Tri rapide

Diviser pour régner

Ouverture hors programme de terminale NSI. Le tri rapide n'est pas exigible au baccalauréat ; il illustre le paradigme diviser pour régner et se compare au tri fusion, qui est, lui, au programme.

Objectifs.
Prérequis : diviser pour régner, récursivité
À l'issue de ce chapitre, vous saurez :

- expliquer le principe du tri rapide (choix du pivot, partition, appels récursifs) ;
- analyser sa complexité dans le meilleur cas ($O(n \log n)$) et dans le pire cas ($O(n^2)$) ;
- comparer le tri rapide au tri fusion en termes de complexité, stabilité et performances pratiques ;
- implémenter le tri rapide en Python.

Principe du tri rapide

Le tri rapide fonctionne selon le principe suivant :

1. On choisit un élément du tableau appelé **pivot**.
2. On réorganise le tableau de sorte que tous les éléments plus petits que le pivot soient à sa gauche et tous les éléments plus grands soient à sa droite.
3. On applique récursivement le tri rapide sur les deux sous-tableaux (gauche et droite).

Exemple : soit le tableau `[7, 2, 1, 6, 8, 5, 3, 4]`. Si on choisit 4 comme pivot :

- la liste des nombres inférieurs à 4 est `[2, 1, 3]` ;
- celle des nombres égaux au pivot est `[4]` ;
- et celle des nombres supérieurs à 4 est `[7, 6, 8, 5]`.

En concaténant les trois listes, on obtient `[2, 1, 3, 4, 7, 6, 8, 5]` puis on trie récursivement `[2, 1, 3]` et `[7, 6, 8, 5]`.

Diviser pour régner

Exercice : le paradigme diviser pour régner

1. Rappelez en quelques lignes le principe du paradigme **diviser pour régner**.
2. En quoi le tri rapide utilise-t-il ce paradigme ?
3. Quelle est la principale différence entre la façon dont le tri fusion et le tri rapide divisent le problème ?

Analyse de la complexité

Complexité dans le meilleur cas

Dans le meilleur cas, le pivot divise toujours le tableau en deux parties à peu près égales.

Exercice : meilleur cas

1. Si on divise toujours le tableau en deux parties égales, combien de niveaux de récursivité y aura-t-il pour un tableau de taille n ?
2. À chaque niveau, quelle est la complexité totale pour partitionner tous les sous-tableaux ?
3. Quelle est donc la complexité dans le meilleur cas ? Comparez avec le tri fusion.

Cas particuliers défavorables

Exercice : pire cas

1. Considérez le tableau déjà trié : `[1, 2, 3, 4, 5, 6, 7, 8]`. Si on choisit systématiquement le **premier élément** comme pivot, que se passe-t-il à chaque étape ?
2. Combien de niveaux de récursivité y aura-t-il dans ce cas ?
3. Quelle est la complexité dans le pire cas du tri rapide ?
4. Donnez d'autres exemples de tableaux qui conduisent au pire cas (selon le choix du pivot).

Comparaison avec le tri fusion

Exercice : tri rapide vs tri fusion

1. Complétez le tableau suivant :

Critère	Tri fusion	Tri rapide
Meilleur cas		
Cas moyen		
Pire cas		
Complexité spatiale		
Stabilité		

2. Le tri fusion est dit **stable**. Qu'est-ce que cela signifie ? Le tri rapide est-il stable ?
3. Dans quelles situations préférerait-on le tri fusion au tri rapide ?
4. Malgré son pire cas en $O(n^2)$, pourquoi le tri rapide est-il souvent utilisé en pratique ?

Implémentation en Python

Exercice : compléter le code

Voici une implémentation simple du tri rapide à compléter :

```python
def tri_rapide(tab):
    if len(tab) <= 1:
        return tab

    pivot = tab[0]

    petits = [x for x in tab[1:] if .........]
    egaux = [x for x in tab if ...]
    grands = [x for x in tab[1:] if .........]

    return ..................... + egaux + .....................
```

1. Complétez les trois compréhensions de liste pour séparer les éléments selon leur valeur par rapport au pivot.
2. Complétez la dernière ligne pour combiner les résultats des appels récursifs.
3. Testez votre implémentation sur les tableaux `[3, 6, 8, 10, 1, 2, 1]`, `[1, 2, 3, 4, 5]` et `[5, 4, 3, 2, 1]`.

Pour aller plus loin

Exercice : optimisations

1. Comment pourrait-on améliorer le choix du pivot pour éviter le pire cas ?
2. Recherchez ce qu'est le **tri rapide randomisé**. En quoi cela améliore-t-il les performances ?
3. Pour de petits tableaux (taille < 10), le tri rapide n'est pas optimal. Quelle optimisation peut-on envisager ?

À retenir.

— Le **tri rapide** suit le paradigme diviser pour régner : il partitionne le tableau autour d'un **pivot**, puis trie récursivement chaque partie.
— Sa complexité est $\mathbf{O(n \log n)}$ en moyenne et dans le meilleur cas, mais $\mathbf{O(n^2)}$ dans le pire cas (pivot systématiquement mal choisi).
— Contrairement au tri fusion, le travail principal se fait lors de la **division** (partition) et non lors de la combinaison.
— En pratique, le tri rapide est souvent très performant ; un choix aléatoire ou médian du pivot rend le pire cas extrêmement improbable.