STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 16

Partitionner autour d’un pivot

Comprenez partition, pivot, cas dégénéré et aléa. Étudiez une version pédagogique avec copies sans lui attribuer les propriétés d’un tri en place.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Partitionner autour d’un pivot et reconnaître les cas dégénérés.
- Les trois partitions conservent toutes les occurrences inférieures, égales ou supérieures au pivot.
- Les appels excluent la partition égale, ce qui garantit la diminution pour les données autorisées.

Prérequis. Avoir étudié les chapitres DSA précédents ; connaître tableaux, invariants et récursivité.

Partir d’un besoin

Après avoir choisi une valeur pivot, toutes les valeurs plus petites peuvent-elles être triées indépendamment de celles qui sont plus grandes ?

Définir la structure et les conventions

Le **tri rapide** partitionne l’entrée autour d’un **pivot**, une valeur choisie dans la suite. Notre version crée trois groupes : inférieur, égal et supérieur au pivot. Elle trie récursivement les groupes inférieur et supérieur, puis concatène les trois groupes.

L’invariant de partition est : chaque valeur déjà lue se trouve exactement dans le groupe conforme à sa comparaison au pivot. Le groupe égal conserve toutes les occurrences du pivot. Les appels récursifs excluent ce groupe ; leur taille diminue donc pour des nombres finis totalement ordonnés. Les NaN sont exclus du contrat, car leurs comparaisons ne suivent pas cet ordre.

Python utilise ici des compréhensions de listes : `[value for value in values if value < pivot]` garde les valeurs qui satisfont le filtre. MATLAB crée un masque logique avec values < pivot, puis utilise ce masque pour sélectionner les éléments. Les deux versions construisent des groupes supplémentaires ; elles ne trient pas en place.

Suivre les états

Pivot 3 dans [3, 1, 3, 2, 5]

Groupe	Contenu
Inférieur	[1, 2]
Égal	[3, 3]
Supérieur	[5]

Comparer les réalisations

La sortie vaut [1, 2, 3, 3, 5] et l’entrée est conservée. Enregistrez la version MATLAB dans quick_sort_demo.m. Le premier élément est toujours choisi comme pivot : aucune randomisation n’est réalisée par le code.

Pseudo-code

```
// example.pseudo
// quick_sort
// Partition into values < pivot, == pivot and > pivot.
// Recursively sort the lower and upper groups.
// Join sorted lower, equal, sorted upper.
// Never recurse on the equal group.
```

Python

```python
# quick_sort.py
def quick_sort(values):
    if len(values) <= 1:
        return values.copy()
    pivot = values[0]
    lower = [value for value in values if value < pivot]
    equal = [value for value in values if value == pivot]
    upper = [value for value in values if value > pivot]
    return quick_sort(lower) + equal + quick_sort(upper)
original = [3, 1, 3, 2, 5]
ordered = quick_sort(original)
print(ordered)
```

MATLAB

```
% quick_sort_demo.m
original = [3, 1, 3, 2, 5];
ordered = quick_sort(original);
disp(ordered);
function ordered = quick_sort(values)
    if numel(values) <= 1
        ordered = values;
        return;
    end
    pivot = values(1);
    lower = values(values < pivot);
    equal = values(values == pivot);
    upper = values(values > pivot);
    ordered = [quick_sort(lower), equal, quick_sort(upper)];
end
```

Justifier et compter

Une partition coûte Θ(n). Des divisions équilibrées donnent Θ(n log n) ; choisir toujours un extrême sur des valeurs distinctes conduit à Θ(n²). Un pivot aléatoire peut donner un temps attendu O(n log n), mais ce n'est pas une garantie du pivot fixe montré ici. Les groupes copiés occupent O(n) au pic dans le cas équilibré et peuvent retenir O(n²) valeurs cumulées dans une chaîne d'appels dégénérée. La profondeur peut atteindre n : les limites de récursion sont alors un risque concret.

Prévoir puis expérimenter

Comparez le premier pivot d'une suite croissante de valeurs distinctes et celui d'une suite entièrement égale. La première provoque une division déséquilibrée ; la seconde termine après une partition grâce au groupe égal.

Exercice corrigé

▼ Peut-on annoncer que cette fonction utilise O(log n) mémoire parce que le tri rapide est souvent présenté ainsi ?

Non. La mémoire dépend de l'implémentation et des partitions. Ces versions copient les groupes et peuvent conserver de nombreux tableaux dans les appels en attente.

À retenir

- Les trois partitions conservent toutes les occurrences inférieures, égales ou supérieures au pivot.
- Les appels excluent la partition égale, ce qui garantit la diminution pour les données autorisées.
- Un pivot fixe peut dégénérer ; la version avec groupes copiés ne possède pas les garanties d'une variante en place.

Version web : https://www.clearcode.fr/courses/programming/dsa/search-sort/quick-sort