ARBRES BINAIRES DE RECHERCHE

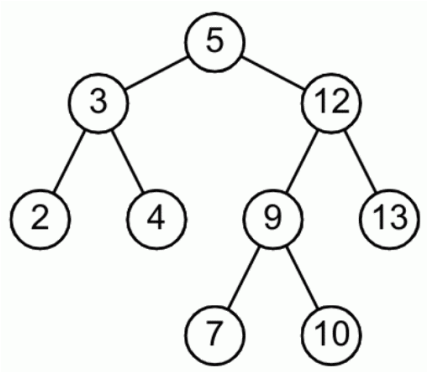
I) Définition

> On appelle **arbre binaire de recherche (ABR)**, un arbre binaire construit de telle sorte que pour chaque nœud :
> étiquettes du sous-arbre gauche < étiquette du nœud < étiquettes du sous-arbre droit

Remarques :

- Les ABR sont très utiles quand on manipule un ensemble souvent modifié de valeurs (insertions ou suppressions) sur lesquelles on doit faire des recherches fréquentes.
- Dans la plupart des cas, on interdira les étiquettes en double dans un ABR.

Ex :

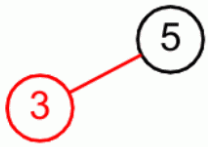
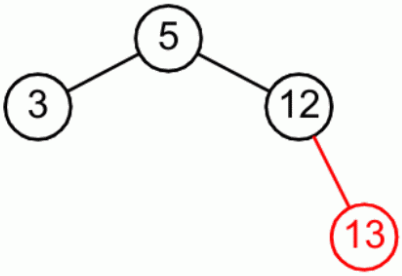
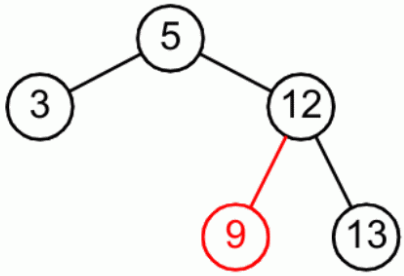
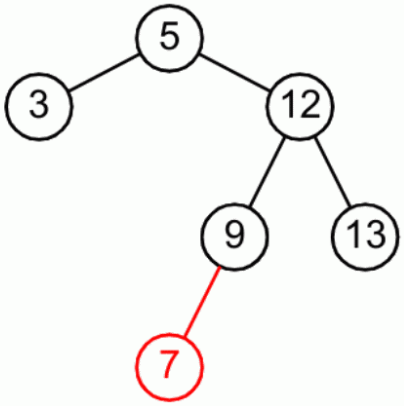
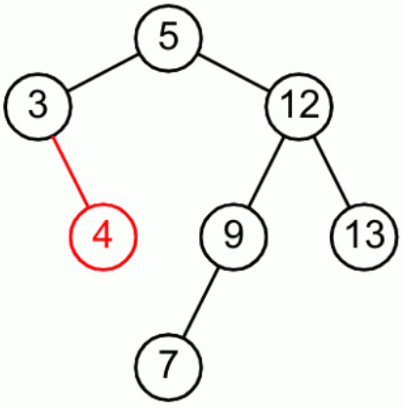
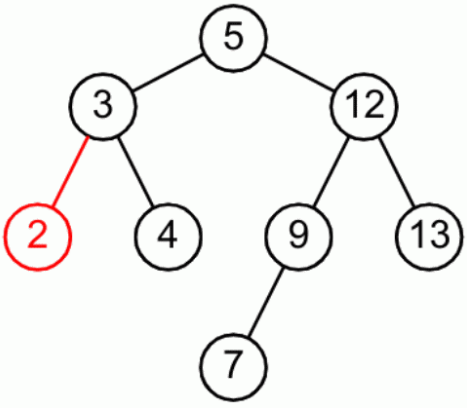
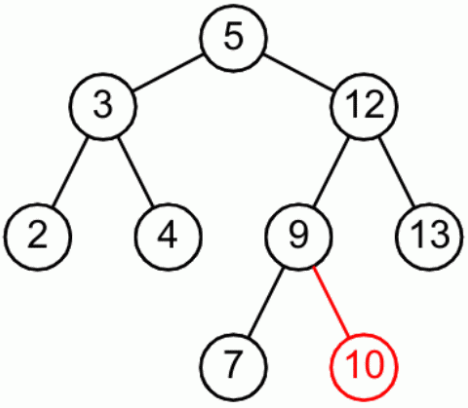


II) Construire un arbre binaire de recherche

Ex : On souhaite créer un ABR à partir de la liste [5, 3, 12, 13, 9, 7, 4, 2, 10].

Le premier item de la liste va être la racine de l'arbre. Puis, en partant de cette racine, on va insérer tous les items qui suivent en respectant la règle de construction vue précédemment :

étiquettes du sous-arbre gauche < étiquette du nœud < étiquettes du sous-arbre droit

Insertion du 5 à la racine 	Insertion du 3 	Insertion du 12 
Insertion du 13 	Insertion du 9 	Insertion du 7 
Insertion du 4 	Insertion du 2 	Insertion du 10 

Algorithme récursif :

On considérera dans l'algorithme ci-dessous que :

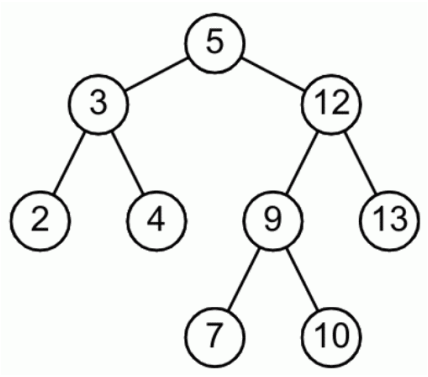
- nœud n'est pas vide.
- on veut créer un arbre sans doublon (si la valeur à insérer dans l'arbre existe déjà, on ne fait rien)

```
insère(valeur, nœud)
    si valeur < nœud :
        si nœud a un fils gauche :
            insère(valeur, fils gauche)
        sinon :
            créer un fils gauche avec cette valeur
    si valeur > nœud :
        si nœud a un fils droit :
            insère(valeur, fils droit)
        sinon :
            créer un fils droit avec cette valeur
```

A faire : En utilisant cet algorithme, faire l'arbre des appels récursifs dans le cas où on appelle insère(10, noeud5) dans l'arbre de la page précédente à la dernière étape.

III) Parcours d'un arbre binaire de recherche

Parcourir l'arbre ci-dessous avec les 4 méthodes de parcours d'un arbre binaire déjà rencontrées :



Parcours en largeur d'abord :

Parcours préfixe :

Parcours infixe :

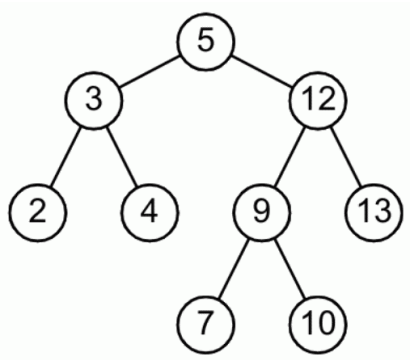
Parcours postfixe :

Lequel de ces parcours permet d'obtenir une version triée de la liste initiale ?

IV) Recherche dans un arbre binaire de recherche

Voici un algorithme récursif pour chercher une valeur dans un ABR :

```
recherche(valeur, nœud)
    si nœud est vide
        renvoyer faux
    sinon si nœud = valeur
        renvoyer vrai
    sinon si nœud > valeur
        renvoyer recherche(valeur, fils gauche)
    sinon
        renvoyer recherche(valeur, fils droit)
```



Utiliser cet algorithme pour compléter le tableau ci-dessous :

Valeur cherchée :	4	1	8	17	10
Réponse de l'algorithme :	Vrai				
Nombre d'appels récursifs :	3				

Complexité de l'algorithme :

Pour parcourir un ABR contenant n éléments, il faut dans le pire des cas un nombre d'étapes égal à la hauteur h de l'arbre.

Or si l'arbre est raisonnablement équilibré, on a :

$$n \approx 2^{h+1} - 1 \text{ , donc } n + 1 \approx 2^{h+1} \text{ , donc } h + 1 \approx \log_2(n + 1).$$

La complexité de l'algorithme ci-dessus est donc **logarithmique**, c'est à dire nettement meilleure qu'une recherche dans une liste non triée ! Par exemple, une recherche dans un ABR bien équilibré contenant 1000 chiffres se fait en maxi 10 étapes.

Avec quel algorithme vu en 1ère peut-on faire le lien ?