STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 14

Agrandir un préfixe déjà trié

Implémentez le tri par insertion, justifiez son invariant et sa stabilité, puis comptez les décalages selon l'ordre initial.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Insérer une valeur dans un préfixe trié sans perdre une occurrence.
- Le préfixe déjà traité reste trié ; key protège la prochaine valeur à insérer.
- Les décalages stricts préservent l'ordre relatif des clés égales.

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

Partir d'un besoin

Comme des cartes que l'on range dans sa main, peut-on insérer chaque nouvelle valeur dans la partie déjà ordonnée ?

Définir la structure et les conventions

Un **tri** doit produire un ordre tout en conservant chaque occurrence des valeurs initiales. Le **tri par insertion** maintient un préfixe trié, puis insère la valeur suivante en décalant vers la droite les valeurs strictement plus grandes.

L'invariant avant le tour index est que les positions précédentes sont triées et contiennent exactement les valeurs initiales de ce préfixe. La variable key conserve la valeur à insérer avant que sa case soit écrasée. position recule jusqu'au bon emplacement. Le court-circuit protège l'accès quand position dépasse le début.

Un tri est **stable** s'il conserve l'ordre relatif des éléments ayant la même clé. Décaler seulement les valeurs strictement supérieures laisse la nouvelle occurrence après les égales déjà présentes. Le contrat Python modifie la liste reçue ; MATLAB renvoie son vecteur trié sans modifier la variable de l'appelant.

Suivre les états

Préfixes du tri de [3, 1, 2]

Valeur insérée	Suite après insertion
3 (initialisation)	[3, 1, 2]
1	[1, 3, 2]
2	[1, 2, 3]

Comparer les réalisations

La sortie vaut [1, 2, 3]. Python ne renvoie pas une nouvelle liste : la fonction modifie values. MATLAB, dans insertion_sort_demo.m, conserve original et affecte le résultat à values. Les deux contrats doivent être testés séparément.

Pseudo-code

```
// example.pseudo
// insertion_sort
FOR index FROM 1 TO length(values) - 1 INCLUSIVE DO
    key <- values[index]
    position <- index - 1
    WHILE (position >= 0) AND (values[position] > key) DO
        values[position + 1] <- values[position]
        position <- position - 1
    END WHILE
    values[position + 1] <- key
END FOR
```

Python

```
# insertion_sort.py
def insertion_sort(values):
    for index in range(1, len(values)):
        key = values[index]
        position = index - 1
        while position >= 0 and values[position] > key:
            values[position + 1] = values[position]
            position = position - 1
        values[position + 1] = key
values = [3, 1, 2]
insertion_sort(values)
print(values)
```

MATLAB

```
% insertion_sort_demo.m
original = [3, 1, 2];
values = insertion_sort(original);
disp(values);
function values = insertion_sort(values)
    for index = 2:numel(values)
        key = values(index);
        position = index - 1;
        while position >= 1 && values(position) > key
            values(position + 1) = values(position);
            position = position - 1;
        end
        values(position + 1) = key;
    end
end
```

Justifier et compter

Une entrée déjà triée exige Θ(n) comparaisons et aucun décalage. L'ordre inverse exige n(n - 1)/2 décalages : Θ(n²). Le travail local Python utilise Θ(1) mémoire auxiliaire. MATLAB peut devoir conserver une copie du vecteur pour préserver l'entrée ; ne lui attribuez pas la même garantie de stockage.

Prévoir puis expérimenter

Tracez [1, 2, 3], [3, 2, 1], [2, 2, 1], le vide et une seule valeur. Pour deux éléments étiquetés de même clé, vérifiez qu'un décalage strict conserve leur ordre.

Exercice corrigé

▼ **Que perd-on en remplaçant > par ≥ dans le décalage ?**

La stabilité : une nouvelle occurrence peut passer devant une occurrence égale déjà dans le préfixe, bien que les clés soient toujours triées.

À retenir

- Le préfixe déjà traité reste trié ; key protège la prochaine valeur à insérer.
- Les décalages stricts préservent l'ordre relatif des clés égales.
- Le meilleur cas est linéaire, le pire quadratique ; les contrats de mutation Python et MATLAB diffèrent.

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