

# Arbres binaires de recherche

## Chapitre 14

---

NSI2

8 décembre 2022

Un arbre binaire de recherche est un arbre binaire

Un arbre binaire de recherche est un arbre binaire

- dont tous les nœuds comportent des valeurs du même type qu'il est possible de comparer (entiers, flottants, chaînes de caractères...);

Un arbre binaire de recherche est un arbre binaire

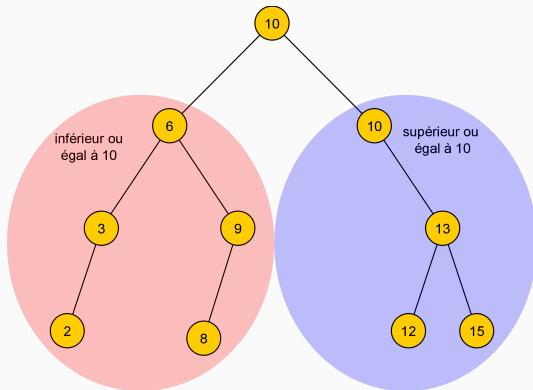
- dont tous les nœuds comportent des valeurs du même type qu'il est possible de comparer (entiers, flottants, chaînes de caractères...);
- dont les valeurs de tous les nœuds situés dans le sous-arbre gauche d'un nœud sont inférieures ou égales à la valeur de ce nœud;

Un arbre binaire de recherche est un arbre binaire

- dont tous les nœuds comportent des valeurs du même type qu'il est possible de comparer (entiers, flottants, chaînes de caractères...);
- dont les valeurs de tous les nœuds situés dans le sous-arbre gauche d'un nœud sont inférieures ou égales à la valeur de ce nœud;

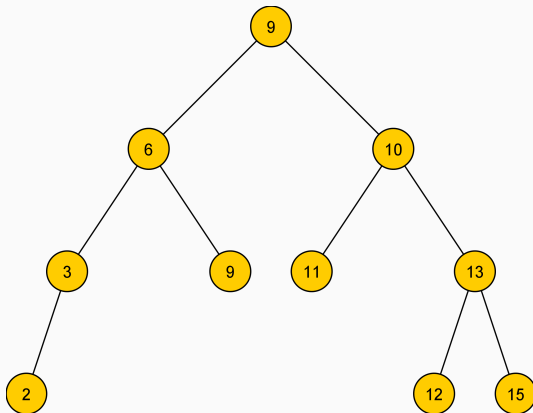
# Exemple

Voici un ABR :



# Exemple

Ceci n'est pas un ABR









# Ajout d'un élément dans un ABR

# Suppression d'un élément dans un ABR

Ceci est hors programme.

# Intérêts

---

Le coût de recherche ou d'ajout d'un élément dans un ABR est proportionnel à sa hauteur.

Dans le pire des cas, où l'ABR est dégénéré, on ne gagne pas grand chose par rapport à une liste.

Le meilleur des cas serait d'avoir un arbre parfait.

Il existe des méthodes pour, lors de l'ajout d'un élément dans un ABR, s'assurer que celui-ci reste bien *équilibré*. Elles ne sont pas au programme de terminale.

Dans ce cas la hauteur  $h$  de l'ABR est dite *logarithmique*, c'est à dire que si  $N$  désigne le nombre de nœuds, il existe une constante  $C$  telle que

$$h \leq C \cdot \log_2(N)$$

Alors, avec un ABR équilibré, les opérations de recherche et d'ajouts sont très efficaces.

# Exemple

Imaginons un ABR parfait de hauteur  $h$ , il possède  $N = 2^{h+1} - 1$  éléments.

Le temps de recherche ou d'ajout d'un élément (dans le pire des cas) est proportionnel à  $h$ .

Prenons  $h = 50$ , alors notre ABR comporte 2 251 799 813 685 247 éléments, et on trouve ou on ajoute un nouvel élément en environ 50 étapes au maximum !