NSI · TERMINALE / COURS

Les arbres binaires de recherche

Astrolab Cours · astrolabcours.fr

01 Le problème

Un annuaire contient un million de noms. Les parcourir un à un pour en trouver un est trop long. Un tableau trié permet la dichotomie, mais insérer un nouveau nom oblige à tout décaler. Peut-on chercher vite et insérer vite ?

01 L'idée : plus grand ou plus petit

Dans le jeu « je pense à un nombre », chaque réponse « plus grand » ou « plus petit » élimine la moitié des possibilités. Un arbre binaire de recherche range les valeurs pour qu'à chaque nœud, on se pose cette question et qu'on ne descende que d'un côté.

01 La propriété d'ordre

Un **arbre binaire de recherche** (ABR) est un arbre binaire où, pour chaque nœud de valeur v :

- toutes les valeurs du sous-arbre gauche sont **inférieures** à v ;
- toutes les valeurs du sous-arbre droit sont **supérieures** à v.

01 Chercher

```
def appartient(a, x):
    if a is None:
        return False
    if x == a.valeur:
        return True
    if x < a.valeur:
        return appartient(a.gauche, x)
    return appartient(a.droit, x)
```

On ne descend que dans un seul sous-arbre : le nombre de comparaisons est au plus la hauteur de l'arbre.

Insérer

On descend comme pour chercher, et on place la nouvelle valeur à l'endroit vide où l'on arrive : elle devient une feuille.

© Astrolab Cours · astrolabcours.fr/nsi/terminale/arbres-binaires-de-recherche/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

```python
def inserer(a, x):
    if a is None:
        return Noeud(x)
    if x < a.valeur:
        a.gauche = inserer(a.gauche, x)
    else:
        a.droit = inserer(a.droit, x)
    return a
```

Le parcours infixe trie

Gauche, racine, droit : on visite d'abord les plus petites valeurs, puis la racine, puis les plus grandes. Le parcours infixe d'un ABR donne les valeurs **dans l'ordre croissant**.

Équilibré ou filiforme

Un arbre **équilibré** de n nœuds a une hauteur proche de $\log_2 n$: environ 20 pour un million de valeurs. Mais en insérant 1, 2, 3, 4… dans cet ordre, chaque valeur va à droite de la précédente : l'arbre est **filiforme**, de hauteur n.

Exemple guidé

On insère dans un arbre vide, dans cet ordre : 8, 3, 10, 1, 6, 14, 4. Puis on cherche 4.

1. **8** devient la racine. **3** < 8 : à gauche. **10** > 8 : à droite.
2. **1** : plus petit que 8, puis que 3, il va à gauche de 3. **6** : à gauche de 8, à droite de 3.
3. **14** : à droite de 8, puis de 10. **4** : gauche de 8, droite de 3, gauche de 6.
4. **Chercher 4** : 4 < 8 on descend à gauche, 4 > 3 à droite, 4 < 6 à gauche, trouvé. Quatre comparaisons, au lieu de sept en lisant tout.

Le parcours infixe (gauche, racine, droite) donne 1, 3, 4, 6, 8, 10, 14 : la liste triée.

Retour au problème

Dans un arbre équilibré d'un million de noms, la hauteur est d'environ 20. Chercher ou insérer coûte une vingtaine de comparaisons, sans rien décaler : on ajoute juste une feuille au bon endroit.

Les pièges

- La propriété porte sur **tout** le sous-arbre, pas seulement sur les fils directs.
- Insérer des valeurs triées donne un arbre filiforme, donc lent.
- La plus petite valeur est tout en bas à gauche, pas forcément une feuille quelconque.

© Astrolab Cours · astrolabcours.fr/nsi/terminale/arbres-binaires-de-recherche/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits