

Chapitre 6 :

Algorithmes de tri et de recherche

1

Tableaux: deux problèmes classiques

- Les tableaux permettent de stocker plusieurs éléments de même type au sein d'une seule entité
- Il faut cependant savoir retrouver ou ranger les éléments dans cette entité, c'est le but **des algorithmes de recherche et de tri**.
- **La recherche** : consiste à trouver un élément dans un tableau.
 - ✓ l'élément peut ne pas être présent
 - ✓ l'élément peut être présent à plusieurs endroits
 - ✓ si le tableau a plusieurs dimensions, il faut fouiller chaque dimension
- **Le tri** consiste à ordonner les éléments du tableau dans l'ordre croissant ou décroissant

Tableaux: deux problèmes classiques

- Recherche d'un élément dans un tableau
 - ✓ Recherche séquentielle
 - ✓ Recherche dichotomique
- Tri d'un tableau
 - ✓ Tri par sélection
 - ✓ Tri à bulle
 - ✓ Tri par insertion

Recherche Séquentielle

```
Tableau T[100]: Entier
Variables i, x, N: Entier
    Trouve: Booléen
Début
...
Trouve ← Faux
i ← 0
TantQue ( i < N  ET Trouve = Faux )
    Si ( T[i] = x ) alors
        Trouve ← Vrai
    FinSi
    i ← i + 1
FinTantQue
Si ( Trouve = vrai ) Alors
    Ecrire ( x, "appartient au tableau" )
Sinon
    Ecrire ( x, "n'appartient pas au tableau" )
FinSi
Fin
```

**Saisie d'un tableau
de taille N**

Consiste à
comparer
successivement
les éléments du
tableau avec la
valeur à
chercher

Recherche dichotomique

Dans le cas où le tableau est ordonné, on peut améliorer l'efficacité de la recherche en utilisant la **recherche dichotomique**

Principe : diviser par 2 le nombre d'éléments dans lesquels on cherche la valeur x à chaque étape de la recherche. Pour cela on compare x avec $T[\text{milieu}]$:

- ✓ Si $x < T[\text{milieu}]$, il suffit de chercher x dans la 1^{ère} moitié du tableau entre ($T[0]$ et $T[\text{milieu}-1]$)
- ✓ Si $x > T[\text{milieu}]$, il suffit de chercher x dans la 2^{ème} moitié du tableau entre ($T[\text{milieu}+1]$ et $T[N-1]$)
- ✓ Si $x = T[\text{milieu}]$, on arrête la recherche.

Recherche dichotomique

On considère le tableau suivant avec $x=11$

X

11

T

3	7	8	8	11	15	20	24
0	1	2	3	4	5	6	7

$Milieu = (0+7)/2 = 3$ (On effectue la division entière)

➡ $x > T[milieu]$, On continue donc la recherche dans **la 2ème moitié**

3	7	8	8	11	15	20	24
0	1	2	3	4	5	6	7

$Milieu = (4+7)/2 = 5$

➡ $x < T[milieu]$, On continue donc la recherche dans **la 1ère moitié**

3	7	8	8	11	15	20	24
0	1	2	3	4	5	6	7

➡ $x = T[milieu]$, on **arrête**.

Algorithme Recherche dichotomique

```
inf ← 0
sup ← N-1
Trouve ← Faux
TantQue ((inf ≤ sup) ET (Trouve = Faux) )
    milieu ← (inf + sup) / 2
    Si (T[milieu] = x) Alors
        Trouve ← Vrai
    Sinon
        Si (T[milieu] < x) Alors
            inf ← milieu + 1
        Sinon
            sup ← milieu - 1
        FinSi
    FinSi
FinTantQue
Si (Trouve = vrai) alors
    Ecrire ("x appartient au tableau")
Sinon
    Ecrire ("x n'appartient pas au tableau")
FinSi
```

Tri d'un tableau

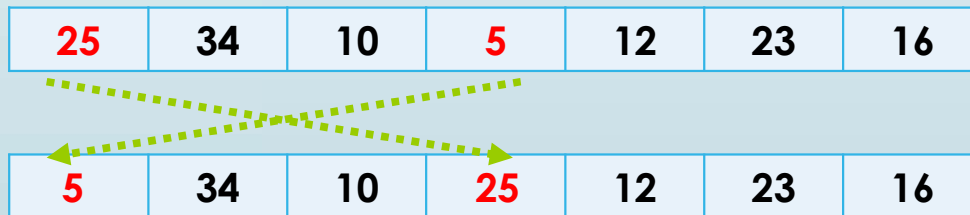
- Le tri consiste à ordonner les éléments du tableau dans un ordre croissant ou décroissant
- Il existe plusieurs algorithmes connus pour trier les éléments d'un tableau :
 - ✓ Le tri par sélection
 - ✓ Le tri par bulle
 - ✓ Le tri par insertion
 - ✓ Le tri par comptage
 - ✓ ...

Tri par sélection

Principe

Consiste à sélectionner successivement l'élément minimal parmi ceux restant. Il fonctionne de la manière suivante :

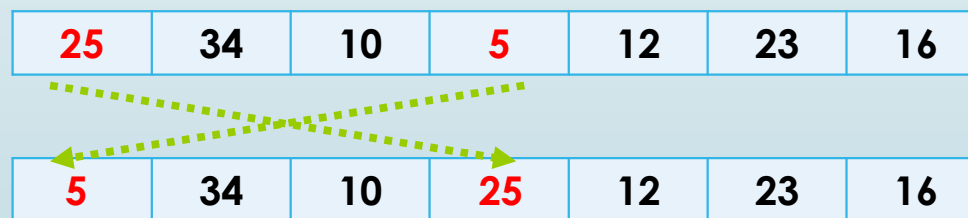
- On cherche le plus petit élément du tableau et on le place à la première position
- Après, on cherche le plus petit élément dans les (N-1) qui restent et on le place en deuxième position, et ainsi de suite



25	34	10	5	12	23	16
5	34	10	25	12	23	16

Tri par sélection

Objectif : C'est d'aller chercher le plus petit élément du tableau pour le mettre en premier, puis de recommencer à partir du second, d'aller chercher le plus petit élément pour le mettre en second etc...



Tri par sélection

Exemple :

9	6	2	8	5
---	---	---	---	---

- **Etape 1:** on cherche le plus petit parmi les 5 éléments du tableau. On l'identifie en troisième position, et on l'échange alors avec l'élément 1 :

2	6	9	8	5
---	---	---	---	---

- **Etape 2:** on cherche le plus petit élément, mais cette fois à partir du deuxième élément. On le trouve en dernière position, on l'échange avec le deuxième:

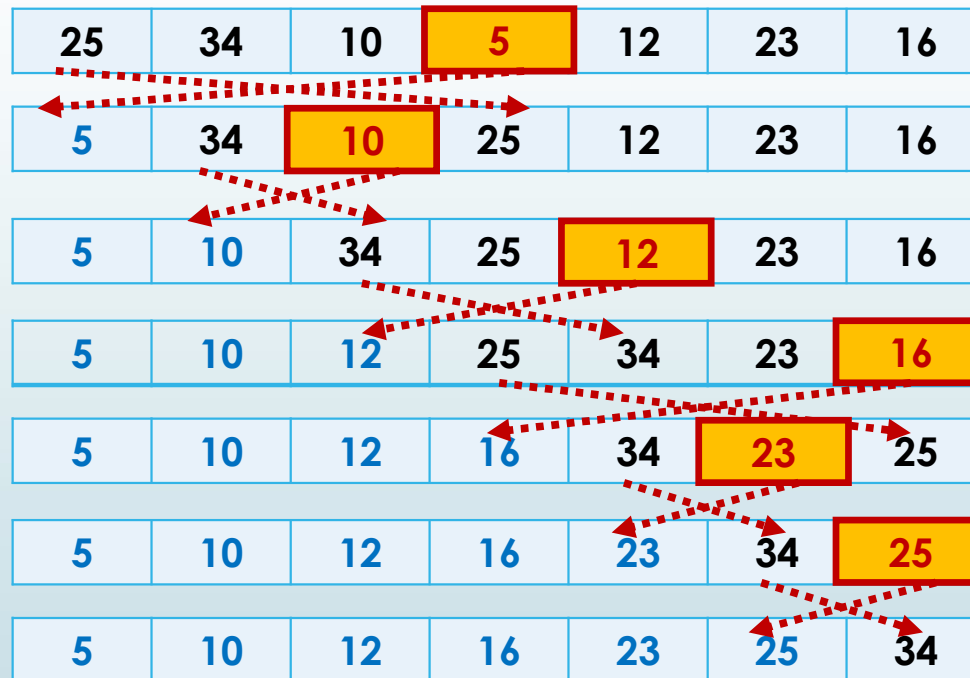
2	5	9	8	6
---	---	---	---	---

- **Etape 3:** on cherche le plus petit élément, mais cette fois à partir du troisième élément. On le trouve en dernière position, on l'échange avec le troisième:

2	5	6	8	9
---	---	---	---	---

Tri par sélection

Exemple



Tri par sélection

```
Pour i=0 à N-2
  posmin ← i
  Pour j=i à N-1
    Si (T[j] < T[posmin]) alors
      posmin ← j
    Finsi
  FinPour
  z ← T[posmin]
  T[posmin] ← T[i]
  T[i] ← z
FinPour
```

Tri par bulle

Principe

- Le tri à bulles est une variante du tri par sélection. Il consiste à remonter petit à petit un élément trop grand vers la fin du tableau en comparant les éléments deux à deux
- Parcourir les éléments du tableau de gauche à droite:
 - ✓ Dès que l'on rencontre deux éléments consécutifs qui ne sont pas dans le bon ordre ($T[i] > T[i+1]$), on les échange
 - ✓ Recommencer tant qu'il y a un changement d'éléments à effectuer

Tri par bulle

5	1	4	2	8
---	---	---	---	---

- **Itération 1 :**

- **Etape 1:** Comparer 1er élément avec 2ème . Si 1er > 2ème , échanger les deux éléments

1	5	4	2	8
---	---	---	---	---

- **Etape 2:** Comparer 2ème élément avec 3ème . Si 2ème > 3ème , échanger les deux éléments

1	4	5	2	8
---	---	---	---	---

- **Etape 3:** Comparer 3ème élément avec 4ème . Si 3ème > 4ème , échanger les deux éléments

1	4	2	5	8
---	---	---	---	---

-

- **Itération ... :** Recommencer à partir du début tant que vous avez opéré au moins un échange

1	2	4	5	8
---	---	---	---	---

Tri par bulle

Exemple

- Itération 1

52	10	1	25
10	52	1	25
10	1	52	25
10	1	25	52

- Itération 2

10	1	25	52
1	10	25	52

- Itération 3

1	10	25	52
---	----	----	----

Tri par bulle (version 1)

Répéter

$np \leftarrow 0$

Pour $i=0$ à $N-2$

Si $(T[i] > T[i+1])$ **alors**

$np \leftarrow np + 1$

$z \leftarrow T[i]$

$T[i] \leftarrow T[i+1]$

$T[i+1] \leftarrow z$

Finsi

FinPour

Jusqu'à $(np=0)$

Tri par bulle (version 2)

```
Permut ← vrai
TantQue (Permut=vrai)
  Permut ← Faux
  Pour i de 1 à N-2
    Si t[i]>t[i+1] alors
      temp ← t[i]
      t[i] ← t[i+1]
      t[i+1] ← temp
      Permut ← Vrai
    FinSi
  FinPour
FinTantQue
```

Tri par insertion

Le tri par insertion consiste à sélectionner un élément du tableau et à l'insérer directement à la bonne position dans la partie du tableau déjà triée. On procède en trois étapes :

- On place l'élément à trier dans une variable temporaire.
- Tant que les éléments du tableau qui précèdent l'élément à trier lui sont supérieurs, on décale ces éléments d'une position en récupérant l'espace vide laissé par l'élément à trier.
- On insère ensuite la variable temporaire à la nouvelle position laissée vacante par le décalage.

Tri par insertion : comment ça marche ?

Analogie : comment trier son jeu de cartes ?

1. je suppose que les cartes que j'ai en main sont déjà triées
2. je prend une nouvelle carte
3. je parcours les cartes que j'ai en main jusqu'à trouver sa position
4. j'insère
5. je recommence
6. on peut commencer avec aucune carte en main



Tri par insertion

Étape 1 : le deuxième élément 17 est placé dans une variable temporaire qui est comparée aux éléments qui le précèdent. Chacun est décalé tant qu'il est supérieur à l'élément à trier.

48	17	25	9	34			48	25	9	34			17	48	25	9	34
17 en temporaire					Décalage de 48					17 à la nouvelle position							

Étape 2 : 25 est comparé aux éléments qui le précèdent et chacun est décalé jusqu'à ce que l'élément ne soit plus supérieur au troisième.

17	48	25	9	34		17		48	9	34	17	25	48	9	34
25 en temporaire						Décalage de 48					25 à la nouvelle position				

Étape 3 : 9 est comparé aux éléments qui le précèdent. Ici comme dans l'étape 1 on s'arrête forcément au premier élément.

17	25	48	9	34			17	25	48	34		9	17	25	48	34
9 en temporaire					Décalage de 17, 25 et 48					9 à la nouvelle position						

Étape 4 : 34 est comparé aux éléments qui le précèdent. Seul 48 lui est supérieur.

9	17	25	48	34		9	17	25		48	9	17	25	34	48
34 en temporaire						Décalage de 48					34 à la nouvelle position				

Tri par insertion

```
Pour i=1 à N-1
  x ← T[i]
  j ← i
  TantQue (j>0 et T[j-1]>x)
    T[j] ← T[j-1]
    j ← j-1
  FinTantQue
  T[j] ← x
FinPour
```