STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 24

Construire un index de fréquences vérifiable

Assemblez dictionnaire, tri et contrats dans un classement de codes. Testez occurrences, égalités, entrée vide et conservation des données.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Assembler comptage, classement et vérifications d'occurrences.
- Séparez comptage, classement et critères de départage.
- Vérifiez ordre, occurrences, conservation de l'entrée et somme des compteurs.

Prérequis. Avoir étudié les chapitres DSA précédents ; connaître recherche, tris et raisonnement sur les coûts.

Partir d'un besoin

Un journal contient des codes d'événements. Il faut afficher chaque code distinct avec son nombre d'occurrences, par fréquence décroissante puis par code croissant en cas d'égalité. Quelles structures rendent chaque étape explicite ?

Définir la structure et les conventions

Le **contrat** accepte une suite de codes textuels parmi A, B et C, afin de fixer un ordre de comparaison identique dans les deux langages. La sortie contient chaque code une seule fois avec son compteur, classée selon deux critères. L'entrée ne doit pas être modifiée ; une entrée vide donne un classement vide.

Décomposez en deux étapes : un index associatif compte les occurrences, puis un tri ordonne les entrées de cet index. L'invariant du comptage est : après chaque événement, la somme des compteurs est égale au nombre d'événements déjà lus. Le tri ne doit changer ni les clés ni leurs fréquences.

Python trie les couples (code, compteur) avec la clé (-compteur, code) : le signe moins transforme l'ordre décroissant des comptes en ordre croissant de clés. Les couples sont comparés d'abord sur leur première composante, puis sur la seconde en cas d'égalité. MATLAB calcule les comptes des codes distincts puis trie la matrice [-compte, rang alphabétique]. Les cellules stockent les textes ; strcmp compare leur contenu.

Suivre les états

Classement de B, A, B, C, A

Étape	Résultat
Comptage	A:2, B:2, C:1
Tri décroissant par compte	A et B avant C
Départage alphabétique	A:2, B:2, C:1

Comparer les réalisations

Le classement est A:2, B:2, C:1. La fonction MATLAB renvoie séparément les étiquettes et les comptes. counts(:) transforme le vecteur en colonne ; l'apostrophe transpose le vecteur numérique des rangs ; ~ ignore la matrice triée pour ne conserver que les indices de permutation. Enregistrez frequency_index_demo.m avec sa fonction locale.

Pseudo-code

```
// example.pseudo
// frequency_index
// Count each code, then order entries by decreasing count.
// Break ties by increasing code.
// Check: one output per distinct code, total counts = input length.
// Preserve the original input sequence.
```

Python

```python
# frequency_index.py
def frequency_index(codes):
    counts = {}
    for code in codes:
        counts[code] = counts.get(code, 0) + 1
    return sorted(counts.items(), key=lambda entry: (-entry[1], entry[0]))
codes = ["B", "A", "B", "C", "A"]
ranking = frequency_index(codes)
print(ranking)
```

MATLAB

```
% frequency_index_demo.m
codes = {'B', 'A', 'B', 'C', 'A'};
[labels, counts] = frequency_index(codes);
disp(labels);
disp(counts);
function [labels, counts] = frequency_index(codes)
    labels = unique(codes);
    counts = zeros(1, numel(labels));
    for index = 1:numel(labels)
        counts(index) = sum(strcmp(codes, labels{index}));
    end
    k = numel(labels);
    keys = [-counts(:), (1:k)'];
    [~, order] = sortrows(keys, [1, 2]);
    labels = labels(order);
    counts = counts(order);
end
```

Justifier et compter

Pour n événements et k codes distincts, le dictionnaire Python donne un comptage O(n) attendu sous les hypothèses de hachage, suivi d'un tri O(k log k), avec O(k) mémoire supplémentaire. La version MATLAB choisit une autre réalisation : elle compare chaque code distinct aux n entrées, donc O(nk) pour cette étape, plus le coût de unique et du tri. Elle sert de réalisation lisible, pas de preuve que les deux codes ont la même complexité.

Prévoir puis expérimenter

Écrivez les sorties attendues avant de tester : entrée vide ; un seul événement ; cinq fois A ; A, B, C en ordre inverse ; fréquences toutes égales. Pour une extension à d'autres alphabets ou à du texte Unicode, définissez d'abord la règle de comparaison des codes.

Exercice corrigé

▼ Quelles propriétés faut-il vérifier en plus du classement ?

Chaque code doit apparaître exactement une fois, chaque compteur doit être correct, leur somme doit valoir n et l'entrée doit être conservée. Un classement bien ordonné peut quand même avoir perdu des occurrences.

À retenir

- Séparez comptage, classement et critères de départage.
- Vérifiez ordre, occurrences, conservation de l'entrée et somme des compteurs.
- Deux réalisations du même contrat peuvent avoir des complexités différentes.

Version web : https://www.clearcode.fr/courses/programming/dsa/strategies/project