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@ensicaen.fr

ENSICAEN
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE

Plan...

- Les algortihmes de tri
  - Définition d'un algorithme de tri,
  - Le tri par minimum successifs,
  - Le tri a 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 ← a
    a ← b
    b ← temp
fin

Tri par minimum successif...

- Principe
  - Le tri par minimum successif est
    - 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 <101, 115, 30, 63, 47, 20>, on va avoir les boucles suivantes :

- i=1 <101, 115, 30, 63, 47, 20>
- i=2 <20, 115, 30, 63, 47, 101>
- i=3 <20, 30, , 63, 47, 101>
- i=4 <20, 30, , 63, 115, 101>
- i=5 <20,30, 47, 63, 115, 101>
- Donc en sortie : <20, 30, 47, 63, 101, 155>

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 ← rang

pour i ←rang+1 **à** nbElements **faire**

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

finsi

finpour

retourner indiceCherche

fin

Tri par minimum successif...

- L'algorithme de tri est donc :
  procédure effectuerTriParMimimumSuccessif **(**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
  $\longrightarrow$ Parcours du tableau.

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 <101, 115, 30, 63, 47, 20>, on va avoir les boucles suivantes :

- i=1 <101, 115, 30, 63, 47, 20>
  - <101, 115, 30, 63, 20, 47>
  - <101, 115, 30, 20, 63, 47>
  - <101, 115, 20, 30, 63, 47>
  - <101, 20, 115, 30, 63, 47>
- i=2 <20, 101, 115, 30, 63, 47>
- i=3 <20, 30,101, 115, 47, 63>
- i=4 <20, 30,47,101, 115, 63>
- i=4 <20, 30, 47, 63, 101, 115>
- Donc en sortie : <20, 30, 47, 63, 101, 155>

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

            finsi
        finpour
    finpour
fin
```

Tri à bulles : Complexités

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

  Compléxité en $\mathcal{O}(n^2)$.

- Nombre d'échanges (pire des cas):

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

- 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
    - à droite du pivot.

La partition

procédure partition **(**E/S t: Tableau[1..MAX] d'**Entier**; E :premier, dernier : **Naturel**, S: indPivot: **Naturel)**

Déclaration compteur, i : **Naturel**,pivot: **Entier**

début

compteur $\leftarrow$ premier

pivot $\leftarrow$ t[premier]

pour i $\leftarrow$premier+1 **à** dernier **faire**

si $t(i) < pivot$ **alors**

compteur $\leftarrow$ compteur+1

echange(t[i],t[compteur]);

finsi

finpour

echanger(T,compteur,premier)

indPivot $\leftarrow$ compteur

fin

Exemple de partition

$6^{(c)}$	$3^{(i)}$	0	9	1	7	8	2	5	4
6	$3^{(i,c)}$	0	9	1	7	8	2	5	4
6	3	$0^{(i,c)}$	9	1	7	8	2	5	4
6	3	$0^{(c)}$	$9^{(i)}$	1	7	8	2	5	4
6	3	$0^{(c)}$	9	$1^{(i)}$	7	8	2	5	4
6	3	0			7	8	2	5	4
6	3	0	$1^{(c)}$	9	7	8	$2^{(i)}$	5	4
6	3	0	1	$2^{(c)}$	7	8	$9^{(i)}$	5	4
6	3	0	1	2	$5^{(c)}$	8	9	$7^{(i)}$	4
6	3	0	1	2	5	$4^{(c)}$	9	7	$8^{(i)}$

4	3	0	1	2	5	$6(c)$	9	7	8

Le tri rapide

- Algorithme :

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

 Déclaration pivot : **Naturel**

début

 si gauche$<$droite **alors**

  partition(t,gauche,droite,pivot)

  triRapide(t,gauche,pivot-1)

  triRapide(t,pivot+1,droite)

 finsi

fin

Exemple. . .

Dans l'exemple précédent on passe de :$<6,3,0,9,1,7,8,2,5,4>$ à $<4,3,0,1,2,6,8,9,7>$ et on se relance sur :

- $<4,3,0,1,2>$ et
- $<8,9,7>$

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.

  $$T(n) = n + T(p) + T(q)$$

  p, q taille des sous tableaux gauche et droits.
  Dans le meilleur des cas $p = q$ et:

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

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

  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 $\leftarrow$ 0

  tant que (i$\leq$MAX) et (tab[i] $\neq$ x) **faire**

    i $\leftarrow$ 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+1] d'Éléments, x : Élément**)**
: Naturel

  Déclaration i : Naturel

début
  i ← 0

  tant que x>tab[i] **faire**
    i ← i+1
  fintantque
  retourner i
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

        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 <20, 30, 47, 63, 101, 115>.
- i=1 <20(g), 30, 47(m), 63, 101, 115(d)>.
- i=2 <20, 30, 47, 63(g), 101(m), 115(d)>.
-