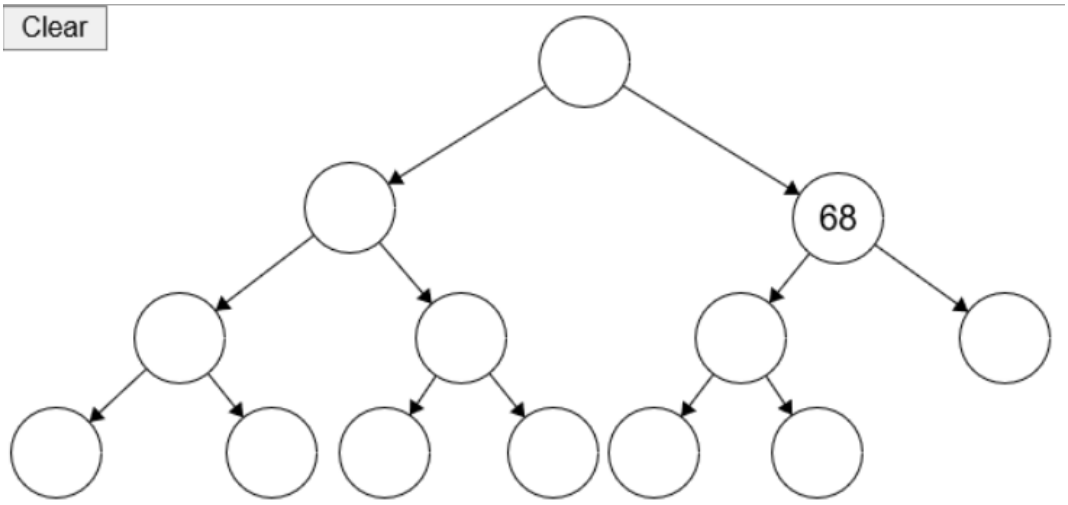
Question 6

Correct

Note de 2,00 sur 2,00

Dans un tas (min-heap) du type ci-dessous, où **il n'y a pas deux fois la même valeur** :
In a heap (min-heap) of the type below, **where there is not twice the same value** :

Clear



1. au maximum, combien de nœuds de l'arbre ont une valeur strictement inférieure à 68 ?
   At most, how many nodes of the tree have a value strictly lower than 68 ?
   8
   ✔
2. au minimum, combien de nœuds de l'arbre ont une valeur strictement inférieure à 68 ?
   at least, how many nodes of the tree have a value strictly lower than 68
   1
   ✔
3. au maximum, combien de nœuds de l'arbre ont une valeur strictement supérieure à 68 ?
   at most, how many nodes in the tree have a value strictly greater than
   11
   ✔
4. au minimum, combien de nœuds de l'arbre ont une valeur strictement supérieures à 68 ?
   at least, how many nodes of the tree have a value strictly greater than 68
   4
   ✔

Question 7

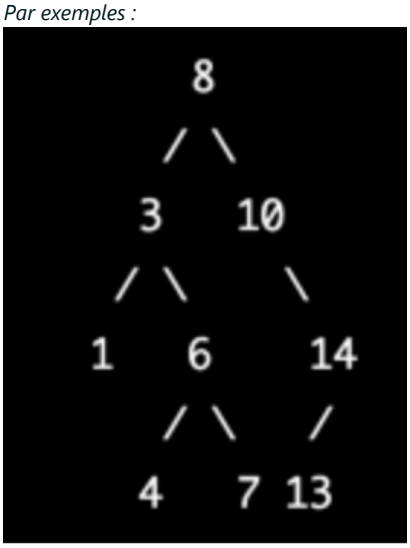
Correct

Note de 8,00 sur 8,00

Implémenter dans la classe BST (pour Binary Search Tree) la méthode qui recherche **le chemin le plus court** entre deux valeurs de nœuds donnés.

Si l'un des noeuds n'est pas dans l'arbre, lever une exception NoSuchElementException avec le message "Element not found".

Par exemples :



Le chemin le plus court de 1 à 3, c'est 1 3
Le chemin le plus court de 1 à 6, c'est 1 3 6
Le chemin le plus court de 7 à 1, c'est 7 6 3 1

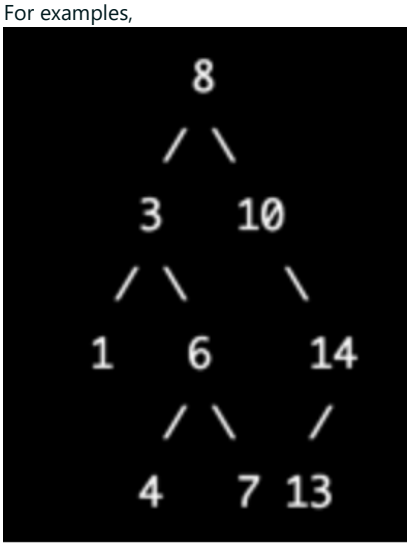
Dans la classe BST, vous pouvez définir toutes les méthodes dont vous avez besoin.

Indices : Trouver l'ancêtre commun des deux nœuds, puis parcourir l'arbre à partir de cet ancêtre commun jusqu'à chacun des deux nœuds. Attention, le chemin vers le nœud de départ doit être inversé (par exemple, l'ancêtre commun à 7 et 1 est 3, le chemin de 3 à 7 est 3, 6, 7, ...)

Implement in the BST (for Binary Search Tree) class the method that searches **the shortest path** between two given node values.

If one of the nodes is not in the tree, throw a *NoSuchElementException* exception with the message "Element not found".

For examples,



The shortest path from 1 to 3 is 1 3

The shortest path from 1 to 6 is 1 3 6

The shortest path from 7 to 1 is 7 6 3 1

Tips : Find the common ancestor of two nodes, then traverse the tree from this common ancestor to the two nodes. Attention, the path to the departure node must be reversed.

In the BST class, you can define all the methods you need.

Par exemple:

Test	Résultat

`//child of`
` BST<Integer> bst = exampleBST();`
`assertEqualsL(Arrays.asList(1,3),bst.shortestPath(1, 3));`
	`test : [1, 3] equals [1, 3] ?`
	`true`

`//Node is not present in the tree`
`BST<Integer> bst = exampleBST();`
`assertThrows(NoSuchElementException.class, () ->`
`bst.shortestPath(1, 9));`
	`test : java.util.NoSuchElementException equals`
	`java.util.NoSuchElementException ?`
	`true`

`//child of`
` BST<Integer> bst = exampleBST();`
`assertEqualsL(Arrays.asList(3,1),bst.shortestPath(3, 1));`
	`test : [3, 1] equals [3, 1] ?`
	`true`

`//Subtree`
`BST<Integer> bst = exampleBST();`
`assertEqualsL(Arrays.asList(1,3,6), bst.shortestPath(1, 6));`
	`test : [1, 3, 6] equals [1, 3, 6] ?`
	`true`

`//long path through the root`
`BST<Integer> bst = exampleBST();`
`assertEqualsL(Arrays.asList(13, 14, 10, 8, 3, 6, 4),`
`bst.shortestPath(13, 4));`
	`test : [13, 14, 10, 8, 3, 6, 4] equals [13, 14, 10, 8, 3, 6, 4] ?`
	`true`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
package ads.poo2.exams.test3;

import java.util.*;

/**
 * A class for Binary Search Trees
 */

public class BST<T extends Comparable<? super T>> implements Iterable<T> {

    public static final String ELEMENT_NOT_FOUND = "Element not found";
    private T element;
    private BST<T> left;
    private BST<T> right;

    public T getElement() {
        return element;
    }
```

	Test	Résultat attendu	Résultat obtenu	

✔	`//child of`
	` BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(1,3),bst.shortestPath(1, 3));`
		`test : [1, 3] equals [1, 3] ?`
		`true`
			`test : [1, 3] equals [1, 3] ?`
			`true`
				✔

✔	`//Node is not present in the tree`
	`BST<Integer> bst = exampleBST();`
	`assertThrows(NoSuchElementException.class, () ->`
	`bst.shortestPath(1, 9));`
		`test : java.util.NoSuchElementException equals java.util.NoSuchElementException ?`
		`true`
			`test : java.util.NoSuchElementException equals java.util.NoSuchElementException ?`
			`true`
				✔

✔	`//child of`
	` BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(3,1),bst.shortestPath(3, 1));`
		`test : [3, 1] equals [3, 1] ?`
		`true`
			`test : [3, 1] equals [3, 1] ?`
			`true`
				✔

	Test	Résultat attendu	Résultat obtenu	

✔	`//Subtree`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(1,3,6),`
	`bst.shortestPath(1, 6));`
		`test : [1, 3, 6] equals [1, 3, 6] ?`
		`true`
			`test : [1, 3, 6] equals [1, 3, 6] ?`
			`true`
				✔

✔	`//same branch bottom to top`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(1,3,8),`
	`bst.shortestPath(1, 8));`
		`test : [1, 3, 8] equals [1, 3, 8] ?`
		`true`
			`test : [1, 3, 8] equals [1, 3, 8] ?`
			`true`
				✔

✔	`//same branch top to bottom`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(8,3,1),`
	`bst.shortestPath(8, 1));`
		`test : [8, 3, 1] equals [8, 3, 1] ?`
		`true`
			`test : [8, 3, 1] equals [8, 3, 1] ?`
			`true`
				✔

✔	`BST<String> bstVocabulary = initVocabulary();`
	`assertEqualsL(Arrays.asList("genievre", "arthur") ,`
	`bstVocabulary.shortestPath("genievre", "arthur"));`
		`test : [genievre, arthur] equals [genievre, arthur] ?`
		`true`
			`test : [genievre, arthur] equals [genievre, arthur] ?`
			`true`
				✔

✔	`//In a subtree`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(7,6,3,1),`
	`bst.shortestPath(7, 1));`
		`test : [7, 6, 3, 1] equals [7, 6, 3, 1] ?`
		`true`
			`test : [7, 6, 3, 1] equals [7, 6, 3, 1] ?`
			`true`
				✔

✔	`//In a subtree`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(1,3,6,4),`
	`bst.shortestPath(1, 4));`
		`test : [1, 3, 6, 4] equals [1, 3, 6, 4] ?`
		`true`
			`test : [1, 3, 6, 4] equals [1, 3, 6, 4] ?`
			`true`
				✔

✔	`//long path through the root`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(13, 14, 10, 8, 3, 6, 4),`
	`bst.shortestPath(13, 4));`
		`test : [13, 14, 10, 8, 3, 6, 4] equals [13, 14, 10, 8, 3, 6, 4] ?`
		`true`
			`test : [13, 14, 10, 8, 3, 6, 4] equals [13, 14, 10, 8, 3, 6, 4] ?`
			`true`
				✔

✔	`BST<String> bstVocabulary = initVocabulary();`
	`assertEqualsL(Arrays.asList("guinevere","genievre",`
	`"buzzards") ,`
	`bstVocabulary.shortestPath("guinevere",`
	`"buzzards"));`
	`assertEqualsL(Arrays.asList("eustace", "dickens",`
	`"claudius") , bstVocabulary.shortestPath("eustace",`
	`"claudius"));`
	`assertEqualsL(Arrays.asList("jubilation", "jaguar",`
	`"inspirit", "hello","arthur","genievre", "buzzards")`
	`, bstVocabulary.shortestPath("jubilation",`
	`"buzzards"));`
	`assertEqualsL(Arrays.asList("eustace") ,`
	`bstVocabulary.shortestPath("eustace", "eustace"));`
		`test : [guinevere, genievre, buzzards] equals [guinevere, genievre, buzzards] ?`
		`true`
		`test : [eustace, dickens, claudius] equals [eustace, dickens, claudius] ?`
		`true`
		`test : [jubilation, jaguar, inspirit, hello, arthur, genievre, buzzards] equals [jubilation, jaguar, inspirit, hello, arthur, genievre, buzzards] ?`
		`true`
		`test : [eustace] equals [eustace] ?`
		`true`
			`test : [guinevere, genievre, buzzards] equals [guinevere, genievre, buzzards] ?`
			`true`
			`test : [eustace, dickens, claudius] equals [eustace, dickens, claudius] ?`
			`true`
			`test : [jubilation, jaguar, inspirit, hello, arthur, genievre, buzzards] equals [jubilation, jaguar, inspirit, hello, arthur, genievre, buzzards] ?`
			`true`
			`test : [eustace] equals [eustace] ?`
			`true`
				✔

Tous les tests ont été réussis ! ✔

▸ **Montrer / masquer la solution de l'auteur de la question (Java)**

Correct

Note pour cet envoi : 8,00/8,00.

Question 8

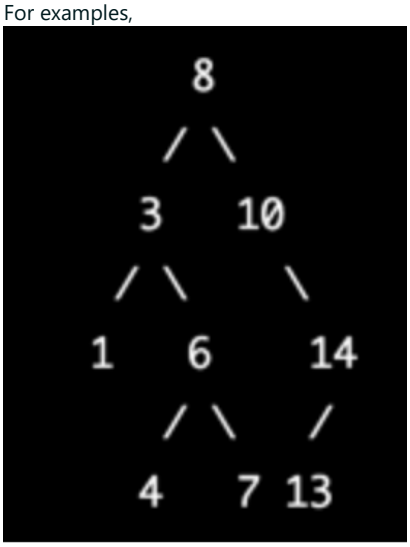
Correct

Note de 4,00 sur 4,00

Implémenter dans la classe BST (pour Binary Search Tree) la méthode qui renvoie l**es k plus petits éléments triés de l'arbre.**

Implement in the BST (for Binary Search Tree) class the method that returns the **k smallest** sorted **elements** of the tree.

For examples,



Par exemple:

Test	Résultat

`BST<Integer> bst = exampleBST();`
`assertEqualsL(Arrays.asList(1,3,4,6,7,8,10,13,14),`
`bst.kSmallestElements(9));`
	`test : [1, 3, 4, 6, 7, 8, 10, 13, 14] equals [1, 3, 4, 6, 7, 8, 10, 13, 14] ?`
	`true`

`BST<Integer> bst = exampleBST();` `assertEqualsL(Arrays.asList(), bst.kSmallestElements(0));`	`test : [] equals [] ?` `true`
`BST<Integer> bst = exampleBST();` `assertEqualsL(Arrays.asList(1,3), bst.kSmallestElements(2));`	`test : [1, 3] equals [1, 3] ?` `true`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?

Falling back to raw text area.

```
package ads.poo2.exams.test3;

import java.util.*;

/**
 * A class for Binary Search Trees
 */

public class BST<T extends Comparable<? super T>> implements Iterable<T> {

    public static final String ELEMENT_NOT_FOUND = "Element not found";
    private T element;
    private BST<T> left;
    private BST<T> right;

    public T getElement() {
        return element;
    }
```

	Test	Résultat attendu	Résultat obtenu	
✔	`BST<Integer> bst = exampleBST();` `assertEqualsL(Arrays.asList(1,3,4,6,7,8,10,13,14), bst.kSmallestElements(9));`	`test : [1, 3, 4, 6, 7, 8, 10, 13, 14] equals [1, 3, 4, 6, 7, 8, 10, 13, 14] ?` `true`	`test : [1, 3, 4, 6, 7, 8, 10, 13, 14] equals [1, 3, 4, 6, 7, 8, 10, 13, 14] ?` `true`	✔
✔	`BST<Integer> bst = exampleBST();` `assertEqualsL(Arrays.asList(), bst.kSmallestElements(0));`	`test : [] equals [] ?` `true`	`test : [] equals [] ?` `true`	✔
✔	`BST<Integer> bst = exampleBST();` `assertEqualsL(Arrays.asList(1), bst.kSmallestElements(1));`	`test : [1] equals [1] ?` `true`	`test : [1] equals [1] ?` `true`	✔
✔	`BST<Integer> bst = exampleBST();` `assertEqualsL(Arrays.asList(1,3), bst.kSmallestElements(2));`	`test : [1, 3] equals [1, 3] ?` `true`	`test : [1, 3] equals [1, 3] ?` `true`	✔

✔	`//same branch bottom to top`
	`BST<Integer> bst = exampleBST();`
	`assertEqualsL(Arrays.asList(1,3,4,6), bst.kSmallestElements(4));`
		`test : [1, 3, 4, 6] equals [1, 3, 4, 6] ?`
		`true`
			`test : [1, 3, 4, 6] equals [1, 3, 4, 6] ?`
			`true`
				✔

Tous les tests ont été réussis ! ✔

▸ **Montrer / masquer la solution de l'auteur de la question (Java)**

Correct

Note pour cet envoi : 4,00/4,00.