

## Les algorithmes de tri

Nicolas Delestre et Michel Mainguenaud

{Nicolas.Delestre,Michel.Mainguenaud}@insa-rouen.fr

Adapté pour l'ENSICAEN par

Luc Brun

{luc.brun}@insa-rouen.fr



# Plan...

---

- Les algorithmes de tri
  - Définition d'un algorithme de tri,
  - Le tri par minimum successifs,
  - Le tri à bulles,
  - Le tri rapide.
- Les algorithmes de recherche.
  - Recherche séquentielle non triée
  - Recherche séquentielle triée,
  - Recherche dichotomique.

# Définition d'un algorithme de Tri

---

- Les tableaux permettent de stocker plusieurs éléments de même type au sein d'une seule entité,
- Lorsque le type de ces éléments possède un ordre total, on peut donc les ranger en ordre croissant ou décroissant,
- Trier un tableau c'est donc ranger les éléments d'un tableau en ordre croissant ou décroissant
  - Dans ce cours on ne fera que des tris en ordre croissant
- Il existe plusieurs méthodes de tri qui se différencient par leur complexité d'exécution et leur complexité de compréhension pour le programmeur.
  - Examinons tout d'abord : *le tri par minimum successif*

# La procédure échanger...

---

Tous les algorithmes de tri utilisent une procédure qui permet d'échanger (de permuter) la valeur de deux variables. Dans le cas où les variables sont entières, la procédure échanger est la suivante :

**procédure** échanger (E/S a,b : Entier)

**Déclaration** temp : Entier

**début**

    temp  $\leftarrow$  a

    a  $\leftarrow$  b

    b  $\leftarrow$  temp

**fin**

# Tri par minimum successif...

---

## ■ Principe

- Le tri par minimum successif est ce que l'on appelle un tri par sélection :
  - Pour une place donnée, on sélectionne l'élément qui doit y être positionné
- De ce fait, si on parcourt la tableau de gauche à droite, on positionne à chaque fois le plus petit élément qui se trouve dans le sous tableau droit
  - Ou plus généralement : Pour trier le sous-tableau  $t[i..nbElements]$  il suffit de positionner au rang  $i$  le plus petit élément de ce sous-tableau et de trier le sous-tableau  $t[i+1..nbElements]$

# Tri par minimum successif...

---

Par exemple, pour trier  $\langle 101, 115, 30, 63, 47, 20 \rangle$ , on va avoir les boucles suivantes :

- $i=1 \langle 101, 115, 30, 63, 47, 20 \rangle$
- $i=2 \langle 20, 115, 30, 63, 47, 101 \rangle$
- $i=3 \langle 20, 30, 115, 63, 47, 101 \rangle$
- $i=4 \langle 20, 30, 47, 63, 115, 101 \rangle$
- $i=5 \langle 20, 30, 47, 63, 115, 101 \rangle$
- Donc en sortie :  $\langle 20, 30, 47, 63, 101, 155 \rangle$

Il nous faut donc une fonction qui pour soit capable de déterminer le plus petit élément (en fait l'indice du plus petit élément) d'un tableau à partir d'un certain rang

# Fonction indiceDuMinimum...

---

**fonction** indiceDuMinimum (t : Tableau[1..MAX] d'Entier ; rang, nbElements : Naturel) : Naturel

**Déclaration** i, indiceCherche : Naturel

**début**

    indiceCherche  $\leftarrow$  rang

**pour** i  $\leftarrow$  rang+1 à nbElements **faire**

**si** t[i]<t[indiceCherche] **alors**

            indiceCherche  $\leftarrow$  i

**finsi**

**retourner** indiceCherche

**finpour**

**fin**

# Tri par minimum successif...

---

- L'algorithme de tri est donc :

**procédure** effectuerTriParMinimumSuccessif (E/S t : Tableau[1..MAX]  
d'Entier; E nbElements : Naturel)

**Déclaration** i, indice : Naturel

**début**

**pour** i  $\leftarrow$  1 à nbElements-1 **faire**

    indice  $\leftarrow$  indiceDuMinimum(t,i,nbElements)

**si** i  $\neq$  indice **alors**

        echanger(t[i],t[indice])

**finsi**

**finpour**

**fin**



# Complexité

---

- Recherche du minimum sur un tableau de taille  $n$   
→ Parcours du tableau.

$$T(n) = n + T(n - 1) \Rightarrow T(n) = \frac{n(n + 1)}{2}$$

Complexité en  $\mathcal{O}(n^2)$ .

# Le tri à bulles

---

- Principe de la méthode : Sélectionner le minimum du tableau en parcourant le tableau de la fin au début et en échangeant tout couple d'éléments consécutifs non ordonnés.

# Tri à bulles : Exemple

---

Par exemple, pour trier  $\langle 101, 115, 30, 63, 47, 20 \rangle$ , on va avoir les boucles suivantes :

- $i=1 \langle 101, 115, 30, 63, 47, 20 \rangle$ 
  - $\langle 101, 115, 30, 63, 20, 47 \rangle$
  - $\langle 101, 115, 30, 20, 63, 47 \rangle$
  - $\langle 101, 115, 20, 30, 63, 47 \rangle$
  - $\langle 101, 20, 115, 30, 63, 47 \rangle$
- $i=2 \langle 20, 101, 115, 30, 63, 47 \rangle$
- $i=3 \langle 20, 30, 101, 115, 47, 63 \rangle$
- $i=4 \langle 20, 30, 47, 101, 115, 63 \rangle$
- $i=4 \langle 20, 30, 47, 63, 101, 115 \rangle$
- Donc en sortie :  $\langle 20, 30, 47, 63, 101, 155 \rangle$

# Tri à bulles : l'algorithme

---

**procédure** TriBulles (E/S t : Tableau[1..MAX] d'Entiers, nbElements : Naturel)

**Déclaration** i, k : Naturel

**début**

**pour** i ← 0 à nbElements-1 **faire**

**pour** k ← nbElements-1 à i+1 **faire**

**si** t[k] < t[k-1] **alors**

    echanger(t[k], t[k-1])

**finsi**

**finpour**

**finpour**

**fin**

# Tri à bulles : Complexités

---

- Nombre de tests (moyenne et pire des cas) :

$$T(n) = n + T(n - 1) \Rightarrow T(n) = \frac{n(n + 1)}{2}$$

Complexité en  $\mathcal{O}(n^2)$ .

- Nombre d'échanges (pire des cas):

$$E(n) = n - 1 + n - 2 + \dots + 1 \rightarrow \mathcal{O}(n^2)$$

- Nombre d'échange (en moyenne)  $\mathcal{O}(n^2)$  (calcul plus compliqué)

En résumé : complexité en  $\mathcal{O}(n^2)$ .

# Le tri rapide

---

- Principe de la méthode
  - Choisir un élément du tableau appelé *pivot*,
  - Ordonner les éléments du tableau par rapport au pivot
  - Appeler récursivement le tri sur les parties du tableau à gauche et à droite du pivot.

# Le tri rapide

---

## ■ Algorithme :

**procédure** TriRapide (E/S t : Tableau[1..MAX] d'Entier; gauche,droit : Naturel)

**Déclaration** i,j : Naturel; pivot,x : Entier

**début**

i ← gauche; j ← droit; pivot ← t[(i+j)/2]

**répéter**

**tant que** t[i] < pivot **faire** i ← i+1 **fin tant que**

**tant que** t[j] > pivot **faire** j ← j-1 **fin tant que**

**si** i ≤ j **alors**

    echanger(tab[i],tab[j])

    i ← i+1

    j ← j-1

**finsi**

**jusqu'à ce que** i > j

**si** gauche < j **alors**

    TriRapide(t,gauche,j)

**finsi**

**si** i < droit **alors**

    TriRapide(t,i,droit)

**finsi**

**fin**

# Exemple...

---

Par exemple, pour trier  $\langle 101, 115, 30, 63, 47, 20 \rangle$ , on va avoir les itérations suivantes :

- $\langle 101(i), 115, 30(p), 63, 47, 20(j) \rangle$
- $\langle 20, 115(i), 30, 63, 47(j), 101 \rangle$
- $\langle 20, 47, 30(i), 63(j), 115, 101 \rangle$
- $\langle 20, 47(j), 30, 63(i), 115, 101 \rangle$

Et on relance le processus sur les deux sous tableaux  $\langle 20, 47 \rangle$  et  $\langle 63, 115, 101 \rangle$ .

- $\langle 63(g), 115(p), 101(d) \rangle$
- $\langle 63, 115(i), 101(j) \rangle$
- $\langle 63, 101, 115 \rangle$



# Complexité

---

- Le tri par rapport au pivot nécessite de parcourir le tableau. On relance ensuite le processus sur les deux sous tableaux à gauche et à droite du pivot.

$$T(n) = n + 2T\left(\frac{n}{2}\right)$$

Posons  $n = 2^p$ . On obtient :

$$\begin{aligned} T(p) &= 2^p + 2T(p-1) \\ &= p2^p + 2^p \end{aligned}$$

En repassant en  $n$  :  $T(n) = \log_2(n).n + n$ . La complexité est donc en  $\mathcal{O}(n \log_2(n))$  (dans le meilleur des cas).

# Algorithmes de recherche

---

- Recherche dans un tableau non trié.

**fonction** rechercheNonTrie (tab : Tableau[0..MAX] d'Éléments, x :  
Élément) : Naturel

**Déclaration** i : Naturel

**début**

    i ← 0

**tant que** (i ≤ MAX) et (tab[i] ≠ x) **faire**

        i ← i+1

**fintantque**

**si** i = MAX+1 **alors**

**retourner** MAX+1

**finsi**

**retourner** i

**fin**

# Algorithmes de recherche

---

- Recherche séquentielle dans un tableau trié.

**fonction** rechercheSeqTrie (tab : Tableau[0..MAX] d'Éléments, x : Élément) :  
Naturel

**Déclaration** i : Naturel

**début**

i ← 0

**tant que** x > tab[i] **faire**

i ← i+1

**fintantque**

**si** x = tab[i] **alors**

retourner i

**finsi**

retourner MAX+1

**fin**

# Algorithme de recherche

---

**fonction** rechercheDicoTrie (tab : Tableau[0..MAX] d'Éléments, x : Élément) :  
Naturel

**Déclaration** gauche, droit, milieu : Naturel

**début**

gauche ← 0; droit ← MAX

**tant que** gauche ≤ droit **faire**

    milieu ← (gauche + droit) div 2

**si** x = tab[milieu] **alors retourner** milieu **finsi**

**si** x < tab[milieu] **alors**

        droit ← milieu - 1

**sinon**

        gauche ← milieu + 1

**finsi**

**fintantque**

**retourner** MAX + 1

**fin**

# Exemple

---

- On cherche 101 dans  $\langle 20, 30, 47, 63, 101, 115 \rangle$ .
- $i=1$   $\langle 20(g), 30, 47(m), 63, 101, 115(d) \rangle$ .
- $i=2$   $\langle 20, 30, 47, 63(g), 101(m), 115(d) \rangle$ .
- et on renvoi l'indice de 101.