Algorithmes de Tri. Exemples, Compléxités et Applications

Motivations Les tris offrent une bonne introduction à l'algorithme et l'analyse de complexité. Certains algorithmes exploitent plus efficacement des données triés.

Point Historique [CERV]

I. Introduction au tri [COR2]

A. Définitions

Def 1 **Problème de Tri** Entrée : suite de nombres $< a_1, ..., a_n >$. Sortie : Permutation (réorganisation) $< a_{\sigma(1)}, ..., a_{\sigma(n)} >$ de la suite donnée en entrée telle que $a_{\sigma(1)} \leq ... \leq a_{\sigma(n)}$.

Exemple 2 $< 42_1, 41_2, 1_3, 2_4, 1_5 > \to < 1_5, 1_3, 2_4, 41_2, 42_1 >$

Définition 3 **Un tri est en place** s'il n'utilise qu'une quantité constante de mémoire (mis à part pour l'entrée).

Définition 4 **Un tri est stable** si $\forall i < j, a_i \leq a_j \Rightarrow \sigma(i) \leq \sigma(j)$

Exemple 5 L'algorithme utilisé pour l'exemple 2 n'est *pas* stable.

Définition 6 Parfois, la taille de l'entrée à trier peut excéder la taille de la mémoire directement accessible. On parle alors de tri **externe**. Sinon on parle de tri **interne**.

Définition 7 **Un tri est dit *par comparaisons*** s'il détermine la sortie en comparant les éléments en entrée.

B. Outils pour l'analyse de complexité.

Idée 8 Nous allons utiliser la complexité asymptotique dans le pire cas afin de comparer les algorithmes de tri entre eux.

Notation 9 **Notations de Landau** Soient f, g deux fonctions. On note:

- $f = O(g)$ si $\exists N, c \in \mathbb{R}_+, \forall x > N, |f(x)| \leq c\,|g(x)|$
- $f = \Omega(g)$ si $\exists N, c \in \mathbb{R}_+^*, \forall x > N, |f(x)| \geq c\,|g(x)|$
- $f = \Theta(g)$ si $f = O(g)$ et $g = O(f)$

Théorème 10 **Theorème Maitre.** Pour la relation, $T(1) = O(1)$ et $\forall n \in \mathbb{N}, T(n) = aT\left(\frac{n}{b}\right) + O(n^c)$ avec $c = \log_b(a)$, on a $T(n) = O(n^c \log(n))$.

II. Algorithmes de tris [COR2]

A. Tris par comparaisons

Idée 11 S'inspirer d'un tri d'un main de cartes.

Algo 13 **Tri par Insertion**

```
TRI_INSERTION(T):
(*)POUR j = 1 à |T| - 1:
     clé = T[j]
     i = j - 1
     TANT QUE i >= 0 et T[i] > clé:
         T[i + 1] = T[i]
         i = i - 1
     T[i + 1] = clé
```

Complexité 12 $O(n^2)$ dans le pire cas ($n = |T|$ la taille du tableau à trier).

Propriété 14 Ce tri est stable et en place.

Invariant 15 Après le tour de boucle `(*)POUR`, `T[0 ... k]` est trié.

Remarque 16 Il est également possible dde définir un algorithme qui choisit itérativement le plus petit des élèments non selectionnés et construit ainsi un tableau de sortie. c'est le **tri par selection**.

Complexité 17 La complexité du tri par selection est de $O(n^2)$

Idée 18 Utiliser l'approche « diviser pour régner ».

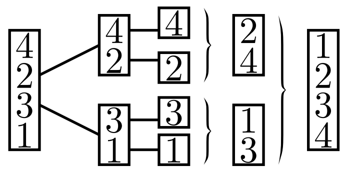
Algorithme 19 **Tri Fusion**

```
TRI_FUSION(T, p, r):
  SI p < r:
    q = floor(p + r / 2)
    TRI_FUSION(T, p, q)
    TRI_FUSION(T, q + 1, r)
    FUSION(T, p, q, r)
```

```
FUSION(T, p, q, r):
  L, R = copie(T[p:q], T[q+1:r])
  Ajouter +inf à L et R
  i = j = 1
  POUR k = p à r:
    SI L[i] <= R[j]:
      T[k] = L[i]; i += 1
    SINON: T[k] = R[j]; j += 1
```

Complexité 20 $O(n \log(n))$

Exemple 21



Propriété 22 Le tri fusion est stable mais *pas* en place !

DEV

Algorithme 23 **Le Tri Rapide** choisit un pivot, qu'il place entre les élèment plus petits que le pivot triés récursivement et les élèments plus grands que le pivot triés récursivement.

Complexité 24 $O(n^2)$ au pire cas mais $O(n\log(n))$ en moyenne.

Propriété 25 Ce tri est en place, mais non stable.

B. Tris utilisant une structure de données

Définition 26 **Un tas max** est un arbre binaire quasi-complet tel que la valeur de tout noeud est plus petite que celle de sont père.

Algorithme 27 **Le tri par Tas** construit un tas max, puis extrait itérativement l'élèment le plus grand du tas.

Complexité 28 $O(n\log(n))$

Propriété 29 Ce tri est en place mais non stable.

Remarque 30 Il est également possible de faire un **tri par ABR** dans lequel on insère un à un les éléments dans l'ABR, on fait ensuite un parcours infixe en profondeur.

Remarque 31 Tout tri par comparaisons exige $\Omega(n\log(n))$ comparaisons dans le pire cas.

Remarque 32 Même si contrairement au tri fusion et au tri par tas, le tri rapide n'est pas **optimal**, il reste très efficace en pratique d'où son nom.

Remarque 33 Il est possible de rendre un tri stable en indexant chaque élèment par son indice dans le tableau $a_i \to (a_i, i)$ et en utilisant un ordre lexicographique.

Remarque 34 Le tri par insertion est en générale très efficace sur de petites entrées, ou quand l'entrée est déjà triée.

C. Tri Linéaires

Idée 35 Si on sait que tous les éléments du tableau sont des entiers plus petits que $k \in \mathbb{N}$, alors on peut utiliser cette hypothèse pour faire un tri qui n'est pas par comparaisons.

Algo 37 **Tri Comptage**

```
TRI_COMPTAGE(A, B, k):
  C[0 ... k] = [0 ... 0]
  POUR j = 0 à |A| - 1:
    C[A[j]] += 1
  POUR i = 1 à k:
    C[i] += C[i - 1]
  POUR j = |A| - 1 à 0:
    B[C[A[j]]] = A[j]
    C[A[j]] -= 1
```

Complexité 36 $O(n+k)$

Propriété 38 Le tri n'est pas en place mais est stable.

Application 39 On peut utiliser ce tri pour définir le **tri par base** :

```
TRI_BASE(A, d):
  POUR i = 1 à d:
    TRI_COMPTAGE(A)
    selon le i° chiffre
```

Exemple 40

$$\begin{array}{ccc} 3 & 5 & 6 \\ 1 & 2 & 7 \\ 2 & 5 & 6 \end{array} \to \begin{array}{ccc} 3 & 5 & 6 \\ 2 & 5 & 6 \\ 1 & 2 & 7 \end{array} \to \begin{array}{ccc} 1 & 2 & 7 \\ 3 & 5 & 6 \\ 2 & 5 & 6 \end{array} \to \begin{array}{ccc} 1 & 2 & 7 \\ 2 & 5 & 6 \\ 3 & 5 & 6 \end{array}$$

Complexité 41 $O(d*(n+k))$

III. Applications des tris [COR2]

Application 42 Un tableau trié permet la **recherche Dichotomique** d'un élément en $O(\log(n))$.

Exemple 43 42 ? $\boxed{1|2|3|4|42|100} \to \boxed{4|42|100} \to \boxed{42}$ ✓

Application 44 Sur le même principe, une **insertion dichotomique** dans une structure adaptée permet de définir un **tri par insertion dichotomique** en $O(n\log(n))$.

Application 45 La recherche ou élimination de doublons On trie d'abord puis on parcourt en comparant les éléments adjacents. Ceci se fait en $O(n\log(n))$ contre $O(n^2)$ pour l'approche naïve.

A. Algorithme Gloutons

Application 46 L'Algorithme de Kruskal trouve un arbre couvrant de poids minimum en parcourant les arêtes triées par poids croissants.

Application 47 D'autres algos gloutons requiert de trier les entrées selon une métrique : Problème du sac à dos, Problème d'emploi du temps.

Exemple 48 Une exécution de l'algorithme de Kruskal

B. Parcours Intelligents

Application 49 Parcours de Graham [COR3 p.1031] En triant d'abord les points par rapport à leur angle avec le point le plus bas $O(n\log(n))$, le parcours de Graham peut ensuite calculer l'enveloppe convexe de l'ensemble de ces points avec une pile en $O(n\log(n))$.

Exemple 50 Une étape de l'algorithme de Graham

•

Application 51 Tri topologique Pour faire le tri topologique d'un graphe G (c'est à dire ordonner les sommets selon un ordre total compatible avec l'ordre partiel induit par les arcs de G) on peut parcourir G en profondeur et ordonner les sommets par dates décroissantes de fermeture.

Exemple 52 Un tri topologique par parcours en profondeur

IV. Réseaux de tris [COR2 p.681]

Idée 53 Paralléliser des comparaisons !

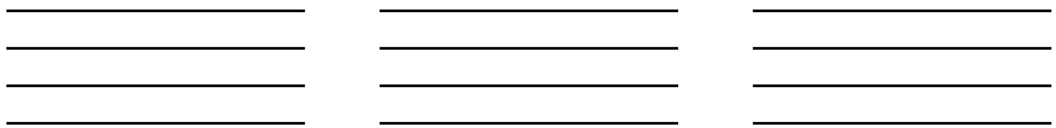
Définition 54 Un comparateur est une brique de base d'un réseau de tri, comparant et ordonnant deux valeurs (fils).

Exemple 55 Deux comparateurs

Def 56 Un réseau de comparaison est un ensemble de n fils sur lesquels on a placé des comparateurs.

Définition 57 On dit que c'est un **réseau de tri** lorsque peu importe les entrées, les valeurs sur les fils de sortie sont triées.

Exemple 58 Même réseau de tri pour un tri par selection ou insertion



Tri par insertion — Même réseau de tri — Tri par selection

Théorème 59 Le principe de zéro-un stipule que si un réseau de tri fonctionne pour toure entrée à valeurs dans $\{0,1\}$, alors il fonctionne pour tour entrée.

Corollaire 60 Pour savoir si on a un réseau de tri, on a donc seulement $2^n \ll O(n!)$ entrées à tester !

Application 61 (Admis) Grâce à une trieuse bitonique et un réseau de fusion, on peut trier n nombres avec un réseau de tri de profondeur $O(\log^2(n))$

Application 62 « **La machine humaine à trier** » est une activité d'**informatique débranché** basée sur les réseaux de tri.

Algorithmes de Tri. Exemples, Compléxités et Applications

I. Introduction au tri [COR2]

A. Définitions

1 Def Problème de Tri
2 Ex Un tri d'une liste
3 Def Un tri est en place
4 Def Un tri est stable
5 Ex
6 Def Tri externe
7 Def Un tri est dit *par comparaisons*

B. Outils pour l'analyse de complexité.

8 Idée
9 Not Notations de Landau
10 Thm Theorème Maitre.

II. Algorithmes de tris [COR2]

A. Tris par comparaisons

11 Idée
12 Complex
13 Algo Tri par Insertion
14 Prop
15 Invariant
16 Rem
17 Complex
18 Idée
19 Algo Tri Fusion
20 Complex
21 Ex
22 Prop
23 Algo Le Tri Rapide
24 Complex
25 Prop

B. Tris utilisant une structure de données

26 Def Un tas max
27 Algo Le tri par Tas
28 Complex
29 Prop
30 Rem
31 Rem
32 Rem
33 Rem
34 Rem

C. Tri Linéaires

35 Idée
36 Complex
37 Algo Tri Comptage
38 Prop
39 App Tri par base
40 Ex
41 Complex

III. Applications des tris [COR2]

42 App Recherche Dichotomique
43 Ex
44 App Tri par insertion dichotomique
45 App La recherche ou élimination de doublons

A. Algorithme Gloutons

46 App L'Algorithme de Kruskal
47 App Sac à dos, emploi du temps
48 Ex Une exécution de l'algorithme de Kruskal

B. Parcours Intelligents

49 App Parcours de Graham [COR3 p.1031]
50 Ex Une étape de l'algorithme de Graham
51 App Tri topologique
52 Ex Un tri topologique par parcours en profondeur

IV. Réseaux de tris [COR2 p.681]

53 Idée Paralléliser des comparaisons !
54 Def Un comparateur
55 Ex Deux comparateurs
56 Def Un réseau de comparaison
57 Def Un réseau de tri
58 Ex Même réseau de tri pour un tri par selection ou insertion
59 Thm Le principe de zéro-un
60 Corollaire
61 App
62 App « La machine humaine à trier »

Remarque

- toutes présentations d'un algorithme de tris est un développements

Bibliographie

[CERV] B. Christian & T. Griffiths, *Penser en Algorithmes.*

[COR2] T. H. Cormen, *Introduction à l'algorithmique (2nd édition).*

[COR3] T. H. Cormen, *Introduction à l'algorithmique (3rd édition).*

Pablo Espana Gutierrez