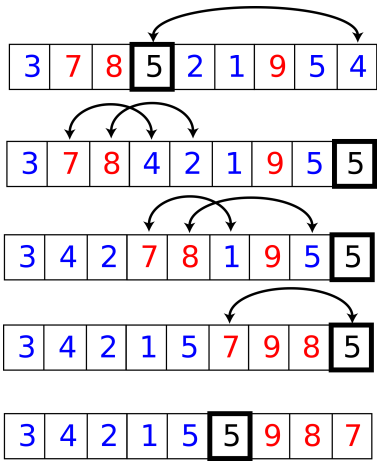
Chapitre 1

Algorithmes de tri



On désigne par *tri* l'opération consistant à ordonner un ensemble d'éléments en fonction de clés sur lesquelles est définie une relation d'ordre.

Les algorithmes de tri ont une grande importance pratique. Ils sont fondamentaux dans certains domaines, comme l'informatique de gestion où l'on tri de manière quasi-systématique des données avant de les utiliser.

L'étude du tri est également intéressante en elle-même car il s'agit sans doute du domaine de l'algorithmique qui a été le plus étudié et qui a conduit à des résultats remarquables sur la construction d'algorithmes et l'étude de leur complexité.

1.1 Tri par insertion

Le tri par insertion est le tri le plus naïf : c'est le tri du joueur de cartes. Le principe est simple.
Considérons une liste L à trier.

- Sélectionner l'élément d'indice 1 de la liste, et le permuter avec l'élément à sa gauche tant que l'élément à sa gauche est strictement supérieur (s'arrêter si on arrive tout à gauche de la liste)
- Sélectionner l'élément d'indice 2 de la liste, et répéter les mêmes opérations
- Continuer jusqu'au dernier élément de la liste

Exemple. Trions la liste $L = [9,4,2,5,1]$ par insertion.

- Étape 1 : l'élément d'indice 1 est 4. L'élément à sa gauche (9) étant supérieur, on permute 4 et 9.

9	4	2	5	1	$\Longrightarrow$	4	9	2	5	1

  Les nombres 4 et 9 ont été **permutés**, c'est à dire qu'on a échangé leurs places.

- Étape 2 : l'élément d'indice 2 est 2. Étant plus petit que les éléments à sa gauche, il se retrouve tout à gauche de la liste.

4	9	2	5	1	$\Longrightarrow$	4	2	9	5	1	$\Longrightarrow$	2	4	9	5	1

  Cette fois-ci, il y a eu 2 permutations : 2 et 9 puis 2 et 4.

- Étape 3 : c'est au tour de 5 : il est déplacé entre le 4 et le 9.

2	4	9	5	1	$\Longrightarrow$	2	4	5	9	1

- Étape 4 : dernière étape. 1 étant plus petit que tous les autres termes, il se retrouve au début de la liste après plusieurs permutations successives.

2	4	5	9	1	$\Longrightarrow$	2	4	5	1	9	$\Longrightarrow$	2	4	1	5	9	$\Longrightarrow$	2	1	4	5	9

  $\Longrightarrow$

1	2	4	5	9

La liste est triée !

1	2	4	5	9

Compréhension

Question 01 Combien de permutations ont été réalisées au total dans l'exemple précédent ?

Question 02 On considère une liste de longueur 5.

1. Quel serait, dans le **meilleur des cas**, le nombre de permutations à effectuer pour trier la liste ? Donner un exemple.
2. Quel serait, dans le **pire des cas**, le nombre de permutations à effectuer pour trier la liste ? Donner un exemple.

Question 03 On considère une liste de longueur $n \geqslant 2$.

1. Quel serait, dans le meilleur des cas, le nombre minimum de permutations à effectuer pour trier la liste ?
2. On peut démontrer que dans le pire des cas, le nombre de permutations à effectuer est égal à $\dfrac{n(n-1)}{2}$.
   (a) Pour une liste de longueur 10, combien de permutations sont à effectuer dans le pire des cas ?
   (b) Vérifier que le tri d'une liste de longueur 200 nécessite environ 4 fois plus de permutations que le tri d'une liste de longueur 100.
   (c) Vérifier que le tri d'une liste de longueur 300 nécessite environ 9 fois plus de permutations que le tri d'une liste de longueur 100.

☞ *Lorsqu'on utilise un algorithme, il est important de connaître sa* ***complexité****. Dans le cas de notre algorithme de tri par insertion, le nombre d'****opérations élémentaires*** *(ici les permutations mais aussi les opérations de comparaison entre les termes de la liste) est proportionnel à n^2, où n est la taille de la liste à trier.*

On dit alors que la complexité de l'algorithme du tri par insertion est ***quadratique****, car lorsqu'on multiplie la taille de la liste par n, le nombre d'opérations élémentaires est multiplié par n^2. On parle alors de « complexité en $O(n^2)$ ».*

En pratique, cela signifie que si le tri d'une liste de longueur n prend 1 seconde, alors le tri d'une liste de longueur $2n$ prendra 4 secondes !

Programmation

Question 04 On considère la liste `L = [3,7,8,4,0,5]`.

1. Quelle(s) instruction(s) Python permet(tent) de permuter 7 et 0 ? 8 et 5 ?
2. Compléter le programme suivant afin qu'il déplace 4 à la « bonne position », conformément à l'algorithme de tri par insertion, c'est à dire en permutant successivement les éléments de la liste :

```
1  L = [3,7,8,4,0,5]
2
3  i = ... # Indice de l'élément à déplacer
4
5  # Tant que l'élément de gauche est supérieur
6  while L[...] < L[...] and i > 0:
7      ..... # Permutation des éléments
8      i = i - 1
```

Avant : `L = [3,7,8,4,0,5]`

Après : `L = [3,4,7,8,0,5]`

3. À quoi sert la condition `i > 0` ?

Question 05 Écrire une fonction `triInsertion(liste)` qui effectue un tri par insertion sur `liste`.

1.2 Tri par sélection

Le tri par sélection est un autre exemple de tri simple. Voici le principe :

- on recherche le plus petit élément de la liste, en parcourant tous les termes de la liste, et on le permute avec l'élément d'indice 0
- on recherche le plus petit élément de la liste, **en partant de l'indice 1**, et on le permute avec l'élément d'indice 1
- on continue de cette façon jusqu'à arriver au dernier élément de la liste

Exemple. Trions la liste $L = [5,4,2,9,1]$ par sélection.

- Étape 1 : on parcourt la liste à la recherche du plus petit élément.

5	4	2	9	1	$\Longrightarrow$	5	4	2	9	1	$\Longrightarrow$	5	4	2	9	1	$\Longrightarrow$	5	4	2	9	1

$\Longrightarrow$

5	4	2	9	1

Après parcours des 5 valeurs, il s'avère que le minimum est 1, situé à l'indice 4.
On permute alors les éléments d'indices 0 et 4, c'est à dire 5 et 1 :

5	4	2	9	1	$\Longrightarrow$	1	4	2	9	5

- Étape 2 : on parcourt la liste à partir de l'indice 1 à la recherche du plus petit élément.

1	4	2	9	5	$\Longrightarrow$	1	4	2	9	5	$\Longrightarrow$	1	4	2	9	5	$\Longrightarrow$	1	4	2	9	5

On ne parcourt ici que 4 valeurs, la première étant nécessairement plus petite que toutes les autres.
Ici, le minimum est 2. On l'échange avec 4 situé à l'indice 1 :

1	4	2	9	5	$\Longrightarrow$	1	2	4	9	5

- Étape 3 : on parcourt la liste à partir de l'indice 2 à la recherche du plus petit élément.

1	2	4	9	5	$\Longrightarrow$	1	2	4	9	5	$\Longrightarrow$	1	2	4	9	5

C'est 4. Il est déjà à la bonne place.

1	2	4	9	5

- Étape 4 : on parcourt la liste à partir de l'indice 3 à la recherche du plus petit élément.

1	2	4	9	5	$\Longrightarrow$	1	2	4	9	5

C'est 5. On le place à l'indice 3 :

1	2	4	5	9

La liste est triée !

1	2	4	5	9

Compréhension

Question 06 On considère l'exemple précédent de tri par sélection.

1. Donner le nombre total de permutations réalisées.
2. On appelle *opération de lecture* le fait d'accéder à une valeur de la liste pour lire sa valeur. Dans l'étape 1, on dénombre ainsi 5 opérations de lecture.

   Donner le nombre total d'opérations de lecture.

Question 07 On considère maintenant une liste de longueur 5 quelconque.

1. Donner, dans le meilleur et dans le pire des cas, le nombre de permutations réalisées au cours du tri par séléction de la liste. Donner un exemple de liste pour chaque cas.
2. Quel est le nombre total d'opérations de lecture ?
   Y a-t-il comme précédemment un « meilleur des cas » et un « pire des cas » ?

Question 08 Déterminer le nombre total d'opérations de lectures lors du tri par séléction :

1. d'une liste de longueur 4
2. d'une liste de longueur 6

Question 09 On peut montrer que pour une liste de longueur n, le nombre d'opérations de lecture est $\frac{n(n+1)}{2} - 1$.

1. Combien d'opérations de lectures sont nécessaires pour une liste de longueur $n = 100$? $n = 200$? $n = 300$?
2. Quelle est la complexité de l'algorithme de tri par sélection ?

Programmation

Question 10 Écrire une fonction `indice_minimum(liste)` qui retourne l'indice du minimum de la liste donnée.

```
>>> indice_minimum([9,4,2,5,1])
4
>>> indice_minimum([4,5,3,6,8])
2
```

Question 11 Écrire une fonction `indice_minimum(liste, indice_depart)` qui retourne l'indice du minimum de la liste donnée, en ne considérant que les éléments d'indices supérieurs à `indice_depart`.

```
>>> indice_minimum([9,4,2,5,1], 0)
4
>>> indice_minimum([1,6,3,2,9], 1)
3
```

Question 12 Programmer une fonction `triSelection(liste)` qui effectue un tri par sélection sur `liste`.