Algorithmes de tri interne [tr] (3)
Méthodes par échanges

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Version 21 mai 2018

Table des matières

Algorithmes de tri interne [tr] (3)

Méthodes par échanges

Les **méthodes par échanges** effectuent, jusqu'à ce que le tableau soit trié, des « passages » successifs dont chacun diminue le nombre d'inversions.

Méthodes par échanges

```
Action trParEchanges ( DR A : Element [ NMAX ] ; n : Entier )
Début
  |  TantQue ( il existe deux indices ix Et jx tels que inversion ( A , ix , jx ) ) Faire
  |    |  permuterTab ( A , ix , jx )
  |  FinTantQue
Fin
```

Tout le problème réside dans le choix des éléments `ix` et `jx` à permuter. Une méthode naïve conduit à l'algorithme du « tri bulles ». Une autre, beaucoup plus élaborée, donnera le « tri rapide » qui est un algorithme par segmentation appliquant le principe de l'équilibrage.

1 Tri bulles

Nom anglais : *bubble sort*

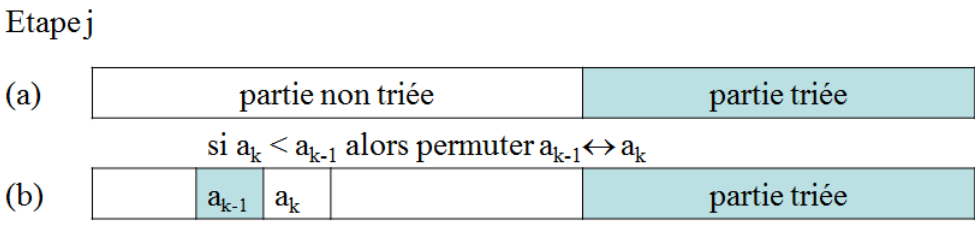
Propriétés : tri interne, sur place, non stable

Complexité : dans tous les cas en $\Theta(n^2)$

Visualisation : http://www.sorting-algorithms.com/bubble-sort

1.1 Principe du tri bulles

Le **tri bulles** intervertit toute paire d'éléments consécutifs `(A[k-1],A[k])` non ordonnés. Après le premier parcours, l'élément maximum se retrouve en `A[n]`. On recommence avec la nouvelle sous-séquence `(A[1],...,A[n-1])`, et ainsi de suite jusqu'à épuisement de toutes les sous-séquence (la dernière est un couple).



Remarque

Le sobriquet « tri bulles » vient d'une interprétation imagée selon laquelle l'algorithme fait « remonter » petit à petit les éléments « légers » vers la « surface ».

1.2 Version récursive

Procédure trBullesRec

(Tri bulles en récursif)

```
Action trBullesRec ( DR A : Element [ NMAX ] ; n : Entier )
Début
  |  Si ( n > 1 ) Alors
  |     |  echangerBulles ( A , n )
  |     |  trBullesRec ( A , n - 1 )
  |  FinSi
Fin
Action echangerBulles ( DR A : Element [ NMAX ] ; n : Entier )
Variable k : Entier
Début
  |  Pour k <- 2 à n Faire
  |     |  Si ( A [ k ] < A [ k - 1 ] ) Faire
  |     |     |  permuterTab ( A , k , k - 1 )
  |     |  FinSi
  |  FinPour
Fin
```

1.3 Complexité du tri bulles

Nombre de comparaisons

Dans la procédure récursive, trier un tableau de n éléments revient à faire remonter une bulle puis à trier récursivement un tableau de $n-1$ éléments. Si $C(n)$ désigne le nombre de comparaisons pour trier un tableau de taille n et $I(n)$ le nombre de comparaisons pour faire remonter une bulle dans un tableau de taille n, on a :

- $C(1) = 0$: lorsque le tableau n'a qu'un élément, on ne fait aucune comparaison
- pour $n > 1$: $C(n) = I(n) + C(n-1)$

Donc :

$$C(n) = I(n) + I(n-1) + ... + I(2)$$

Or $I(k) = k - 1$, d'où :

$$C(n) = \sum_{k=1}^{n-1} k = \frac{(n-1)n}{2} \in O(n^2)$$

Nombre d'échanges

Le nombre d'échanges dans le pire cas (complexité au pire = majorant du nombre d'échanges) est celui où le tableau est classé dans l'ordre inverse et donc chaque cellule doit être permutée : dans ce cas il y a donc autant d'échanges que de tests :

$$E(n) = C(n) \in O(n^2)$$

En moyenne il est également $\Theta(n^2)$ si l'élément maximal est initialement réparti aléatoirement entre les indices 1,2,..,n.

1.4 Conclusion

Différentes améliorations sont possibles (voir exercices), mais elles ne changent pas de façon significative la complexité du tri bulles qui est vicié à la base par son approche « microscopique » du tableau, chaque élément n'étant jamais comparé qu'à ses voisins immédiats. Le tri bulles est un des plus mauvais algorithmes de tri connus. Si l'on veut un algorithme de tri simple et facile à programmer, on se tournera vers le tri par insertion, toujours préférable au tri bulles.

Si l'on veut obtenir une méthode efficace procédant par échanges, il faut s'écarter de l'idée naïve du tri bulles et se tourner vers un algorithme dû à Hoare [Hoare 62], [Hoare 71] qui fournie une des meilleurs méthodes connues (sinon la meilleure). On l'appelle habituellement « tri rapide » (*quicksort*). Le nom de tri par segmentation lui convient également assez bien.