```python
from doctest import testmod


class Noeud:
    '''
    Classe implémentant un noeud d'arbre binaire
    disposant de 3 attributs :
    - valeur : la valeur de l'étiquette,
    - gauche : le sous-arbre gauche.
    - droit : le sous-arbre droit.
    '''
    def __init__(self, v, g, d):
        self.valeur = v
        self.gauche = g
        self.droite = d


class ABR:
    '''
    Classe implémentant une structure
    d'arbre binaire de recherche.
    '''

    def __init__(self):
        '''Crée un arbre binaire de recherche vide'''
        self.racine = None

    def est_vide(self):
        '''Renvoie True si l'ABR est vide et False sinon.'''
        return self.racine is None

    def parcours(self, tab = []):
        '''
        Renvoie la liste tab complétée avec tous les
        éléments de l'ABR triés par ordre croissant.

        Tests et Exemples:
        >>> a = ABR()
        >>> a.insere(7)
        >>> a.insere(3)
        >>> a.insere(9)
        >>> a.insere(1)
        >>> a.insere(9)
        >>> a.parcours()
        [1, 3, 7, 9, 9]
        '''
        if self.est_vide():
            return tab
        else:
            self.racine.gauche.parcours(tab)
            # ajoute la valeur de la racine au tableau
            tab.append(self.racine.valeur)
            # parcours récursif à droite
            self.racine.droite.parcours(tab)
            return tab

    def insere(self, element):
        '''Insère un élément dans l'arbre binaire de recherche.'''
        if self.est_vide():
            self.racine = Noeud(element, ABR(), ABR())
        else:
            if element < self.racine.valeur:
                self.racine.gauche.insere(element)
            else :
                self.racine.droite.insere(element)

    def recherche(self, element):
        '''
        Renvoie True si element est présent dans l'arbre
        binaire et False sinon.

        Tests et Exemples:
        >>> a = ABR()
```

```
        >>> a.insere(7)
        >>> a.insere(3)
        >>> a.insere(9)
        >>> a.insere(1)
        >>> a.insere(9)
        >>> a.recherche(4)
        False
        >>> a.recherche(3)
        True
        '''
        if self.est_vide():
            # si l'ABR est vide, element ne peut pas s'y trouver !
            return False
        else:
            if element < self.racine.valeur:
                # recherche récursive dans le sous ABR de gauche
                return self.racine.gauche.recherche(element)
            elif element > self.racine.valeur:
                # recherche récursive dans le sous ABR de droite
                return self.racine.droite.recherche(element)
            else:
                # si element n'est ni > ni < à valeur
                # c'est qu'ils sont égaux !
                return True


if __name__ == '__main__':
    testmod()
```