19 et 21 sept. 2018 | # Algorithmes de tri | I2 — Promo 122

A. Introduction

Trier des données consiste à les ranger en ordre croissant ou décroissant. Il existe plusieurs méthodes pour classer des éléments. Si la méthode utilisée importe peu pour trier un petit nombre d'éléments, il n'en est pas de même pour un nombre élevé. Il est donc intéressant de comparer différents algorithmes de tri afin de savoir quand les utiliser suivant la situation.
Une opération de tri s'exécute en un **temps fini** et mobilise des **ressources mémoires** ; ces deux paramètres se dénomment **complexité temporelle** et **complexité spatiale**. Au début de l'informatique, ces deux critères ont eu une importance à peu près égale pour comparer l'efficacité d'un algorithme. Depuis que l'on peut acquérir des ordinateurs possédant plus de 1 Go de mémoire vive à faible coût, le temps d'exécution est devenu l'élément qualitatif le plus perceptible par les utilisateurs.
Les algorithmes de tri sont basés sur des comparaisons successives entre les données et leur complexité a le même ordre de grandeur que le nombre de comparaisons réalisées.

B. Tri basique

1. Présentation

Le principe du tri basique est le suivant :

- On parcourt la liste à trier. Si on trouve 2 éléments consécutifs qui ne sont pas dans le bon ordre, on les échange et on repart au début de la liste.
- Quand on est arrivé à la fin de la liste, elle est triée.

Exercice 1 Effectuer l'algorithme « à la main » sur la liste [24,15,53,34,0,90].
Compter le nombre de comparaisons et le nombre d'échanges.

Exercice 2 Écrire le pseudo-code de l'algorithme du tri basique.

Exercice 3 Pour une liste de 6 termes :

- Combien y a-t-il de comparaisons au minimum ? Pour quelles listes ?
- Combien y a-t-il de comparaisons au maximum ? Pour quelles listes ?
- Combien y a-t-il d'échanges au minimum ? Pour quelles listes ?
- Combien y a-t-il d'échanges au maximum ? Pour quelles listes ?

Exercice 4 Pour une liste de n termes :

- Combien y a-t-il de comparaisons au minimum ? Combien d'échanges au minimum ?
- Combien y a-t-il de comparaisons au maximum ? Combien d'échanges au maximum ?

Exercice 5 Modifier votre pseudo-code pour compter le nombre de comparaisons et le nombre d'échanges effectués.

2. Implémentation en Python

Exercice 6 Coder la fonction **tri_basique(L)** qui retourne la liste **L** triée en utilisant la technique du tri basique, le nombre de comparaisons et le nombre d'échanges effectués.

Exercice 7 Coder la fonction **existe(L,e)** qui renvoie *True* si la liste **L** contient l'élément **e** ou *False* dans le cas contraire.

Exercice 8 Coder la fonction **creation_liste(n,a,b)** qui renvoie une liste de **n** entiers au hasard compris entre **a** et **b sans doublons**.
Utiliser la méthode *randint* de la bibliothèque *random*.

```
import random

# Générer de manière aléatoire un entier compris entre a et b
x=random.randint(a,b)
```

Exercice 9 Compléter le tableau ci-dessous en générant 100 listes aléatoires de 10, 100 puis 250 entiers compris entre 1 et 1000 afin d'évaluer en moyenne le nombre de comparaisons, le nombre d'échanges et le temps d'exécution du tri basique.
Utiliser la méthode *clock* de la bibliothèque *time*.

```
import time

t1=time.clock()

# Code de l'algorithme à évaluer

t2=time.clock()

# Affichage du temps d'exécution
print(t2-t1)
```

Nombre d'éléments	Nombre de comparaisons	Nombre d'échanges	Temps d'exécution
10			
100			
250			

C. Tri à bulles ou tri par propagation

1. Présentation

Le principe du tri à bulles est le suivant :

- On parcourt une première fois la liste à trier. Si on trouve 2 éléments consécutifs qui ne sont pas dans le bon ordre, on les échange et on continue. Après cette étape, le plus grand élément est repoussé à la fin du tableau telle une bulle d'air qui remonte à la surface de l'eau.
- On reprend alors la même chose jusqu'à l'avant-dernier élément. Les 2 plus grands sont alors repoussés, dans l'ordre, en fin de liste.
- On recommence ainsi jusqu'après avoir trié les 2 premiers éléments.

Exercice 10 Effectuer l'algorithme « à la main » sur la liste [24,15,53,34,0,90]
Compter le nombre de comparaisons, le nombre d'échanges. Comparer ces nombres au tri basique.

Exercice 11 Écrire le pseudo-code de cet algorithme de tri tout en comptabilisant le nombre de comparaisons et le nombre d'échanges effectués.

Exercice 12 Pour un tableau de n termes :

- Combien y a-t-il de comparaisons ? Remarquer que ça ne dépend pas du tableau.
- Combien y a-t-il d'échanges au minimum ? Pour quelles listes ? Comparer au tri basique.
- Combien y a-t-il d'échanges au maximum ? Pour quelles listes ? Comparer au tri basique.
- Y a-t-il des cas où le tri basique est plus rapide que le tri à bulles ?

2. Implémentation en Python

Exercice 13 Coder la fonction **tri_bulles(L)** qui retourne la liste **L** triée en utilisant la technique du tri à bulles, le nombre de comparaisons et le nombre d'échanges effectués.

Exercice 14 Compléter le tableau ci-dessous en générant 100 listes aléatoires de 10, 100 puis 250 entiers compris entre 1 et 1000 afin d'évaluer en moyenne le nombre de comparaisons, le nombre d'échanges et le temps d'exécution du tri à bulles.

Nombre d'éléments	Nombre de comparaisons	Nombre d'échanges	Temps d'exécution
10			
100			
250			

Remarquez que l'ordre de grandeur du nombre moyen de comparaisons pour le tri à bulles est $\frac{n^2}{2}$.