Parcours séquentiels: résumé

qkzk

2025-06-21

Parcours séquentiels ou méthode par balayage

On emploie un *parcours séquentiel* ou un algorithme *par balayage* si on peut répondre à une question en examinant une fois chaque valeur d'une collection.

Exemples

- Parmi mes élèves, qui a déjà vu le film Titanic ? Pour connaître la réponse, il suffit d'interroger chaque élève et poser la question "As-tu ... ?". Je garde une trace de ceux qui répondent "oui". C'est un parcours séquentiel.
- Parmi mes élèves, en ai-je deux qui se ressemblent beaucoup ? Pour chaque élèves, **j'examine chaque autre élève** et les compare. Alors je ne peux répondre en un seul parcours. Ce n'est pas un parcours séquentiel.
- Cette liste est-elle triée par ordre croissant ? J'examine chaque terme selon son indice et le compare au suivant. Si un couple n'est pas rangé dans le bon ordre, je réponds que non. Si j'arrive au bout, alors oui. Parcours séquentiel.
- Trier cette liste par odre croissant. Quelle que soit la méthode employée, je dois à un moment, reparcourir une partie de la collection. Impossible de répondre par un parcours séquentiel.

Situations où l'indice n'est pas nécessaire :

Calculer la moyenne d'une `list` de valeurs en n'employant ni `sum` ni `len`

- Se résout par parcours séquentiel,
- Ne nécessite aucun indice.

On ajoute les éléments un par un et on compte leur effectif. On renvoie le quotient.

Il faut donc commencer par initialiser leurs valeurs à 0.

Ce problème entre dans une catégorie plus restreinte, d'algorithmes de *cumul* :

- On crée un objet vide/égal à 0/égal à 1 (selon l'opération employée),
- On parcourt une fois la collection avec `for elt in tab:`
- On met à jour l'objet créée plus tôt,
- On renvoie cet objet ou un calcul fait avec lui.

```python
def moyenne(tab: list) -> float:
    """
    Calcule la moyenne de tab, une liste non vide d'entiers et renvoie sa moyenne
    N'emploie ni sum ni len
    """
    total = 0
    longueur = 0
    for elt in tab:
        total = total + elt
        longueur = longueur + 1
    return total / longueur
```

Déterminer l'élément le plus petit d'une collection non vide d'entiers sans utiliser `min`

Mêmes remarques.

On initialise `plus_petit` avec la première valeur. On parcourt la collection et, à chaque tour, on compare l'élément courant avec `mini`. Si nécessaire on met à jour.

```
def mini(tab: list) -> int:
    """
    Renvoie l'élément le plus petit de tab, supposée non vide
    N'utilise pas min
    """
    plus_petit = tab[0] # !!! on a initialisé avec la première valeur, tab étant non vide
    for elt in tab:
        if elt < plus_petit:
            plus_petit = elt # on ne met à jour que lorsqu'un meilleur candidat se présente
    return elt
```

Extraire les voyelles d'un mot

On reçoit un mot, on fabrique la collection de ses voyelles (avec répétition si nécessaire)

```
>>> extraire_voyelles("bonjour")
["o", "o", "u"]
```

- on crée une liste pour recevoir les voyelles,
- on parcourt chaque lettre et, si c'est une voyelle, on l'ajoute à la liste,
- renvoyer ça.

```
VOYELLES = "aeiouy"

def extraire_voyelles(mot: str) -> list:
    coll = []
    for lettre in mot:
        if lettres in VOYELLES:
            coll.append(lettre)
    return coll
```

Et bien d'autres...

Situations où l'indice est indipensable

Alors on emploie `for i in range(len(...)):`

Si l'énoncé évoque *une position* dans la collection, il faut sûrement l'indice.
Ce n'est pas suffisant comme critère, comme on va le voir.

Déterminer l'indice du jour où la temperature est la plus basse

On reçoit une collection de température, une par jour et on veut savoir quel est le jour où la température était la plus basse. Pas le choix, on demande explicitement un indice.

```
def jour_de_temperature_minimale(temperatures: list) -> int:
    """
    Renvoie l'indice du jour de `temperatures` où la valeur est minimale
    temperatures est supposée non vide
    """
    ind_mini = 0
    for i in range(len(temperatures)):
        if temperatures[i] < temperatures[ind_mini]:
            ind_mini = i
    return ind_mini
```

```
>>> jour_de_temperature_minimale([3, 10, 2, 7, 14, 18])
2
```

Déterminer la dernière occurence d'un élément

Attention aux nuances : ici on demande la **dernière** occurence... Si on demande la **première** occurence, l'algorithme change.

Même approche, on compare élément par élément. Lorsqu'on a terminé, on renvoie.

```
def derniere_occurence(collection: list, elt: int) -> int:
    """
    Renvoie le dernier indice de `elt` dans `collection`.
    -1 si elt ne figure pas.
    """
    occ = -1 # initialiser avec la valeur par défaut
    for i in range(len(collection)):
        if collection[i] == elt:
            occ = i
    return occ
```

Déterminer la première occurence d'un élément

Attention aux nuances : ici on demande la **première** occurence...

Cette fois, grande différence : lorsqu'on a trouvé on peut s'arrêter !

```
def premiere_occurence(collection: list, elt: int) -> int:
    """
    Renvoie le premier indice de `elt` dans `collection`.
    -1 si elt ne figure pas.
    """
    occ = -1 # initialiser avec la valeur par défaut
    for i in range(len(collection)):
        if collection[i] == elt:
            return i # la différence est là !!!
    return occ
```

Déterminer si une collection est triée

On reçoit une collection homogène d'éléments comparables deux à deux. On veut savoir si elle est triée.

On doit comparer un élément et son suivant, il faut s'arrêter un pas avant le dernier élément

```
>>> est_trie([1, 3, 5])
True
>>> est_trie([5, 1, 3])
False
```

```
def est_trie(collection: list) -> bool:
    """Vrai ssi la collection est triée"""
    for i in range(len(collection) - 1): # le moins 1 est fondamental
        if collection[i] > collection[i + 1]:
            return False
    return True
```

Coût

Généralement, le coût d'un parcours séquentiel est *linéaire.* Cela signifie que la durée d'exécution est proportionnelle à la taille du tableau.

Situation : "Il faut 1 secondes pour exécuter ce programme si un tableau de taille n."

Quelle devrait être la durée d'exécution sur un tableau de taille $10n$, $100n$?

Coût	Notation	Durée pour $10n$ éléments	Pour $100n$ éléments	Exemple d'algo
Constant	$O(1)$	1 seconde	1 seconde	"A-t-on `x` dans un dictionnaire ?"
Log.	$O(\log n)$	$\log(10)$ s ≈ 2.30 s	$2\log(10) \approx 4.60$ s	Recherche dichotomique
Linéaire	$O(n)$	10 secondes	100 secondes	Parcours séquentiel
Quadratique	$O(n^2)$	$10^2 = 100$ secondes	$\approx 3h$	Tri par sélection, par insertion
Exponentiel	$O(2^n)$	$2^{10} = 1024$ secondes	$2^{100} \approx 4 \times 10^{22}$ ans	Déterminer le chemin le plus court passant par n points