



Langage et algorithmique CR3. Tableaux et Algo de tris

A. Malek TOUMI

toumiab@ensta-bretagne.fr

2017/2018

ENSTA Bretagne



Rappel :

Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Rappel :

Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Rappel :

Les types intégrés

toggle

reset

- Les types simples :

- Entiers signés (**int**), Réels (IEEE 754) (**float**) et Complexes (**complex**), Booléens (**bool**)
- => tous les types simples sont non modifiables



Rappel :

Les types intégrés

toggle

reset

- Les types simples :
 - Entiers signés (**int**), Réels (IEEE 754) (**float**) et Complexes (**complex**), Booléens (**bool**)
 - => tous les types simples sont non modifiables
- Les types composites (containers) :
 - Les séquences : Chaînes de caractères (**str**) ; Listes (**list**) et Tuples (**tuple**)
 - Les maps (hashs) : Dictionnaires (**dict**)
 - Les ensembles : le type **set** et le type **frozenset**



Rappel :

Les types intégrés

toggle

reset

- Les types simples :
 - Entiers signés (**int**), Réels (IEEE 754) (**float**) et Complexes (**complex**), Booléens (**bool**)
 - => tous les types simples sont non modifiables
- Les types composites (containers) :
 - Les séquences : Chaînes de caractères (**str**) ; Listes (**list**) et Tuples (**tuple**)
 - Les maps (hashs) : Dictionnaires (**dict**)
 - Les ensembles : le type **set** et le type **frozenset**
 - => Les types **str**, et **tuple** ne sont pas modifiables
 - => Les types **list**, et **dict** sont modifiables



Rappel :

toggle

reset

Les types intégrés

- Les types simples :
 - Entiers signés (**int**), Réels (IEEE 754) (**float**) et Complexes (**complex**), Booléens (**bool**)
 - => tous les types simples sont non modifiables
- Les types composites (containers) :
 - Les séquences : Chaînes de caractères (**str**) ; Listes (**list**) et Tuples (**tuple**)
 - Les maps (hashs) : Dictionnaires (**dict**)
 - Les ensembles : le type **set** et le type **frozenset**
 - => Les types **str**, et **tuple** ne sont pas modifiables
 - => Les types **list**, et **dict** sont modifiables

Le type tableau/matrice

- Pas de type tableau en Python



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Les tableaux

[toggle](#)[reset](#)

Définition (Tableau)

Structure de données contenant plusieurs éléments du **même type**

- Les listes (list)
- Utilisation des tableaux (array) de Numpy, Scipy Matplotlib
- => Convertir une liste en tableau avec les fonctions **array()**, **matrix()** de Numpy



Les tableaux

toggle

reset

Définition (Tableau)

Structure de données contenant plusieurs éléments du **même type**

- Les listes (list)
- Utilisation des tableaux (array) de Numpy, Scipy Matplotlib
- => Convertir une liste en tableau avec les fonctions **array()**, **matrix()** de Numpy

Exemple (Création d'un tableau Numpy)

```
import numpy as np          # import avec alias
v = np.array([.2, 4 ,5])    # crée un vecteur de 3 réels
v = np.array(range(10), dtype = np.float) # crée un vecteur
                                         # de 10 réels
m = np.array([[0,2],[np.pi,5]]) # création d'une matrice
                               # de 2 lignes et 2 colonnes de réels
```



Les tableaux

[toggle](#)[reset](#)

Exemple (Création d'un tableau Numpy)

```
import numpy as np
np.zeros((3,2)) # crée un tableau de 0.0
                  # de 3 lignes et 2 colonnes
np.ones((3,2))  # crée un tableau de 1.0
M = np.linspace(0,1,11) # renvoie [0,0.1,..., 1.0]
N = np.arange(2,5)       # renvoie [2,3,4]
```



Opérations

[toggle](#)[reset](#)

- Opération élément pas élément : addition ($A+B$), produit ($A*B$), puissance ($A^{**}B$)
exemple : `tab+2`; `tab*2`; `tab**3`; `tab + tab*2`; `tab**tab`
- Produit matriciel : `dot(A,B)`, `A@B`
- Concaténation : `np.concatenate((A,B))`.
exemple : `np.concatenate((tab,tab), axis=0)`

Propriétés

Les dimensions des vecteurs-matrices doivent être conformes à l'opération souhaitée.



Copie d'un tableau

[toggle](#)[reset](#)

Copie de tableau

```
import numpy as np
tab = np.array(range(10))
tab2 = tab # tab et tab2 référencent la même
          # zone mémoire
# Solutions à retenir
tab2 = np.copy(tab)
tab2 = tab.copy()
```



Accès aux éléments d'un tableau

toggle

reset

Remarque

Utiliser la vectorisation quand c'est possible

Exemple ($\sin(t)$ pour $t \in [0, 999999]$)



Accès aux éléments d'un tableau

togglereset

Remarque

Utiliser la vectorisation quand c'est possible

Exemple ($\sin(t)$ pour $t \in [0, 999999]$)

```
import numpy as np
res []
for t in range(1000000):
    res += [np.sin(t)]
```



Accès aux éléments d'un tableau

togglereset

Remarque

Utiliser la vectorisation quand c'est possible

Exemple ($\sin(t)$ pour $t \in [0, 999999]$)

```
import numpy as np
res []
for t in range(1000000):
    res += [np.sin(t)]

t = np.arange(1000000)
res = np.sin(t)
```



Accès aux éléments des tableaux

togglereset

- Accéder à un élément = accéder à une case du tableau
- Opérateur []
- Première case : 0
- Tableaux multidimensionnels : plusieurs indices ex. $M[i,j]$ ou $M[i][j]$
- Taille du tableau : **shape(tab)**, **len(tab)**, **len(tab[0])**, ...

Exemple (Accès aux éléments des tableaux)

```
L = np.array([2, 4 ,5])
L[1]          # renvoie 4
len(L)        # renvoie 3
M = np.array([[0,1,5],[2,7,10]])
M[1,1] ; M[1][1] # renvoient 7
M[0,:] ; M[0][:] # renvoient la première ligne
```



Accès aux éléments des tableaux

[toggle](#)[reset](#)

- Accéder à un élément = accéder à une case du tableau
- Opérateur []
- Première case : 0
- Tableaux multidimensionnels : plusieurs indices ex. $M[i,j]$ ou $M[i][j]$
- Taille du tableau : **shape(tab)**, **len(tab)**, **len(tab[0])**, ...

Exemple (Accès aux éléments des tableaux)

```
L = np.array([2, 4 ,5])
L[1]          # renvoie 4
len(L)        # renvoie 3
M = np.array([[0,1,5],[2,7,10]])
M[1,1] ; M[1][1] # renvoient 7
M[0,:] ; M[0][:] # renvoient la première ligne
len(M)         # renvoie 2
len(M[0])      # renvoie 3
```



Accès aux éléments des tableaux

[toggle](#)[reset](#)

- Accéder à un élément = accéder à une case du tableau
- Opérateur []
- Première case : 0
- Tableaux multidimensionnels : plusieurs indices ex. $M[i,j]$ ou $M[i][j]$
- Taille du tableau : **shape(tab)**, **len(tab)**, **len(tab[0])**, ...

Exemple (Accès aux éléments des tableaux)

```
L = np.array([2, 4 ,5])
L[1]          # renvoie 4
len(L)        # renvoie 3
M = np.array([[0,1,5],[2,7,10]])
M[1,1] ; M[1][1] # renvoient 7
M[0,:] ; M[0][:] # renvoient la première ligne
np.shape(M)      # renvoie (2,3)
```



Accès aux éléments des tableaux

togglereset

- Parcourir un tableau à l'aide d'une boucle

Exemple

```
import numpy as np
tab = np.array(range (10))
for i in range (len (tab)):
    tab[i] = ....
print (tab)
```



Accès aux éléments des tableaux

togglereset

- Parcourir un tableau à l'aide d'une boucle

Exemple

```
import numpy as np
tab = np.array(range (10))
for i in range (len (tab)):
    tab[i] = ....
print (tab)

for i, val  in enumerate (tab):
    tab[i] = val + ...
print(tab)
```



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Mélange d'un tableau

[toggle](#)[reset](#)

Algorithme 1: Mélange()

Entrées : tab : entier[]

Données : i, n : entier

$n \leftarrow \text{taille}(\text{tab}) - 1$

tant que $n > 0$ **faire**

$i \leftarrow$ nombre aléatoire entre 0 et n inclus

permuter $\text{tab}[n]$ et $\text{tab}[i]$

décrémenter n

fin



Mélange d'un tableau

[toggle](#)[reset](#)

```
import numpy as np
n = len(tab) - 1
while n>0 :
    # générer un nombre aléatoire entre 0 et
    # n inclus
    i = np.random.randint(0, n)
    tab[i], tab[n] = tab[n], tab[i]
    n -= 1
```



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Exercice

[toggle](#)[reset](#)

```
>>>import numpy as np  
>>>tab = np.array([[1, 2, 3], [4, 5]])  
>>>tab? , np.shape(tab) ?
```



Exercice

[toggle](#)[reset](#)

```
>>>import numpy as np  
>>>tab = np.array([[1, 2, 3], [4, 5]])  
>>>tab? , np.shape(tab)?  
tab = erreur?
```



Exercice

[toggle](#)[reset](#)

```
>>>import numpy as np  
>>>tab = np.array([[1, 2, 3], [4, 5]])  
>>>tab? , np.shape(tab)?  
array([[1, 2, 3],  
       [4, 5, 0]] , (2,3)
```



Exercice

[toggle](#)[reset](#)

```
>>>import numpy as np  
>>>tab = np.array([[1, 2, 3], [4, 5]])  
>>>tab? , np.shape(tab)?  
array([[1, 2, 3], [4, 5]]), (2,)
```



Exercice

[toggle](#)[reset](#)

```
>>>import numpy as np  
>>>tab = np.array([[1, 2, 3], [4, 5]])  
>>>tab? , np.shape(tab)?  
# solution :  
array([[1, 2, 3], [4, 5]], dtype=object), (2,)
```



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

7	2	1	8	4
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

7	2	1	8	4
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	7	8	4
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	7	8	4
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	7	8	4
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	7	8	4
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	4	8	7
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	4	8	7
---	---	---	---	---



Principe du tri par sélection

toggle

reset

- Rechercher le plus petit élément du tableau
- Le permutez avec le premier élément
- Recommencez entre le deuxième plus petit et le deuxième élément

Exemple

1	2	4	7	8
---	---	---	---	---



Principe du tri par sélection

toggle

reset

Exemple

18	3	10	25	9	3	11	13	23	8
	i	m							

3	18	10	25	9	3	11	13	23	8
	i				m				

3	3	10	25	9	18	11	13	23	8
	i					m			

3	3	8	25	9	18	11	13	23	10
	i	m							

3	3	8	9	25	18	11	13	23	10
	i			m					

3	3	8	9	10	18	11	13	23	25
	i			m					

3	3	8	9	10	11	18	13	23	25
	i			m					

3	3	8	9	10	11	13	18	23	25
	i			m					

3	3	8	9	10	11	13	18	23	25
	i			m					

3	3	8	9	10	11	13	18	23	25
	i			m					



Algorithme du tri par sélection

togglereset

Algorithme 2: triSel

Entrées : tab : entier[]

Données : indMin, i, j, n : entier

$n \leftarrow \text{taille}(\text{tab})$;

pour $i \in [0, n - 2]$ **faire**



Algorithme du tri par sélection

togglereset

Algorithme 3: triSel

Entrées : tab : entier[]

Données : indMin, i, j, n : entier

$n \leftarrow \text{taille}(tab)$;

pour $i \in [0, n - 2]$ **faire**

$indMin \leftarrow i$;

pour $j \in [i + 1, n - 1]$ **faire**

si $tab[j] < tab[indMin]$ **alors**

$indMin \leftarrow j$;



Algorithme du tri par sélection

toggle

reset

Algorithme 4: triSel

Entrées : tab : entier[]

Données : indMin, i, j, n : entier

$n \leftarrow \text{taille}(tab)$;

pour $i \in [0, n - 2]$ **faire**

$indMin \leftarrow i$;

pour $j \in [i + 1, n - 1]$ **faire**

si $tab[j] < tab[indMin]$ **alors**

$indMin \leftarrow j$;

fin

fin

permuter $tab[indMin]$ et $tab[i]$;

fin



Algorithme tri par sélection

toggle

reset

```
def trisel(tab):
    """ tri par sélection: paramètres
    tab : numpy.array
        tableau à trier.
    """
    for i in range(len(tab)-1):
        # Recherche de la position du
        # plus petit élément
        i_min = i
        for j in range(i+1, len(tab)):
            if tab[j] < tab[i_min]:
                i_min = j
        # Permuter les cases i et i_min
        tab[i], tab[i_min] = tab[i_min], tab[i]
```



Algorithme tri par sélection

togglereset

Listing 1 – Tri par sélection optimisé pour numpy

```
def trisel(tab):
    """ tri par sélection: paramètres
        tab : numpy.array
            tableau à trier.
    """
    for i in range(len(tab)-1):
        # Recherche de la position du
        # plus petit élément
        i_min = np.argmin(tab[i:]) + i
        # Permuter les cases i et i_min
        tab[i], tab[i_min] = tab[i_min], tab[i]
```



Étude de la complexité du tri

togglereset

Définition

Complexité Ordre de grandeur du nombre d'opérations d'un algorithme. Notation $\Theta(g(n))$.

Tri par sélection :

- Boucle externe : $i = n - 2$ fois
- Boucle interne : $n - i$ fois
- Somme :

$$\sum_{i=0}^{n-2} (n - i) = \sum_{j=1}^{n-1} j = \frac{n(n - 1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$C(n) = \frac{1}{2}n^2 - \frac{1}{2}n$$



Étude de la complexité du tri

[toggle](#)[reset](#)

Définition

Complexité Ordre de grandeur du nombre d'opérations d'un algorithme. Notation $\Theta(g(n))$.

Tri par sélection :

- Boucle externe : $i = n - 2$ fois
- Boucle interne : $n - i$ fois
- Somme :

$$\sum_{i=0}^{n-2} (n - i) = \sum_{j=1}^{n-1} j = \frac{n(n - 1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$C(n) \simeq \frac{1}{2}n^2$$



Étude de la complexité du tri

togglereset

Définition

Complexité Ordre de grandeur du nombre d'opérations d'un algorithme. Notation $\Theta(g(n))$.

Tri par sélection :

- Boucle externe : $i = n - 2$ fois
- Boucle interne : $n - i$ fois
- Somme :

$$\sum_{i=0}^{n-2} (n - i) = \sum_{j=1}^{n-1} j = \frac{n(n - 1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$C(n) \simeq n^2$$



Étude de la complexité du tri

togglereset

Définition

Complexité Ordre de grandeur du nombre d'opérations d'un algorithme. Notation $\Theta(g(n))$.

Tri par sélection :

- Boucle externe : $i = n - 2$ fois
- Boucle interne : $n - i$ fois
- Somme :

$$\sum_{i=0}^{n-2} (n - i) = \sum_{j=1}^{n-1} j = \frac{n(n - 1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$C(n) \in \Theta(n^2)$$



Étude de la complexité du tri

togglereset

Définition

Complexité Ordre de grandeur du nombre d'opérations d'un algorithme. Notation $\Theta(g(n))$.

Tri par sélection :

- Boucle externe : $i = n - 2$ fois
- Boucle interne : $n - i$ fois
- Somme :

$$\sum_{i=0}^{n-2} (n - i) = \sum_{j=1}^{n-1} j = \frac{n(n - 1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$C(n) = \Theta(n^2)$$



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

7	2	1	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

7	2	1	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	7	1	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	7	1	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	1	7	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	1	7	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	1	7	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	1	7	8	4
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	1	7	4	8
---	---	---	---	---



Principe du tri bulles

[toggle](#)[reset](#)

- Très simple à mettre en œuvre
- Comparaisons locales : un élément et son successeur
- Faire remonter l'élément le plus grand, puis le deuxième plus grand,
...

Exemple

2	1	7	4	8
---	---	---	---	---



Algorithme du tri bulles

[toggle](#)[reset](#)

Algorithme 5: Algorithme du tri bulles

Entrées : tab : entier[]

Données : i, j, n : entier

$n \leftarrow \text{taille}(\text{tab})$;

pour $i \in [n - 1, 1]$ **faire**



Algorithme du tri bulles

[toggle](#)[reset](#)

Algorithme 6: Algorithme du tri bulles

Entrées : tab : entier[]

Données : i, j, n : entier

$n \leftarrow \text{taille}(\text{tab});$

pour $i \in [n - 1, 1]$ **faire**

pour $j \in [0, i - 1]$ **faire**

|

|



Algorithme du tri bulles

[toggle](#)[reset](#)

Algorithme 7: Algorithme du tri bulles

Entrées : tab : entier[]

Données : i, j, n : entier

$n \leftarrow \text{taille}(\text{tab});$

pour $i \in [n - 1, 1]$ **faire**

pour $j \in [0, i - 1]$ **faire**

si $\text{tab}[j] > \text{tab}[j + 1]$ **alors**



Algorithme du tri bulles

[toggle](#)[reset](#)

Algorithme 8: Algorithme du tri bulles

Entrées : tab : entier[]

Données : i, j, n : entier

$n \leftarrow \text{taille}(\text{tab});$

pour $i \in [n - 1, 1]$ **faire**

pour $j \in [0, i - 1]$ **faire**

si $\text{tab}[j] > \text{tab}[j + 1]$ **alors**

permuter $\text{tab}[j]$ et $\text{tab}[j + 1]$;

fin

fin

fin



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties

5	2	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée

5	2	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée

5	2	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé)

5	2	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée

5	2	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée

2	5	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée
- Continuer en agrandissant la partie triée

2	5	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée
- Continuer en agrandissant la partie triée

2	5	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée
- Continuer en agrandissant la partie triée

2	5	6	3	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée
- Continuer en agrandissant la partie triée

2	3	5	6	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée
- Continuer en agrandissant la partie triée

2	3	5	6	1
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Principe : découper le tableau en deux parties
- Une partie triée
- Une partie non triée
- Insérer le premier nombre non trié (clé) dans la partie triée
- Continuer en agrandissant la partie triée

1	2	3	5	6
---	---	---	---	---



Principe du tri par insertion

toggle

reset

- Exemple : Insérer 11, 2, 16, 10 et 1 dans un tableau en cherchant à le trier.

Insertion de 11 :
en première position

0	1	2	3	4
11				

Insertion de 2 :

0	1	2	3	4
11	2			

Permutation nécessaire

0	1	2	3	4
2	11			
		↑		

Insertion de 16 :

0	1	2	3	4
2	11	16		

Insertion de 10 :

0	1	2	3	4
2	11	16	10	

Permutation

0	1	2	3	4
2	11	10	16	

Permutation

0	1	2	3	4
2	10	11	16	

Insertion de 1 :

0	1	2	3	4
2	10	11	16	1

Permutation :

0	1	2	3	4
2	10	11	1	16

Permutation :

0	1	2	3	4
2	10	1	11	16

Permutation :

0	1	2	3	4
1	2	10	11	16

0	1	2	3	4
1	2	10	11	16



algorithme de tris par insertion

togglereset

Algorithme 9: Algorithme du tri par insertion

Entrées : tab : entier[]

Données : i, j, n, cle : entier

$n \leftarrow \text{taille(tab)}$;

pour $i \in [1, n - 1]$ **faire**



algorithme de tris par insertion

togglereset

Algorithme 10: Algorithme du tri par insertion

Entrées : tab : entier[]

Données : i, j, n, cle : entier

$n \leftarrow \text{taille}(\text{tab})$;

pour $i \in [1, n - 1]$ **faire**

$\text{cle} \leftarrow \text{tab}[i]$ // valeur à insérer dans la partie triée

$j \leftarrow i;$



algorithme de tris par insertion

togglereset

Algorithme 11: Algorithme du tri par insertion

Entrées : tab : entier[]

Données : i, j, n, cle : entier

$n \leftarrow \text{taille}(\text{tab})$;

pour $i \in [1, n - 1]$ **faire**

$\text{cle} \leftarrow \text{tab}[i]$ // valeur à insérer dans la partie triée

$j \leftarrow i$;

tant que $j > 0$ et $\text{tab}[j - 1] > \text{cle}$ **faire**

$\text{tab}[j] \leftarrow \text{tab}[j - 1]$ // décaler l'élément j d'un cran à droite

décrémenter j ;



algorithme de tris par insertion

togglereset

Algorithme 12: Algorithme du tri par insertion

Entrées : tab : entier[]

Données : i, j, n, cle : entier

$n \leftarrow \text{taille}(\text{tab})$;

pour $i \in [1, n - 1]$ **faire**

$\text{cle} \leftarrow \text{tab}[i]$ // valeur à insérer dans la partie triée

$j \leftarrow i$;

tant que $j > 0$ et $\text{tab}[j - 1] > \text{cle}$ **faire**

$\text{tab}[j] \leftarrow \text{tab}[j - 1]$ // décaler l'élément j d'un cran à droite

décrémenter j ;

fin

$\text{tab}[j] \leftarrow \text{cle}$ // insérer cle à sa place

fin



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Principe du tri shell

[toggle](#)[reset](#)

- Proposé en 1959 par Donald L. Shell
- Tri par insertion optimisé
- Inconvénient du tri par insertion : insérer une clé en début de tableau
- Idée : faire un “pré-tri” avec un pas grossier h
- puis affiner le tri en diminuant h
- finir avec $h = 1$ (tri par insertion)
- Choix de la suite h_n importante
- exemple de suite : $u_0 = 1; u_{n+1} = 3u_n + 1$, h_n défini par
 - $h_0 = \text{le plus grand } u_n \text{ tel que } u_n < N$ (N : taille du tableau)
 - $h_{n+1} = \frac{h_n - 1}{3}$



Algorithme du tri shell

[toggle](#)[reset](#)

Algorithme 13: Tri Shell

Entrées : tab : entier[]

Données : i, j, n, cle,h : entier

$n \leftarrow \text{taille}(tab)$;

$h \leftarrow 1$;

tant que $3h + 1 < n$ // calcul de h_0

faire

$| \quad h \leftarrow 3h + 1$;

fin

...



Algorithme du tri shell

[toggle](#)[reset](#)

Algorithme 14: Tri Shell

```
tant que  $h > 0$  faire
    pour  $i \in [h, n - 1]$  faire
        cle  $\leftarrow tab[i]$  // valeur à insérer dans la partie triée
        j  $\leftarrow i$ ;
        tant que  $j \geq h$  et  $tab[j - h] > cle$  faire
            tab[j]  $\leftarrow tab[j - h]$  // décaler de  $h$  crans à droite
            j  $\leftarrow j - h$  ;
        fin
        tab[j]  $\leftarrow cle$  // insérer cle à sa place
    fin
    h  $\leftarrow h/3$  // équivalent à  $(h-1)/3$ 
fin
```



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Autres tris

[toggle](#)[reset](#)

Il existe d'autres tris plus performants mais plus complexes.

- tri par segmentation (quicksort)
- tri par partition/fusion
- tri par arbre binaire de recherche équilibré
- tri par tas

Exemples : <https://moodle.ensta-bretagne.fr/course/view.php?id=1100>



Sommaire

1 Rappel :

Les types intégrés

2 Les Tableaux

Création d'un tableau

Mélange d'un tableau

Exercice

3 Algorithmes de tris

Tri par sélection

Tri bulles

Tri par insertion

Tri shell

Autres tris

4 Horloge



Horloge

toggle

reset

