DSA · FICHE PRATIQUE

Cheatsheet : Arbres de recherche et tas

Distinguer ordre de recherche, hauteur et priorité minimale.

Par Ali ZAINOUL · Cours gratuit

Définitions, contrats et coûts

Distinguer ordre de recherche, hauteur et priorité minimale.

Arbres de recherche et tas

Notion	À retenir
Arbre de recherche	Recherche O(h + 1) ; O(log n) avec équilibrage, Θ(n) possible sinon.
Tas minimum	Minimum Θ(1), insertion et extraction O(log n), hors coût de croissance du stockage.
Invariant du tas	Chaque parent précède ses enfants ; le tableau entier n'est pas trié.

Vérifier sur un exemple

Extraire des priorités

heapq représente un tas minimum dans une liste ; un plus petit nombre signifie ici une priorité plus forte.

Python

```python
# minimum_heap.py
import heapq
priorities = [5, 1, 3]
heapq.heapify(priorities)
result = [heapq.heappop(priorities) for _ in range(3)]
print(result)
# Expected output:
# [1, 3, 5]
```

Résultat expliqué. heapify coûte O(n) ; les trois extractions illustrent l'ordre croissant. Pour n extractions, le coût total est O(n log n).

Erreur à éviter. Un arbre de recherche non équilibré n'a pas automatiquement une hauteur logarithmique.

▼ Pourquoi le deuxième élément du tableau du tas n'est-il pas nécessairement le deuxième minimum ?

L'invariant ne compare que parents et enfants ; il ne trie pas les éléments de branches différentes.

Télécharger cette fiche en PDF · Toutes les fiches DSA

Version web : https://www.clearcode.fr/courses/programming/dsa/cheatsheet/trees-heaps