

Tri lexicographique

UMLV ©

Optimalité des tris par comparaisons : $O(n \log n)$

Classements linéaires

Tris lexicographiques

434

Tri de 4 éléments

UMLV ©

Suite (a, b, c, d) à classer

Méthode optimale :	comparaisons
1. classer a et b	1
2. classer c et d	1
3. comparer les 2 plus petits	1
4. insérer le dernier au sein des trois éléments classés	1 ou 2

Exemple : classer $(9, 1, 5, 17)$

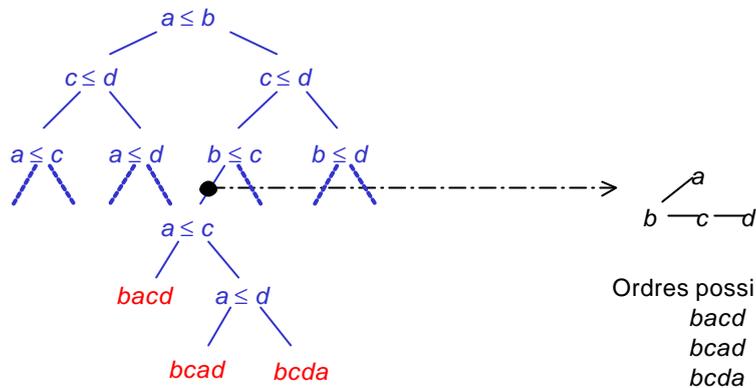
- $b=1 < a=9$
- $c=5 < d=17$
- $b=1 < c=5$ on a $b=1 < c=5 < d=17$
- Insertion de $a=9$ après comparaisons avec c puis d

$\begin{array}{c} / a \\ b \text{ --- } c \text{ --- } d \end{array}$

435

Arbre de décision

UMLV ©



Ordres possibles :
 $bacd$
 $bcad$
 $bcda$

plus longue branche (hauteur) : 5
 plus courte branche : 4
 $4! = 24$ feuilles

436

Tris par comparaisons

UMLV ©

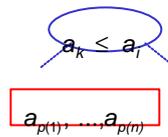
algorithme de tri par comparaisons
 sur a_1, \dots, a_n

arbre de décision
 feuilles : permutations de a_1, \dots, a_n

début

---- $a_k \leq a_i$
si ($a_k \leq a_i$) **alors**

fin



Théorème : \mathcal{A} tri par comparaisons

Alors $T_{\text{MAX}}(\mathcal{A}, n) = \Omega(n \log n)$

Preuve :

$T_{\text{MAX}}(\mathcal{A}, n) =$ hauteur de l'arbre de décision à $n!$ feuilles
 hauteur $\geq \log_2 n! = \Omega(n \log n)$ [car $n! \geq (n/2)^{n/2}$]

437

Analyse en moyenne

UMLV ©

Théorème a tri par comparaisons
permutations des entrées équiprobables

Alors $T_{\text{MOY}}(a, n) = \Omega(n \log n)$

Preuve

pour un arbre binaire à k feuilles (nœuds externes)
soit $l(k)$ la longueur moyenne des branches

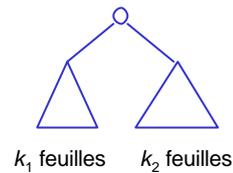
$$l(k) \geq \log_2 k$$

$$k = k_1 + k_2 \quad k_1 < k, \quad k_2 < k$$

$$l(k) = 1 + \frac{k_1}{k} l(k_1) + \frac{k_2}{k} l(k_2)$$

$$\geq 1 + \frac{k_1}{k} \log k_1 + \frac{k_2}{k} \log k_2 \quad \text{minimum pour } k_1 = k_2$$

$$\geq 1 + \frac{1}{2} \log \frac{k}{2} + \frac{1}{2} \log \frac{k}{2} \geq \log_2 k$$



438

Tris par comparaisons optimaux

UMLV ©

Optimaux au pire

Tri par le tas
Tri par fusion
Tri par arbre AVL } $O(n \log n)$

Optimaux en moyenne

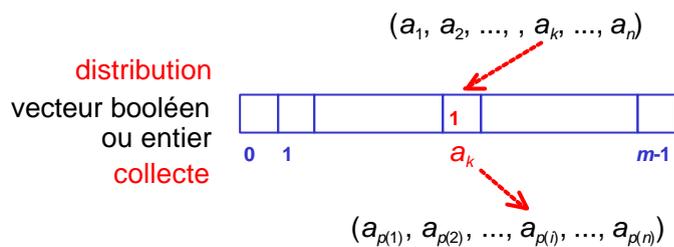
précédents, plus
Tri rapide
Tri par arbre binaire
de recherche } $O(n \log n)$

439

Tri par distribution

UMLV ©

Classer $L = (a_1, a_2, \dots, a_n)$ constituée de n entiers $\in (0, 1, \dots, m-1)$



Temps : $O(n+m)$ (pas de comparaisons)

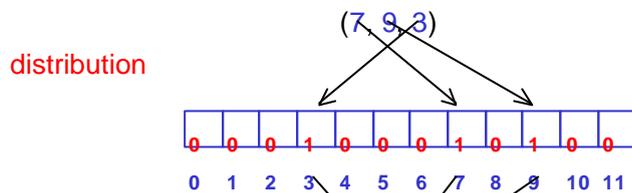
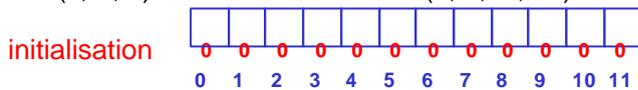
Espace supplémentaire : $O(m)$

440

Exemple

UMLV ©

$L = (7, 9, 3)$ constituée d'entiers $\in (0, 1, \dots, 11)$



441

Tri « sur place »

UMLV ©

Suite (a_1, \dots, a_n) dans une table ; clés = entiers $1, 2, \dots, n$

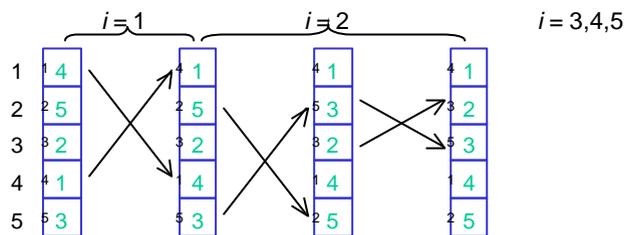
Classement (permutation) :

```

pour  $i \leftarrow 1$  à  $n$  faire
    tant que  $\text{clé}(a_i) \neq i$  faire
        échanger  $(a_i, a_{\text{clé}(a_i)})$ 
    
```

Temps : $O(n)$

Espace supplémentaire constant.



442

Tri par bacs

(sélection de place)

UMLV ©

Classer $L = (a_1, a_2, \dots, a_n)$ clés $\in (0, 1, \dots, m-1)$

Idée : utiliser m bacs et ranger a_i dans $\text{bac}[\text{clé}(a_i)]$

fonction tri-par-bacs (L liste) : liste ;

début

```

1 pour  $i \leftarrow 0$  à  $m-1$  faire vider  $\text{bac}[i]$  ;
2 pour  $a \leftarrow$  premier au dernier élément de  $L$  faire
    ajouter  $a$  à la fin de  $\text{bac}[\text{clé}(a)]$  ;
    $L' \leftarrow$  liste-vide ;
3 pour  $i \leftarrow 0$  à  $m-1$  faire
    $L' \leftarrow L'.\text{bac}[i]$  ;

```

retour (L') ;

fin

Temps : $O(n+m)$ si bonne gestion des bacs [$O(n)$ pour 2, $O(m)$ pour 1 et 3]

Espace supplémentaire : $O(m)$

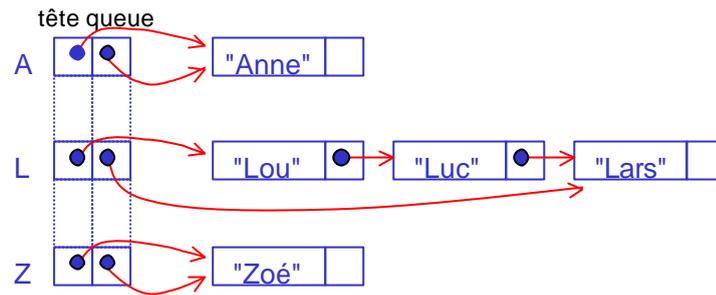
443

Distribution

UMLV ©

Classer ("Zoé", "Lou", "Anne", "Luc", "Lars")
clé = première lettre

Bacs (listes chaînées) : **temps constant par élément**

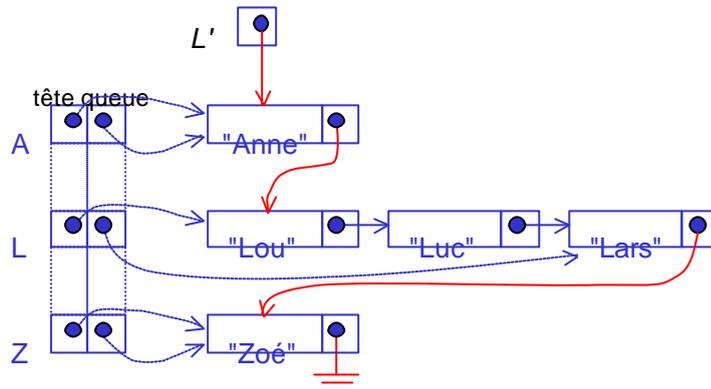


444

Collecte

UMLV ©

Concaténation des bacs : **temps constant par bac**
(tri stable)



445

Exemple

UMLV ©

Classer $L = (a_1, \dots, a_n)$ clés $\in (0, 1, 2, \dots, n^2 - 1)$

Tri en $O(n)$:

1. tri par bacs avec clé : $\text{clé}(a_i) \bmod n$
2. tri par bacs avec clé : $\lfloor \text{clé}(a_i) / n \rfloor$

Équivalent à un tri lexicographique
avec la clé (q, r) telle que $\text{clé}(a) = q.n + r$

$n=10$ $L = (26, 9, 0, 25, 1, 29, 64, 16, 81, 4)$
après 1
(0, 1, 81, 64, 4, 25, 26, 16, 9, 29)
après 2
(0, 1, 4, 9, 16, 25, 26, 29, 64, 81)

446

Ordre lexicographique

UMLV ©

Suites = mots de A^*

Fortement inférieur, \ll

$(a_1, \dots, a_k) \ll (b_1, \dots, b_l)$ ssi

il existe $i, 1 \leq i \leq \min\{k, l\}, a_1 = b_1, \dots, a_{i-1} = b_{i-1}, a_i < b_i$

information

informatique

Ordre \leq

$x, y \in A^*$ $x \leq y$ ssi x préfixe de y ou $x \ll y$

Classement

oubli	bague
orgue	baguette
baguette	orgue
oublier	oubli
bague	oublier

447

Tri lexicographique

UMLV ©

L liste de mots de même longueur k

fonction tri-lexico (L liste) : liste ;

début

pour $i \leftarrow k$ à 1 **pas** -1 **faire**

$L \leftarrow$ tri-par-bacs(L) avec clé = i -ème composante ;

retour (L) ;

fin

Preuve : car "tri-par-bacs" stable

Temps : $O(k(n+m))$ si composantes $\hat{I}(0, 1, \dots, m-1)$

	-	-	-	-	-
oubli	orgue	balai	ouate	bague	bague
orgue	bague	oubli	oubli	balai	balai
bague	ouate	ouate	orgue	orgue	orgue
balai	oubli	orgue	bague	ouate	ouate
ouate	balai	bague	balai	oubli	oubli

448

Tri lexicographique (suite)

UMLV ©

Méthode ordinaire

oubli	bague	bague	bague
orgue	balai	balai	balai
bague	oubli	orgue	orgue
balai	orgue	oubli	ouate
ouate	ouate	ouate	oubli

Temps maximal : $O(k(n+m))$ si composantes $\hat{I}(0, 1, \dots, m-1)$

Temps moyen : $O((n+m) \log n)$

si composantes équiprobables et indépendantes

Alphabet fixe : « m disparaît »

Théorème

On peut classer une suite de chaînes de caractères en temps $O(l)$ où l est la somme de leurs longueurs.

449

Tri lexico par échanges

UMLV ©

Alphabet binaire $A = \{a, b\}$ (ou $\{0,1\}$)

aaab	aaaa	aaab	aaab
baab	abba	aaba	aaba
bbaa	aaba	abba	abba
aaba	bbaa	baab	baab
abba	baab	bbaa	bbaa

Tri analogue au tri rapide (i indice de composante)

fonction tri-lexico-par-échanges (L liste) : liste ;

début

$g \leftarrow 1$; $d \leftarrow n$; $i \leftarrow 1$;

TLE (L, g, d, i);

retour (L);

fin

450

TLE

UMLV ©

$L[p, i] = i$ -ème lettre du mot d'indice p de L

procédure TLE (L, g, d, i);

début

si $g < d$ **et** $i \leq k$ **alors** {

$p \leftarrow g$; $q \leftarrow d$;

répéter

tant que ($L[p, i] = a$ **et** $p < q$ **faire** $p \leftarrow p+1$;

tant que ($L[q, i] = b$ **et** $p < q$ **faire** $q \leftarrow q-1$;

échanger ($L[p, i], L[q, i]$);

jusqu'à ce que $p = q$;

si ($L[q, i] = a$ **alors** $q \leftarrow q+1$;

TLE ($L, g, q-1, i+1$);

TLE ($L, q, d, i+1$);

}

fin

451