# Arbres de recherche binaires

Programmation séquentielle en C, 2020-2021

Orestis Malaspinas (A401), ISC, HEPIA

2021-03-24

Inspirés des slides de F. Glück

# Arbres binaires de recherche

Qu'est-ce qu'un *arbre binaire de recherche* ou *binary search tree*?

**Définition**

1. Arbre binaire: chaque nœud à **au plus** deux sommets enfants.
2. Chaque nœud contient une clé.
3. Pour chaque nœud, tous les nœud du **sous-arbre gauche** ont une clé inférieure ou égale au nœud considéré et tous les nœuds du **sous-arbre droit** du nœud considéré ont une clé supérieure ou égale au nœud considéré.

La propriété 3 est connue sous le nom de propriété ABR (ou BST property).

**Utilité**

- Recherche de clé, *min/max*.
- File de priorité.
- Parcours ordonné (parcours infixe) $=>$ tri.

# Exemple

**Un arbre binaire de recherche d'entiers**

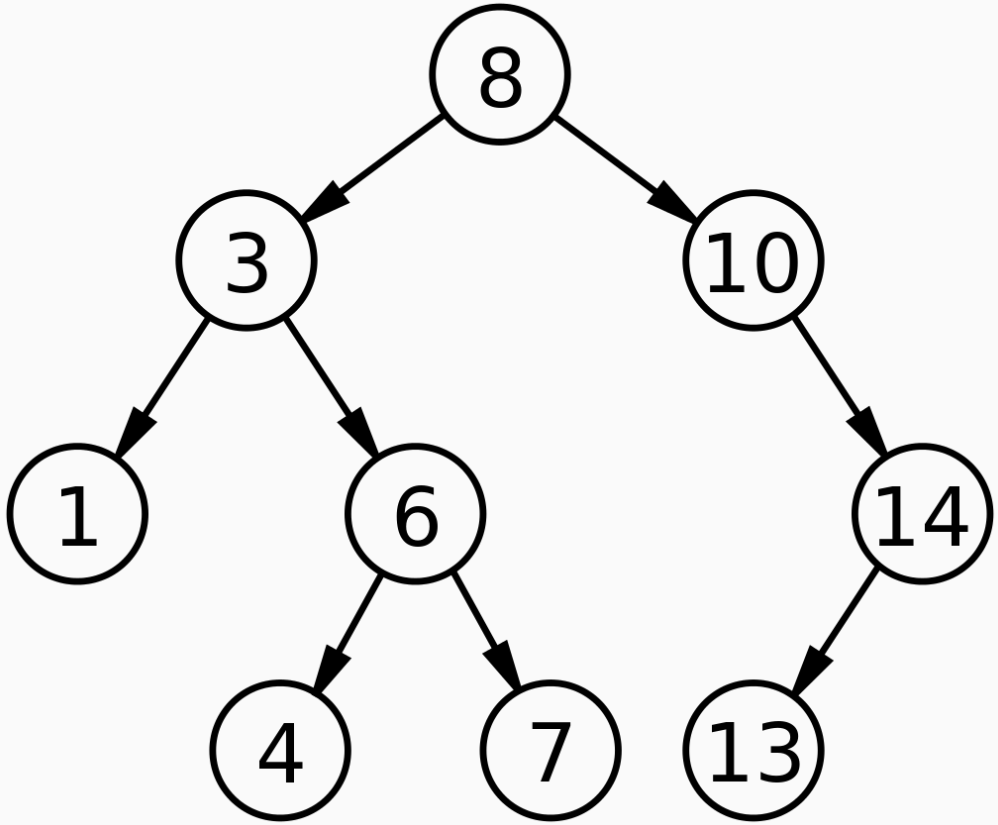


**Figure 1** – Exemple d'ABR contenant 9 valeurs. Source: Wikipedia.

### Fonctionnalités importantes

1. Insertion de clé
2. Recherche de clé
3. Suppression de clé

### Récursivité

La propriété ABR est **récursive** (les sous-arbres doivent être ABR):

- L'enfant de gauche est plus petit que nœud courant?
- L'enfant de droite est plus petit que nœud courant?
- Le sous-arbre de gauche/droite est-il ABR?

# L'insertion

Insertion récursive:

```
tree insert(tree, key) {
    // fin de la récursion, on crée le noeud et retourne
    si l'arbre est vide:
        return create_node(key);
    // recherche d'où insérer
    si la clé est plus petite que le noeud courant:
        si le sous-arbre gauche est vide:
            sous-arbre gauche = create_node(key);
        sinon:
            return insert(sous-arbe gauche, key);
    si la clé est plus grande que le noeud courant:
        si le sous-arbre droit est vide:
            sous-arbre droit = create_node(key);
        sinon:
            return insert(sous-arbre droit, key);
    // on retourne l'arbre!
    return tree;
}
```