CHAPITRE 23

Parcours séquentiel d'un tableau

Dans presque tous les calculs, une grande variété d'arrangements pour la succession des processus est possible, et plusieurs considérations doivent influencer la sélection. L'une des considérations essentielle est de choisir l'arrangement qui tend à réduire au minimum le temps de calcul.

Ada Lovelace

Thème : Algorithmique

Contenus	**Capacités attendues**	**Commentaires**
Parcours séquentiel d'un tableau	Écrire un algorithme de recherche d'une occurrence sur des valeurs de type quelconque. Écrire un algorithme de recherche d'un extremum, de calcul d'une moyenne.	On montre que le coût est linéaire

I. Coût en temps d'un algorithme

Le temps d'exécution d'un algorithme (ou de son implémentation en Python) est le produit du nombre d'instructions qui seront exécutées tout au long de son déroulement par le temps d'exécution d'une instruction.

$$\text{temps d'exécution} = \text{nombre d'instructions} \times \text{temps d'exécution d'une instruction}$$

Activité 1.

On considère 2 algorithmes renvoyant le même résultat à partir du même nombre n de données.

- L'algorithme A1 effectue n^2 successions d'instructions de base.
- L'algorithme A2 effectue $10 \times n$ successions d'instructions de base.

On utilise deux machines aux capacités différentes :

- La machine M1 effectue 1 000 d'instructions de base par seconde (Intel 80286 20 MHz - 1982)
- La machine M2 effectue 100 000 000 d'instructions de base par seconde (Intel i7 3,5 GHz - 2010)

1. Déterminez le temps de calcul pour chaque machine et pour chaque algorithme si le nombre de données à traiter est n = 10, n = 1 000, n = 1 000 000 et n = 1 000 000 000. (Complétez les tableaux page 2)

2. L'ordre de grandeur (la puissance de 10) du nombre de secondes dans une journée est 10^5 ; L'ordre de grandeur du nombre de secondes dans un mois est 10^6 ; L'ordre de grandeur du nombre de secondes dans une année est 10^7.
Toutes ces données peuvent-elles être traitées dans un temps raisonnable ?

Pour l'algorithme A1 :

n =	10	1 000	1 000 000	1 000 000 000
M1				
M2				

Pour l'algorithme A2 :

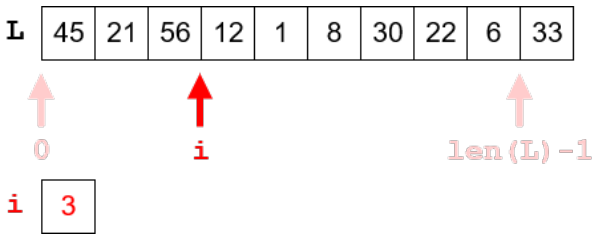
n =	10	1 000	1 000 000	1 000 000 000
M1				
M2				

Conclusion : une machine puissante n'est rien sans un bon algorithme.

II. Les tableaux en algorithmique

II. 1. Structure d'un tableau

Un tableau est une structure de données représentant une séquence finie d'éléments auxquels on peut accéder par leur position (appelé aussi indice ou index) dans la séquence. On parle aussi de tableau indexé.



On ne s'intéresse qu'aux tableaux présentant des données du même type.

Dans le langage Python, les tableaux sont implémentés par des listes.

Exemple 1 (Quelques exemples de tableaux en Python)

```
liste_entiers = [1, 2, 3, 4, 5]
liste_reels = [0.0, 1.0, 2.0, 3.5]
lliste_caracteres = ["toi", "moi", "nous"]
```

II. 2. Parcours séquentiel d'un tableau

Déf. 1 Le parcours séquentiel d'un tableau consiste à faire évoluer un indice, et ainsi à déplacer un pointeur sur chaque case du tableau. La première valeur de l'indice est 0 (le début du tableau). La dernière valeur de l'indice est (longueur - 1).

Dans un pseudo-code, on écrit :

Algorithme : parcours(tableau)

```
/* algorithme qui renvoie le contenu d'un tableau                    */
Entrées : tableau : un tableau
début
    pour indice variant de 0 à longueur exclue faire
        ...
    fin pour
fin
```

Dans la langage Python, le parcours d'un tableau s'implémente par :

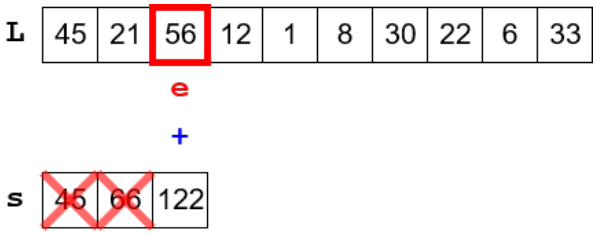
```
for i in range(0, len(tab), 1):
        <instructions>
```

ou bien par :

```
for element in tab:
    <instructions>
```

III. Moyenne des éléments d'un tableau

Pour calculer la moyenne des valeurs d'une liste, il faut calculer la somme de ces valeurs et diviser par le nombre de valeurs contenues dans la liste.
Pour calculer la somme on initialise une variable *somme* à zéro, et on itère sur l'ensemble des éléments de la liste. A chaque itération, on ajoute à *somme* la valeur de *element* .



1. Quelle est la moyenne du tableau proposé ci-dessus ?

2. Complétez l'algorithme en pseudo-code permettant de calculer la moyenne des éléments d'un tableau.

Algorithme : moyenne(tableau)

```
/* Algorithme qui renvoie la moyenne des éléments d'un tableau              */
Entrées : tableau : tableau contenant des nombres
Sorties : moyenne : moyenne des éléments du tableau
1 début
2 |   ..................;
3 |   pour ..................................................... faire
4 |   |   ..................
5 |   fin pour
6 |   ..................................................;
7 |   retourner moyenne
8 fin
```

3. Réalisez l'historique d'exécution pour les tableaux ci-dessous :

45	21

45	21	56

45	21	56	12

45	21	56	12	1

45	21	56	12	1	8

4. Pour ces différents historiques d'exécution, comptez le nombre d'instructions successives. Complétez alors le tableau suivant :

Nombre d'éléments du tableau (n)					
Nombre d'instructions exécutées (C)					

5. Représentez le nombre d'instructions en fonction du nombre d'éléments du tableau. Quelle est la forme du graphique obtenu ?

   Donnez l'équation mathématique reliant C à n :

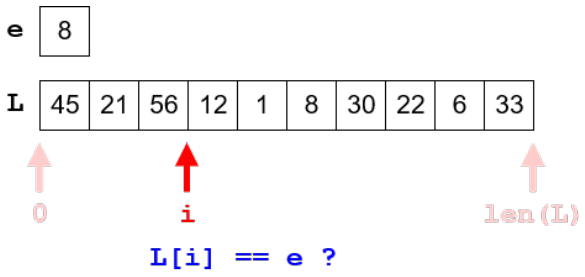
6. Combien d'instructions seraient exécutées pour un tableau contenant :
   (a) 100 éléments ?
   (b) 1 000 éléments ?
   (c) 1 000 000 éléments ?
7. On dit que le coût de cet algorithme est linéaire. Justifiez cette expression.

IV. Recherche d'un élément dans un tableau

Il s'agit de déterminer la position (l'indice) d'un élément x prétendument présent dans un tableau.

Un "pointeur" i, initialisé à zéro, permet d'accéder aux éléments du tableau. À chaque itération, on compare l'élément d'indice i du tableau avec l'élément x recherché. S'ils sont égaux, on renvoie la valeur de l'indice et les itérations cessent. Dans le cas contraire, on incrémente i et on fait

une nouvelle itération. Si l'élément x n'est pas dans le tableau, le pointeur va jusqu'au bout du tableau. Dans ce cas, la valeur retournée est -1.



1. Dans l'exemple proposé ci-dessus, quelle sera la valeur renvoyée ?
2. Proposez un algorithme en pseudo-code (sur le même modèle que le précédent) permettant de rechercher l'élément x dans un tableau.
3. On souhaite réaliser les recherches ci-dessous :

   On recherche 56 dans | 45 | 21 | 56 |

   On recherche 12 dans | 45 | 21 | 56 | 12 |

   On recherche 1 dans | 45 | 21 | 56 | 12 | 1 |

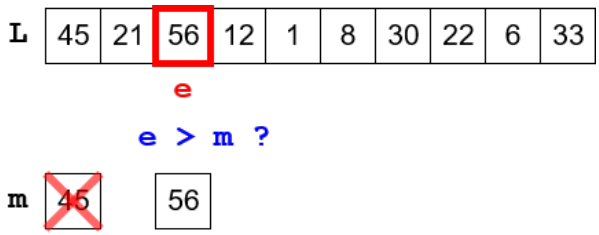
   On recherche 8 dans | 45 | 21 | 56 | 12 | 1 | 8 |

   En reproduisant l'exemple précédent (moyenne des éléments d'un tableau), indiquez si le coût de l'algorithme est linéaire ou non-linéaire.

V. Recherche du maximum dans un tableau

Il s'agit de déterminer l'élément de plus grande valeur présent dans un tableau.

Pour connaître la valeur maximum des valeurs contenues dans un tableau, il faut évaluer tour à tour tous les éléments du tableau. La valeur maximum est mémorisée dans une variable *maximum*, initialisée avec le premier élément du tableau. 'A chaque étape, on compare l'élément du tableau au contenu de la variable *maximum*. Si l'élément est supérieur à *maximum*, *maximum* prend alors la valeur de cet élément.



1. Dans l'exemple proposé ci-dessus, quelle sera la valeur renvoyée ?

2. On initialise la variable *maximum* avec la première valeur du tableau. Justifier cette valeur.

3. Proposez un algorithme en pseudo-code (sur le même modèle que le précédent) permettant de rechercher l'élément de plus grande valeur dans un tableau.

4. On souhaite connaître le maximum de chacun des tableaux ci-dessous :

45	21	56

45	21	12	56

45	21	12	1	56

45	21	12	1	8	56

   En reproduitant l'exemple précédent, indiquez si le coût de l'algorithme est linéaire ou non-linéaire.

VI. Exercices

Pour chacune des fonctions que vous allez écrire, il existe une fonction *test_ ma_ fonction()*. Exécutez cette fonction dans la console pour vérifier votre travail.

Exercice 1.

1. Complétez la fonction *moyenne*() correspondant à l'algorithme de calcul de la moyenne des éléments d'un tableau, et vérifiez votre code avec la fonction de test correspondante.
2. La fonction `mesure_temps_moyenne()` renvoie le temps d'exécution de la fonction *moyenne()* pour n éléments dans le tableau.
   a. Dans la console Python, exécutez l'instruction `help(mesure_temps_moyenne)` pour connaître sa documentation.
   b. Quels sont les arguments de cette fonction ?
   c. Quelle instruction faut-il saisir en console pour connaître le temps d'éxécution de la fonction *moyenne* pour un tableau de *10* éléments ?
   d. Quel est le temps d'éxécution de la fonction *moyenne* pour un tableau de *10* éléments ?
   e. Par combien est multiplié le temps d'exécution si le nombre d'éléments est multiplié par 10 ? Par 100 ? Par 100 000 ?

Exercice 2.

Complétez la fonction *recherche*() correspondant à l'algorithme de recherche d'un élément dans un tableau, et vérifiez votre code avec la fonction de test correspondante.

Exercice 3.

Complétez la fonction *maximum*() correspondant à l'algorithme de recherche du maximum dans un tableau, et vérifiez votre code avec la fonction de test correspondante.

Exercice 4.

Complétez la fonction *separer*() permettant, à partir d'un tableau de nombres, d'obtenir deux tableaux. Le premier comporte les nombres inférieurs ou égaux à un nombre donné, le second les nombres qui lui sont strictement supérieurs.
`separer([45, 21, 56 ,12, 1, 8, 30, 22, 6, 33], 30)` doit renvoyer :
[21, 12, 1, 8, 30, 22, 6], [45, 56, 33]

Exercice 5.
Complétez la fonction *plus_proche*() permettant de rechercher la plus proche valeur d'un nombre dans un tableau. `plus_proche([45, 21, 56 ,12, 1, 8, 30, 22, 6, 33], 20)` doit renvoyer 21

Exercice 6.
Complétez la fonction *compter_position*() permettant de compter le nombre d'occurrences d'une lettre dans une chaîne de caractères et de donner leurs positions sous la forme d'un tableau.
`compter_position("Numérique et Sciences Informatiques !", 'm')` doit renvoyer `2, [2, 27]`.

Exercice 7.
Complétez la fonction *compter_tout*() permettant d'obtenir les nombres d'occurrences de toutes les lettres d'une chaîne de caractères, sous la forme d'un dictionnaire {lettre : nbre occurences}.
`compter_tout("Numérique et Sciences Informatiques !")` doit renvoyer : {'N' : 1, 'u' : 3, 'm' : 2, 'é' : 1, 'r' : 2, 'i' : 3, 'q' : 2, 'e' : 5, ' ' : 4, 't' : 2, 'S' : 1, 'c' : 2, 'n' : 2, 's' : 2, 'I' : 1, 'f' : 1, 'o' : 1, 'a' : 1, '!' : 1}