Les méthodes de tri -NSI
1

1.Historique :

« Betty Holberton (1917-2001) – Informaticienne

Betty Holberton est connue pour avoir été l'une des six programmatrices originelles de l'ENIAC, le premier ordinateur électronique.

Lors de son premier jour de cours à l'université de Pennsylvanie, son professeur de maths lui demande si elle ne serait pas mieux à la maison à élever des enfants. Cela la pousse à continuer ses études en mathématiques, tout en choisissant le journalisme comme option majeure (le journalisme était l'un des seuls domaines ouverts aux femmes en 1940).

Pendant la Seconde Guerre Mondiale, elle est embauchée par la Moore School of Electrical Engineering en tant que calculatrice. Elle est vite choisie pour être l'une des six programmatrices de l'ENIAC, avec Kay McNulty, Marlyn Wescoff, Ruth Lichterman, Betty Jean Jennings, et Fran Bilas. Elle y développe le premier code de construction, la première routine de tri et la première application logicielle.

Après la guerre, en 1959, elle est nommée directrice de la branche Recherche en Programmation du David Taylor Model Basin, laboratoire de mathématiques appliquées. Elle participe également à développer l'Univac (la division informatique de Remington Rand).

En 2001, celle qui est considérée comme la pionnière du logiciel par Donald E. Knuth meurt d'une maladie cardiaque à l'âge de 84 ans.

Source : Wikipedia ».

Pourquoi trier :

Le "**tri**" est l'opération consistant à ordonner un ensemble d'éléments ,

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.

A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm

Les méthodes de tri -NSI
2

Différents algorithmes de tri :

Tris stables	Tris non stables
Le tri par insertion.	Le tri par sélection.
Le tri bulle.	Le tri à peigne.
Le tri Shaker.	Le tri Shell.
Le tri Gnome.	Le tri maximier.
Le tri fusion.	Le tri rapide.

2. Stabilité des algorithmes de tri :

On dit qu'un algorithme de tri est stable s'il ne modifie pas l'ordre initial des clés identiques.

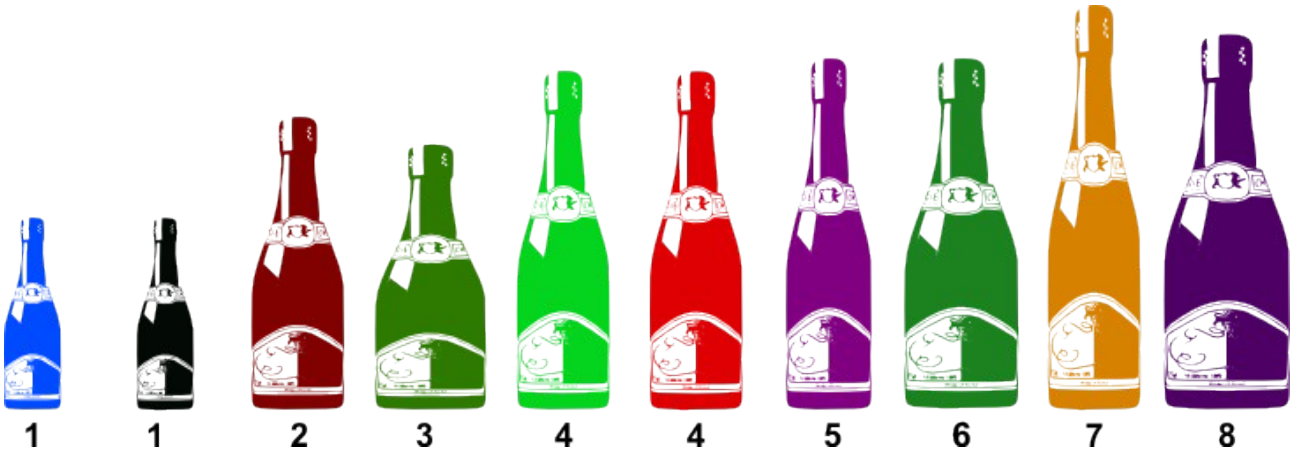
Par exemple, imaginez que vous vouliez trier la collection de bouteilles ci-dessous par ordre de volume (le volume est indiqué sous la bouteille) :

A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm

Si vous obtenez ceci, alors votre tri n'était pas stable :

En effet, la bouteille noire de volume 1 se trouve maintenant avant la bouteille bleue de même volume alors qu'elle devrait être après. Il en est de même pour les deux bouteilles de volume 4 qui sont inversées par rapport à l'ordre initial.

Avec un tri stable, on aurait obtenu :



L'intérêt d'un tri stable est qu'on peut appliquer ce tri successivement, avec des critères différents. Imaginez, par exemple, que dans le cadre d'une campagne de prévention, vous vouliez trier une liste de personnes en fonction de la catégorie d'âge. Vous obtenez une liste avec les personnes les plus âgées en premier. Si maintenant, vous voulez re-trier cette liste en fonction du poids, par exemple, vous aurez, si l'algorithme de tri est stable, les personnes les plus lourdes en premier et, pour les

personnes de même poids, celles qui sont les plus âgées en premier. Ce qu'un algorithme non-stable ne garantirait pas.

3.Tri par insertion :

a. Définition et motivations :

Dans l'algorithme, on parcourt le tableau à trier du début à la fin. Au moment où on considère le *i*-ème élément, les éléments qui le précèdent sont déjà triés. Pour faire l'analogie avec l'exemple du jeu de cartes, lorsqu'on est à la *i*-ème étape du parcours, le *i*-ème élément est la carte saisie, les éléments précédents sont la main triée et les éléments suivants correspondent aux cartes encore en désordre sur la table.

L'objectif d'une étape est d'insérer le *i*-ème élément à sa place parmi ceux qui précèdent. Il faut pour cela trouver où l'élément doit être inséré en le comparant aux autres, puis décaler les éléments afin de pouvoir effectuer l'insertion. En pratique, ces deux actions sont fréquemment effectuées en une passe, qui consiste à faire « remonter » l'élément au fur et à mesure jusqu'à rencontrer un élément plus petit.

b. Alogrithme :

Pseudo-code de l'algorithme présenté :

Les éléments du tableau *T* (de taille *n*) sont numérotés de 0 à *n*-1.

```
procédure tri_insertion(tableau T, entier n)
    pour i de 1 à n - 1

        # mémoriser T[i] dans x
        x ← T[i]

        # décaler vers la droite les éléments de T[0]..T[i-1] qui sont plus grands que x en partant de T[i-1]
        j ← i
        tant que j > 0 et T[j - 1] > x
            T[j] ← T[j - 1]
            j ← j - 1

        # placer x dans le "trou" laissé par le décalage
        T[j] ← x
```

OU

Les méthodes de tri -NSI
5

```
PROCEDURE tri_Insertion ( Tableau a[1:n])
    1.  POUR i VARIANT DE 2 A n FAIRE
    2.      INSERER a[i] à sa place dans a[1:i-1];
    3.FIN PROCEDURE;
```

Le tri par insertion est un tri stable (conservant l'ordre d'apparition des éléments égaux) et un tri en place (il n'utilise pas de tableau auxiliaire).

L'algorithme a la particularité d'être online, c'est-à-dire qu'il peut recevoir la liste à trier élément par élément sans perdre en efficacité.

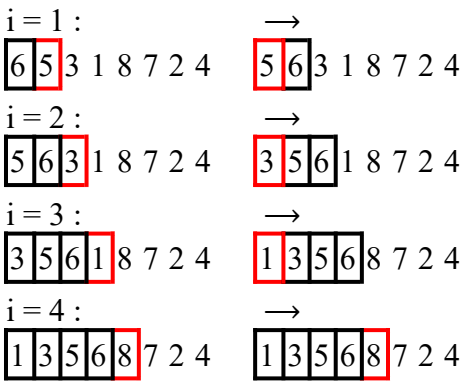
Code python pour tri avec insertion :

```
def tri_insertion(tableau):
    1.  for i in range(1,len(tableau)):
    2.      en_cours = tableau[i]
    3.      j = i
    4.      #décalage des éléments du tableau }
    5.      while j>0 and tableau[j-1]>en_cours:
    6.          tableau[j]=tableau[j-1]
    7.          j = j-1
    8.      #on insère l'élément à sa place
    9.      tableau[j]=en_cours
```

c.Exemple :

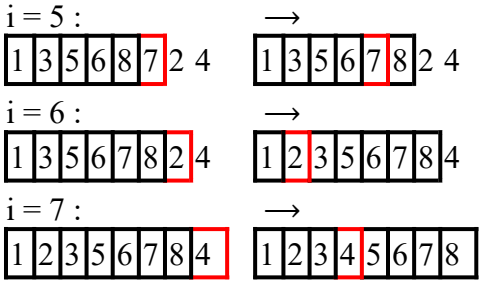
Voici les étapes de l'exécution du tri par insertion sur le tableau [6, 5, 3, 1, 8, 7, 2, 4]. Le tableau est représenté au début et à la fin de chaque itération.

Illustration graphique du tri par insertion :



A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm

Les méthodes de tri -NSI
6



4. Le tri par sélection :

a-Définition :

Le **tri par sélection** (ou **tri par extraction**) est un algorithme de tri par comparaison. Cet algorithme est simple, mais considéré comme inefficace, car il s'exécute en temps quadratique en le nombre d'éléments à trier, et non en temps pseudo linéaire.

b-Tableau :

En informatique, un **tableau** est une structure de données représentant une séquence finie d'éléments auxquels on peut accéder efficacement par leur position, ou *indice*, dans la séquence. C'est un type de conteneur que l'on retrouve dans un grand nombre de langages de programmation.

Dans les langages à typage statique (comme C, Java et OCaml), tous les éléments d'un tableau doivent être du même type. Certains langages à typage dynamique (tels APL et Python) permettent des tableaux hétérogènes.

c- Extremums :

L'expression « élément **extremum** » signifie « élément maximum » ou « élément minimum ».

Dans un ensemble ordonné E, un élément d'une partie A est le **plus grand élément** ou **maximum** de A, s'il appartient à A et est supérieur à tout autre élément de A. L'existence d'un maximum n'est en général pas assurée pour toute partie d'un ensemble ordonné. En revanche, sous condition d'existence, un tel élément est unique (ce qui justifie l'emploi de l'article défini « le » dans la définition). De manière analogue, le **plus petit élément** ou **minimum** est, s'il existe, un élément de A inférieur à tout autre élément de A.

d-Principe du tri par sélection :

Sur un tableau de n éléments (numérotés de 0 à *n-1 , attention un tableau de 5 valeurs (5 cases) sera numéroté de 0 à 4* et non de 1 à 5), le principe du tri par sélection est le suivant :

- rechercher le plus petit élément du tableau, et l'échanger avec l'élément d'indice 0 ;
- rechercher le second plus petit élément du tableau, et l'échanger avec l'élément d'indice 1 ;
- continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.

A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm

Les méthodes de tri -NSI
7

c. Description, pseudo-code et variantes

En pseudo-code, l'algorithme s'écrit ainsi :

```
procédure tri_selection(tableau t, entier n)
   pour i de 0 à n - 2
      min ← i
      pour j de i + 1 à n - 1
         si t[j] < t[min], alors min ← j
      fin pour
      si min ≠ i, alors échanger t[i] et t[min]
   fin pour
fin procédure
```

Une variante consiste à procéder de façon symétrique, en plaçant d'abord le plus grand élément à la fin, puis le second plus grand élément en avant-dernière position, etc.

Le tri par sélection peut aussi être utilisé sur des listes. Le principe est identique, mais au lieu de déplacer les éléments par échanges, on réalise des suppressions et insertions dans la liste.

OU

```
PROCEDURE tri_Selection ( Tableau a[1:n])
1.  POUR i VARIANT DE 1 A n - 1 FAIRE
2.     TROUVER [j] LE PLUS PETIT ELEMENT DE [i + 1:n];
3.     ECHANGER [j] ET [i];
4.FIN PROCEDURE;
```

d. Code Python pour tri par sélection :

```
def tri_selection(tableau):
1.  nb = len(tableau)
2.  for en_cours in range(0,nb):
3.     plus_petit = en_cours
4.     for j in range(en_cours+1,nb) :
5.        if tableau[j] < tableau[plus_petit] :
6.           plus_petit = j
7.     if min is not en_cours :
8.        temp = tableau[en_cours]
9.        tableau[en_cours] = tableau[plus_petit]
10.        tableau[plus_petit] = temp
```

5. Complexité :

A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm

Les méthodes de tri -NSI
8

a. Critères de classification :

La classification des algorithmes de tri est très importante, car elle permet de choisir l'algorithme le plus adapté au problème traité, tout en tenant compte des contraintes imposées par celui-ci. Les principales caractéristiques qui permettent de différencier les algorithmes de tri, outre leur principe de fonctionnement, sont la complexité temporelle, la complexité spatiale et le caractère stable.

b. Principe de fonctionnement :

On distingue les algorithmes procédant par comparaisons successives entre éléments, dits « tris par comparaisons », des algorithmes plus spécialisés faisant des hypothèses restrictives sur la structure des données à trier (par exemple, le tri par comptage, applicable uniquement si les données sont prises dans un ensemble borné connu à l'avance).

Les algorithmes de tri par comparaison lisent les entrées uniquement au moyen d'une fonction de comparaison binaire ou ternaire (lorsque le cas d'égalité est traité différemment). Il existe encore différents principes de fonctionnement au sein de cette classe : certains algorithmes de tri par comparaison procèdent par insertions successives, d'autres par fusions, d'autres encore par sélection.

En l'absence de précisions, on entend habituellement par « algorithme de tri » un algorithme de tri procédant par comparaisons.

c. Complexité algorithmique :

- La complexité temporelle (en moyenne ou dans le pire des cas) mesure le nombre d'opérations élémentaires effectuées pour trier une collection d'éléments. C'est un critère majeur pour comparer les algorithmes de tri, puisque c'est une estimation directe du temps d'exécution de l'algorithme. Dans le cas des algorithmes de tri par comparaison, la complexité en temps est le plus souvent assimilable au nombre de comparaisons effectuées, la comparaison et l'échange éventuel de deux valeurs s'effectuant en temps constant.
- La complexité spatiale (en moyenne ou dans le pire des cas) représente, quant à elle, la quantité de mémoire dont va avoir besoin l'algorithme pour s'exécuter. Celle-ci peut dépendre, comme le temps d'exécution, de la taille de l'entrée. Il est fréquent que les complexités spatiales en moyenne et dans le pire des cas soient identiques. C'est souvent implicitement le cas lorsqu'une complexité est donnée sans indication supplémentaire.

La complexité en temps est souvent notée T et exprimée comme une fonction du nombre n d'éléments à trier à l'aide des notations de Landau O et Θ .

Certains algorithmes de tri simples ont une complexité en temps quadratique,

$$T(n) = O(n^2) \ ,$$

tandis que d'autres, plus élaborés, ont une complexité quasi-linéaire :

$$T(n) = O(n \log(n)) .$$

A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm

Les méthodes de tri -NSI
9

La complexité temporelle en moyenne d'un algorithme basé sur une fonction de comparaison ne peut pas être meilleure que

$$O (n \log (n)) .$$

Les tris qui ne demandent que

$$O (n \log (n))$$

comparaisons en moyenne sont par conséquent dits optimaux. Ce résultat constitue une borne inférieure asymptotique, mais on montre également que le nombre exact de comparaisons nécessaires est minoré par

$$\lceil \log 2 (n !) \rceil .$$

A.LOUGHANI
Sources : Wikipedia et http://lwh.free.fr/pages/algo/tri/tri.htm