Algorithmes de tri interne (1) [tr]
Algorithmique

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Version 21 mai 2018

Table des matières

1 Concepts de base **3**
- 1.1 Qu'est-ce qu'un tri ? . . . 3
- 1.2 Représentation des données . . . 5
- 1.3 Étude de la complexité . . . 6

2 Taxonomie des tris **7**

3 Borne des tris comparatifs **9**
- 3.1 Arbre de décision . . . 9
- 3.2 Borne inférieure . . . 9
- 3.3 Conclusion : Borne inférieure . . . 10

4 Récapitulatif des algorithmes étudiés **11**

5 Conclusion **12**

6 Références générales **12**

Algorithmes de tri interne (1)

Mots-Clés Algorithmes de tris et rangs ■
Requis Axiomatique impérative, Récursivité des actions, Complexité des algorithmes ■
Difficulté • ○ ○

Introduction

Les algorithmes de tri ont une grande importance pratique : ils sont fondamentaux dans certains domaines. L'étude du tri est également intéressante en elle-même car il s'agit de l'un des domaines de l'algorithmique qui a été étudié le plus profondément, conduisant à de remarquables résultats sur la construction des algorithmes et l'étude de leur complexité.

Ce module décrit et analyse les **méthodes de tri interne qui opère par comparaison** : tri par insertion, tri de Shell, tri bulles, tri par sélection, tri par tas, tri fusion et tri rapide.

1 Concepts de base

1.1 Qu'est-ce qu'un tri ?

Terminologie
Trier consiste à ordonner les éléments d'un (multi-)ensemble en fonction de *clés* sur lesquelles est définie une relation d'ordre. Le terme *classement* serait plus exact, mais « tri » est passé dans l'usage en informatique.

Algorithme de tri
Algorithme qui prend en entrée une série de valeurs et produit en sortie une séquence de valeurs ordonnées. Plus formellement, un tri doit produire une sortie satisfaisant les deux propriétés suivantes :

- La séquence de sortie est une **permutation** de la séquence d'entrée
- Chaque élément de la séquence est plus grand que les précédents

Exemple
On cherche à trier la séquence d'entiers :

Entrées : 5, 8, -3, 6, 42, 2, 101, -8, 42, 6
↓ tri (classement par ordre croissant par exemple)
Sortie : -8, -3, 2, 5, 6, 6, 8, 42, 42, 101

Remarques

- Un tri ne supprime pas les doublons.
- Quel que soit l'algorithme de tri, une séquence vide ou réduite à un unique élément est (déjà) triée.

Tri interne v.s. tri externe
On distingue traditionnellement le **tri interne** opérant sur des données présentes en mémoire centrale, et le **tri externe** opérant sur des données appartenant à des fichiers externes. Les problèmes d'optimisation sont différents dans les deux cas :

- En *tri interne*, on cherchera surtout à réduire le nombre de comparaisons et autres opérations internes.
- En *tri externe*, on s'intéressera aussi aux comparaisons, mais la quantité d'entrée et de sorties nécessaires devient un facteur crucial de l'efficacité de l'algorithme.

Nous nous limiterons ici aux algorithmes de tri interne.

L'oeuvre maîtresse de Knuth
Elle consacre environ 400 pages au tri dont 250 pour le tri interne.

Tri comparatif
Tri qui n'utilise qu'un opérateur binaire pour placer les élément.
En général, l'opérateur « inférieur ou égal ».

Cet opérateur doit fournir un ordre total :

- Si $a \leq b$ et $b \leq a$ alors $a = b$ (antisymétrie)
- Si $a \leq b$ et $b \leq c$ alors $a \leq c$ (transitivité)
- $a \leq b$ ou $b \leq a$ pour tout a et b

1.2 Représentation des données

Les méthodes présentées supposent que le (multi-)ensemble est dans une structure tabulaire (tableau ou vecteur).

Inversion dans un tableau A
Couple d'indices i et j tels que $i < j$ et $A[i] > A[j]$.

Tableau trié
Un tableau est **trié** (en ordre croissant) s'il ne contient aucune inversion.
Les algorithmes de tri par ordre décroissant se déduisent par symétrie.

Stabilité d'un algorithme de tri
Un algorithme de tri est **stable** s'il ne modifie jamais l'ordre relatif de deux éléments de clés égales.

Remarque
Ceci peut être un critère important, en particulier si l'on trie selon une certaine clé des éléments déjà triés selon une autre clé. Exemple : un annuaire téléphonique classé par ordre alphabétique des noms, à partir duquel on veut obtenir une liste par rues et numéros : pour chaque immeuble les noms des abonnés doivent rester dans l'ordre alphabétique.

Tri en place
Un tri se fait **en place** s'il ne nécessite pas de placer plus d'un nombre constant d'éléments hors du tableau d'entrée.

Procédure de tri
La représentation des données se traduit au niveau du profil des algorithmes traitant du tri de `n` éléments d'un tableau `A` de taille maximale `NMAX` par :

Opération trierXXX

```
Constante NMAX <- ...
Action trierXXX ( DR A : Element[NMAX] ; n : Entier )
```

où `Element` est un type quelconque ayant défini les opérations de comparaison.

1.3 Étude de la complexité

Opérations élémentaires

Les opérations élémentaires qui permettent les calculs de complexité sur les tris comparatifs sont :

- La **comparaison** de deux éléments.
- La **permutation** (échange des valeurs) de deux éléments.

Les versions (récursive, itérative) étant équivalentes, on pourra raisonner indifféremment sur l'une ou l'autre pour le calcul de la complexité.

Tri indirectionnel

Quand les éléments sont volumineux et les échanges coûteux, une stratégie « payante » consiste à effectuer un **tri indirectionnel** :

1. Créer un tableau de pointeurs (adresse) sur les éléments.
2. Effectuer le tri en permutant uniquement les pointeurs et non plus les éléments.
3. Ranger les éléments dans l'ordre définitif en $O(n)$.
4. Libérer le tableau de pointeurs.

Complexité moyenne

Dans l'étude de la complexité moyenne des algorithmes de tri comparatif, seul compte l'ordre relatif des éléments : les n éléments de la séquence initiale sont supposés différents (ceci ne nuit pas au raisonnement puisque la présence d'égalité entre éléments a tendance à simplifier le tri et non à le compliquer). Ainsi :

Hypothèse 1 : Les éléments forment une permutation de l'ensemble des entiers de 1 à n (d'autres méthodes permettent d'obtenir un résultat du même ordre en donnant une probabilité non nulle aux séquences comportant des éléments égaux).

Hypothèse 2 : Les $n!$ permutations des rangs des éléments de la séquence sont supposées *équiprobables*.

2 Taxonomie des tris

Dans *The Art Of Computer Programming (Vol 3)*, Donald E. KNUTH analyse 25 algorithmes de tri différents. Mais ce n'est qu'une fraction de ce qui existe. Pourquoi tant d'algorithmes ?

- Parce que chacun a des avantages et des inconvénients.
- Ce qui marche bien pour un problème ne marche pas bien pour un autre.

Il n'y a donc pas un « meilleur » tri, il y en a plusieurs. Mais il y en a qui sont très mauvais (inefficaces).

Taxonomie des tris

On peut classer les algorithmes de tri en fonction :

- Des **méthodes** (= approches) employées :
  - *Méthodes par insertion* : chacun des éléments est considéré et placé dans la position correcte par rapport à deux déjà triés.
  - *Méthodes par échange* : si deux éléments sont trouvés dans de mauvaises positions respectives ils sont échangés (permutés).
  - *Méthodes par sélection* : le plus petit (ou grand) élément est repéré et isolé des autres, on procède de même avec le reste.
  - *Méthodes par énumération* : chaque élément est comparé avec tous les autres, sa position finale étant déterminé par le nombre d'autres éléments qu'il surpasse. Note : ces tris ne sont pas étudiés car ce ne sont pas des tris comparatifs.
  - *Méthodes spécialisées* : qui marchent parfaitement pour le problème proposé mais n'est pas efficace pour d'autres.
- Des **complexités** : en n^2 ou $n \lg n$
- De l'**équilibrage** réalisé dans la stratégie « diviser et régner » : Méthodes naïves, *Méthodes par partition.*

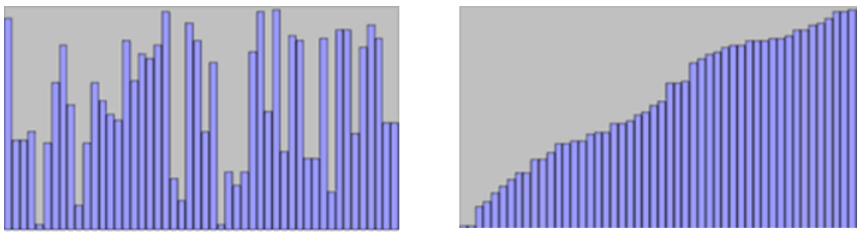
Tris internes étudiés

Les tris internes étudiés sont :

- BS : tri de la bulle (cf. méthodes par échanges)
- HS : tri par tas (cf. méthodes par sélection)
- IS : tri par insertion (cf. méthodes par insertion)
- LS : tri de Shell (cf. méthodes par insertion)
- MS : tri par fusion (cf. méthodes par partition)
- QS : tri rapide (cf. méthodes par partition)
- SS : tri par sélection (cf. méthodes par sélection)

Visualisation graphique des tris

Il est pédagogiquement intéressant de visualiser le comportement d'un tri en cours d'exécution. De nombreuses versions graphiques des tris les plus courants sont disponibles (http://www.sorting-algorithms.com/). Chaque nombre est représenté par une barre verticale (à la manière d'un histogramme) dont la hauteur est proportionnelle à la valeur. La figure (a) montre le tableau avant le tri et la figure (b) illustre le tableau après le tri.



(a) Avant le tri (b) Après le tri

3 Borne des tris comparatifs

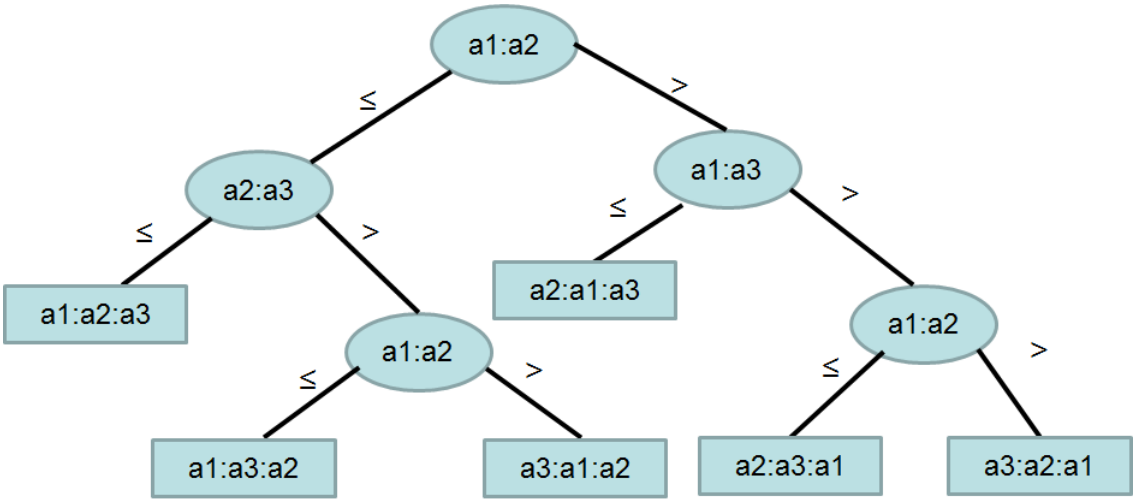
Peut-on trouver une borne sur les tris comparatifs dans le pire des cas ? Nous allons supposer que nous trions un ensemble d'éléments distincts (c.-à-d. aucune comparaison $a_i = a_j$) et que toutes les comparaisons sont de la forme $a_i \leq a_j$.

Le tri étant une permutation, nous allons étudier toutes les permutations possibles. Pour faciliter nos réflexions, nous allons utiliser un arbre de décision.

3.1 Arbre de décision

Arbre de décision
Arbre binaire plein qui permet de lister tous les choix possibles et tous les résultats possibles.

Exemple : Arbre de décision pour le tri de trois éléments
L'arbre de décision pour le tri de trois éléments $< a_1, a_2, a_3 >$ est le suivant :



- Chaque noeud interne représente la comparaison entre deux éléments.
- Chaque feuille représente les permutations devant être effectuées pour avoir l'ensemble trié.
- Dans le pire des cas, il faut donc parcourir la branche la plus longue pour atteindre la « bonne » permutation (mais cela reste le cas optimal).

(Voir [Cormen-AL1] pour plus de détails.)

3.2 Borne inférieure

Nombre d'opérations
Combien d'opérations devons nous effectuer ?

Réponse
Autant que la hauteur de l'arbre.

Théorème
Tout arbre de décision qui trie n éléments a pour hauteur $\Omega(n \lg n)$.

Preuve
Il y a $n!$ permutations possibles, donc l'arbre doit avoir au moins $n!$ feuilles. (Il peut en avoir plus si on fait plusieurs fois les mêmes permutations.) Un arbre binaire de hauteur h a au plus 2^h feuilles, donc :

$$n! \leq 2^h$$

Ceci implique que $h \geq \lg n!$. La formule de STIRLING donne $n! > \left(\frac{n}{e}\right)^n$. En l'injectant dans la relation précédente, on obtient :

$$\begin{aligned} h &> \ln \left(\frac{n}{e}\right)^n \\ &\geq n \lg n - n \lg e \\ &\geq \Omega(n \lg n) \end{aligned}$$

■

3.3 Conclusion : Borne inférieure

Récapitulons :

- Un arbre de décision pour un tri à n éléments a $n!$ feuilles.
- Sa hauteur est **minorée** par $n \lg n$.
- Or un tri comparatif est un arbre de décision.
- Donc sa hauteur est aussi minorée par $n \lg n$.

Mais que représente la hauteur pour le tri ?

- Si on a de la chance, on aura pas besoin d'aller jusqu'à la hauteur.
- C'est donc le pire cas.

Conclusion
Tout **tri comparatif** doit effectuer au moins $\Omega(n \lg n)$ comparaisons pour trier une séquence de n éléments.

Tris comparatifs uniquement
On ne parle que des tris comparatifs. Il existe d'autres tris qui ont une borne inférieure différente.

4 Récapitulatif des algorithmes étudiés

Cette section fournit un résumé de la complexité algorithmique des algorithmes de tri présentés dans le module.

Algorithmes	En Place	Stable	C_{min}	C_{moy}	C_{max}
Insertion séquentielle	oui	oui	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Insertion dichotomique	oui	oui(2)		$\Theta(n \lg n)$	$O(n \lg n)$
Insertion par incréments	oui	NON		$\Theta(n^{1.25})$	?
Sélection par scintillement	oui	oui	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Sélection ordinaire	oui	oui	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$
Sélection quadratique	oui	oui(2)			$O(n\sqrt{n})$
Sélection par priorité(1)	oui	NON			$O(n \lg n)$
Tri par fusion	oui	??	$\Omega(n \lg n)$		$O(n \lg n)$
Tri par segmentation	oui	NON	$\Omega(n \lg n)$	$\Theta(n \lg n)$	$O(n^2)$

1 Tri asymptotiquement optimal : $C_{min}(TC) \in \Omega(n \lg n)$.

2 Tri stable si le test de condition sur la valeur des éléments dans la fonction de sélection ou de recherche dichotomique assure que les insertions s'effectuent après les éléments égaux (s'il y en a).

5 Conclusion

Il existe beaucoup d'autres et de nombreuses variantes de tri comparatif : Tri par sélection des extrémums, Tri tournoi, Tri de Dijkstra...

Attention
Les tris étudiés sont des tris basés sur la comparaison (et l'échange). Ils existent des tris (en $O(n)$) qui utilisent la propriété des valeurs de clés : tri par seaux et tri radix (par exemple).

6 Références générales

Cormen-AL1, chapitres 1 à 8 @BookCormen-AL1, author = Cormen, T.H. AND Leiserson, C.E. AND Rivest, R.L. AND Stein, C., title = Algorithmique, publisher = Dunod Informatique, ISBN = 2-10-003922-9, year = 2010 (3e édition), type = Algorithmique fondamentale, note = Cours avec 957 exercices et 158 problèmes – Compléments en ligne www.dunod.com

Hervé-R1 @InProceedingsHerve-R1, author = Hervé N., title = Les algorithmes de tri, Mémoire Informatique, booktitle = Conservatoire national des arts et Métiers (2004),

Sedgewick-CPP1, chapitres 6, 9, 12 et 13 @BookSedgewick-CC1, author = Sedgewick, R., title = Algorithmes en langage C, publisher = Dunod, year = 2001, isbn = , type = , edition = , school = , note = Version Inter-editions en 1991

Sedgewick-R1 @TechReportSedgewick-R1, author = Sedgewick R. and Bentley J., title = Quicksort is optimal, institution = Stanford University, year = 2002, note = http://www.cs.princeton.edu/~rs/talks/QuicksortIsOptimal.pdf,

Seiden-R1 @TechReportSeiden-R1, author = Seiden S., title = Theoretical Computer Science Cheat Sheet, year = 1994, note = http://www.tug.org/texshowcase/cheat.pdf,