STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 15

Diviser puis fusionner deux suites triées

Construisez le tri fusion, tracez les indices de fusion et reliez division, niveaux de récursion, temps et mémoire auxiliaire.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Décomposer un tri puis fusionner deux suites ordonnées.
- Le cas de base est une suite d'au plus une valeur.
- La fusion choisit la plus petite tête disponible et la gauche sur égalité.

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

Partir d'un besoin

Deux petites suites déjà triées se fusionnent en choisissant toujours la plus petite tête disponible. Peut-on appliquer ce principe après avoir trié séparément deux moitiés ?

Définir la structure et les conventions

Le **tri fusion** divise la suite en deux parties presque égales, trie chacune récursivement, puis fusionne leurs résultats. Une suite de zéro ou une valeur est déjà triée : c'est le cas de base. Les sous-problèmes sont strictement plus petits pour toute taille supérieure à un.

La **fusion** compare les premières valeurs non consommées de deux suites triées. L'invariant est que la sortie contient les plus petites valeurs déjà choisies, en ordre, et que les indices i et j désignent les prochaines candidates. Dès qu'un côté est épuisé, on copie le reste de l'autre.

Les tranches Python values[:middle] et values[middle:] créent les deux moitiés. Les segments MATLAB values(1:middle) et values(middle + 1:end) ont le même rôle. Python extend ajoute toutes les valeurs d'une séquence à une liste. MATLAB préalloue la sortie avec zeros puis la remplit. Les données sont des nombres finis comparables.

Suivre les états

Fusion de [1, 3] et [2, 4]

Candidates	Choix	Sortie
1 et 2	1	[1]
3 et 2	2	[1, 2]
3 et 4	3	[1, 2, 3]
Gauche épuisée	4	[1, 2, 3, 4]

Comparer les réalisations

Le résultat est [1, 2, 3, 4] et original reste inchangé. Les fonctions renvoient un résultat distinct en Python ; la version MATLAB reçoit un vecteur ligne. Enregistrez merge_sort_demo.m avec la fonction locale en fin de script. Choisir le côté gauche sur égalité conserve la stabilité si chaque moitié est elle-même stable.

Pseudo-code

```
// example.pseudo
// merge_sort
// Base case: 0 or 1 value is already sorted.
// Recursively sort the left and right halves.
// Merge their first unconsumed values, choosing the left on equality.
// Copy the remaining suffix when one half is exhausted.
```

Python

```python
# merge_sort.py
def merge_sort(values):
    if len(values) <= 1:
        return values.copy()
    middle = len(values) // 2
    left = merge_sort(values[:middle])
    right = merge_sort(values[middle:])
    merged = []
    i = 0
    j = 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            merged.append(left[i])
            i = i + 1
        else:
            merged.append(right[j])
            j = j + 1
    merged.extend(left[i:])
    merged.extend(right[j:])
    return merged
original = [4, 1, 3, 2]
ordered = merge_sort(original)
print(ordered)
```

MATLAB

```
% merge_sort_demo.m
original = [4, 1, 3, 2];
ordered = merge_sort(original);
disp(ordered);
function ordered = merge_sort(values)
    if numel(values) <= 1
        ordered = values;
        return;
    end
    middle = floor(numel(values) / 2);
    left = merge_sort(values(1:middle));
    right = merge_sort(values(middle + 1:end));
    ordered = zeros(1, numel(values));
    i = 1;
    j = 1;
    for k = 1:numel(values)
        if i <= numel(left) && (j > numel(right) || left(i) <= right(j))
            ordered(k) = left(i);
            i = i + 1;
        else
            ordered(k) = right(j);
            j = j + 1;
        end
    end
end
```

Justifier et compter

Fusionner m valeurs coûte Θ(m). À chaque niveau de division, les tailles fusionnées totalisent n et il y a Θ(log n) niveaux : temps Θ(n log n) pour n ≥ 2. On écrit aussi T(n) = 2T(n/2) + Θ(n) pour des tailles puissances de deux. Dans ces versions, les copies, sorties et moitiés vivantes occupent O(n) mémoire au pic, avec O(log n) appels simultanés. Le total des allocations au fil du calcul n'est pas la mémoire maximale simultanée.

Prévoir puis expérimenter

Fusionnez à la main [1, 2, 2] et [2, 3] en étiquetant les trois occurrences de 2. Vérifiez l'ordre relatif, puis testez une taille impaire et le vide.

Exercice corrigé

▼ **Pourquoi une liste chaînée change-t-elle la manière d'implémenter la fusion ?**

On peut y relier les nœuds au lieu de recopier leurs valeurs. Le principe du tri reste le même, mais les accès et allocations dépendent de la représentation.

À retenir

- Le cas de base est une suite d'au plus une valeur.
- La fusion choisit la plus petite tête disponible et la gauche sur égalité.
- Les divisions donnent un temps Θ(n log n) ; les copies et la sortie demandent un stockage supplémentaire.

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