Notion de tri

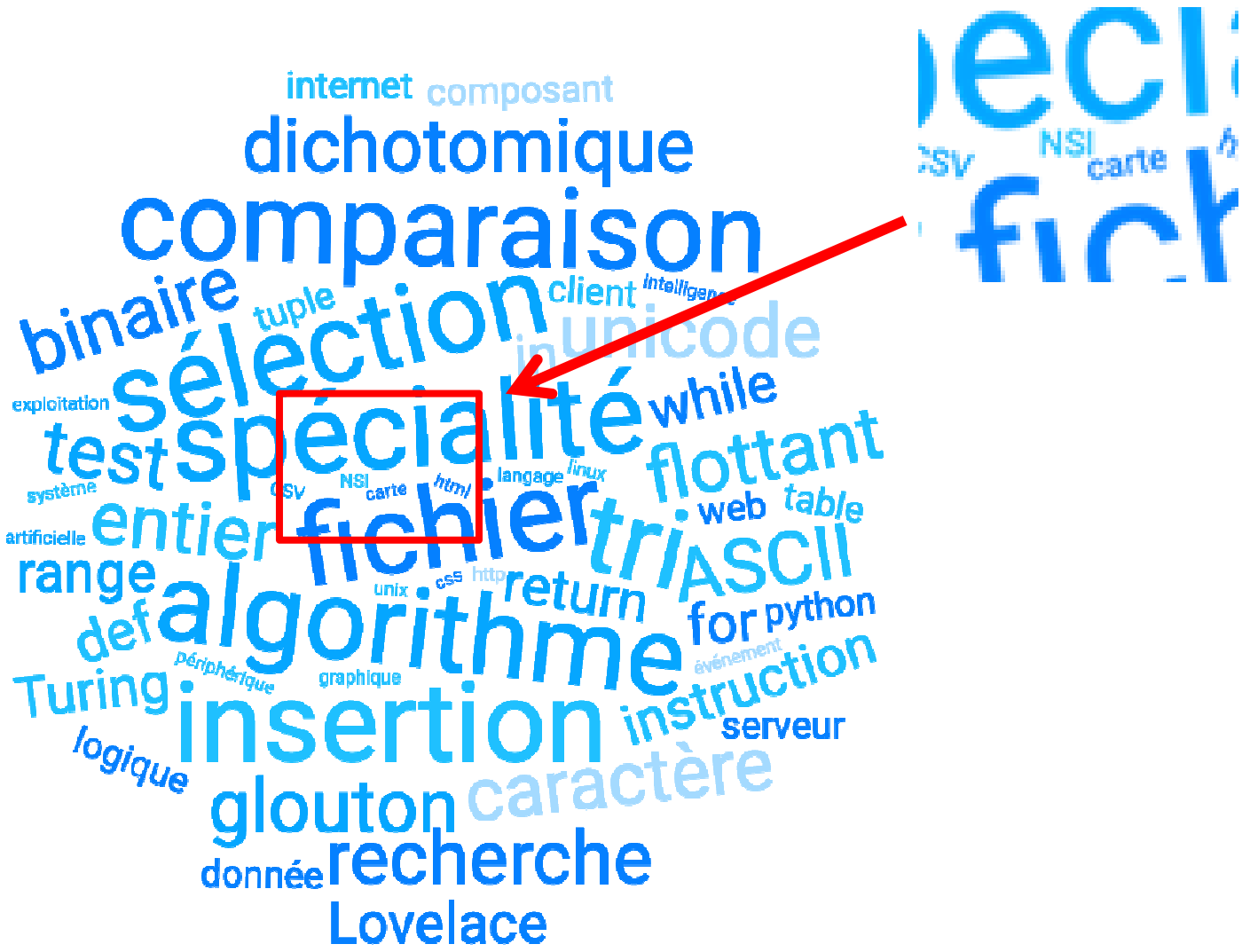
1. Pourquoi trie-t-on ?

2. Le tri par sélection. Principe et programme.

3. Le tri par insertion. Principe et programme.

1. Pourquoi trie-t-on ?


internet
composant
dichotomique
comparaison
binaire
tuple
sélection
client
intelligence
unicode
in
exploitation
spécialité
while
test
flottant
système
CSV
NSI
carte
html
langage
linux
web
table
artificielle
entier
fichier
tri
ASCII
range
unix
css
http
return
for
python
def
algorithme
périphérique
graphique
événement
Turing
instruction
insertion
serveur
logique
caractère
glouton
recherche
donnée
Lovelace


internet
composant
dichotomique
comparaison
binaire
sélection
tuple
client
unicode
spécialité
while
test
flottant
système
CSV
NSI
carte
html
langage
linux
web
table
artificielle
entier
fichier
tri
ASCII
range
def
algorithme
unix
css
http
return
for
python
Turing
périphérique
graphique
événement
instruction
insertion
serveur
logique
caractère
glouton
recherche
donnée
Lovelace

affectation	dichotomique	insertion	python
algorithme	dictionnaire	instruction	range
architecture	donnée	intelligence	recherche
artificielle	entier	internet	réseau
ASCII	étape	javascript	return
binaire	événement	langage	sélection
Boole	exploitation	linux	serveur
booléen	expression	liste	spécialité
bornée	fichier	logique	système
boucle	fichier	Lovelace	table
caractère	flottant	machine	tableau
carte	fonction	maximum	test
client	for	mémoire	tri
communication	glouton	minimum	tuple
comparaison	graphique	NSI	Turing
composant	hexadécimal	ordinateur	unicode
conditionnelle	html	périphérique	unix
css	http	position	variable
CSV	in	processeur	web
def	indice	protocole	while

affectation	dichotomique	insertion	python
algorithme	dictionnaire	instruction	range
architecture	donnée	intelligence	recherche
artificielle	entier	internet	réseau
ASCII	étape	javascript	return
binaire	événement	langage	sélection
Boole	exploitation	linux	serveur
booléen	expression	liste	spécialité
bornée	fichier	logique	système
boucle	fichier	Lovelace	table
caractère	flottant	machine	tableau
carte	fonction	maximum	test
client	for	mémoire	tri
communication	glouton	minimum	tuple
comparaison	graphique	NSI	Turing
composant	hexadécimal	ordinateur	unicode
conditionnelle	html	périphérique	unix
css	http	position	variable
CSV	in	processeur	web
def	indice	protocole	while

Dans la vie courante, les deux verbes *trier* et *classer* ne sont pas synonymes.

Dans la vie courante, les deux verbes *trier* et *classer* ne sont pas synonymes.

Trier ou effectuer un *tri* c'est répartir les éléments en paquets correspondant à un certain critère : par exemple séparer les déchets selon leur nature, les personnes d'une assemblée selon leur sexe ou selon leur langue maternelle.

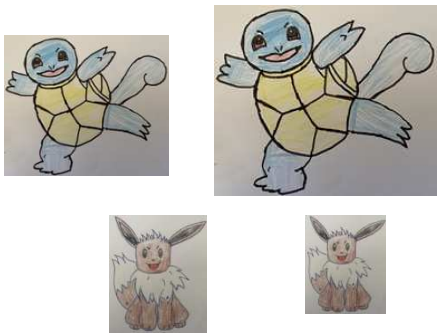
Dans la vie courante, les deux verbes *trier* et *classer* ne sont pas synonymes.

Trier ou effectuer un *tri* c'est répartir les éléments en paquets correspondant à un certain critère : par exemple séparer les déchets selon leur nature, les personnes d'une assemblée selon leur sexe ou selon leur langue maternelle.

Dans la vie courante, les deux verbes *trier* et *classer* ne sont pas synonymes.

Trier ou effectuer un *tri* c'est répartir les éléments en paquets correspondant à un certain critère : par exemple séparer les déchets selon leur nature, les personnes d'une assemblée selon leur sexe ou selon leur langue maternelle.

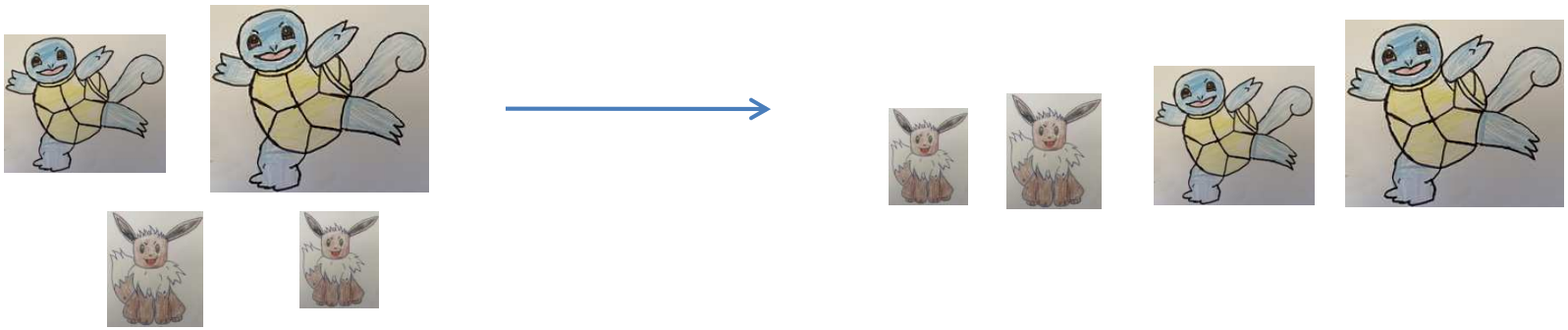
Classer ou effectuer un *classement* c'est mettre des éléments selon un certain ordre : par exemple ranger les personnes d'une assemblée de la plus petite à la plus grande, ou de la plus jeune à la plus âgée.



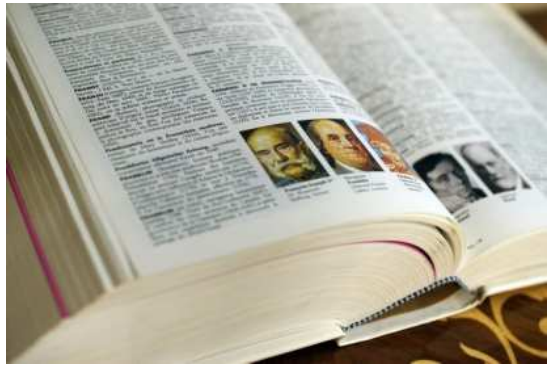
Dans la vie courante, les deux verbes *trier* et *classer* ne sont pas synonymes.

Trier ou effectuer un *tri* c'est répartir les éléments en paquets correspondant à un certain critère : par exemple séparer les déchets selon leur nature, les personnes d'une assemblée selon leur sexe ou selon leur langue maternelle.

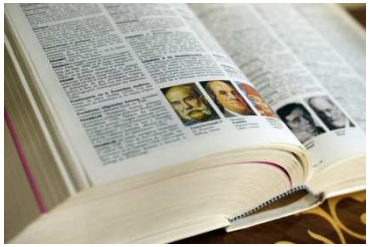
Classer ou effectuer un *classement* c'est mettre des éléments selon un certain ordre : par exemple ranger les personnes d'une assemblée de la plus petite à la plus grande, ou de la plus jeune à la plus âgée.



Dans la vie quotidienne réaliser un tri ou un classement est une opération relativement courante :



Dans la vie quotidienne réaliser un tri ou un classement est une opération relativement courante :



Un tri porte généralement sur un nombre assez important de données.

En informatique les mots *tri* et *trier* sont à prendre avec le sens de *classement* et *classer*.

Trier une liste de nombres entiers.
De nombreux algorithmes de tri existent, plus ou moins efficaces et plus ou moins faciles à mettre en œuvre.

Liste d'entiers dans le désordre.

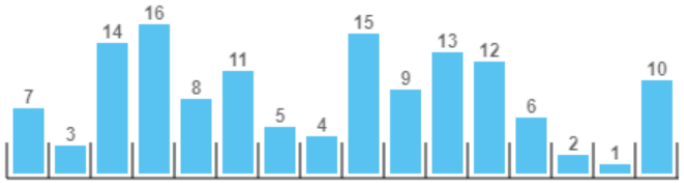
```
[76, 95, 16, 49, 85, 20, 57, 51, 64, 92, 75, 49, 83, 23, 51, 8, 56, 75, 6, 41]
```

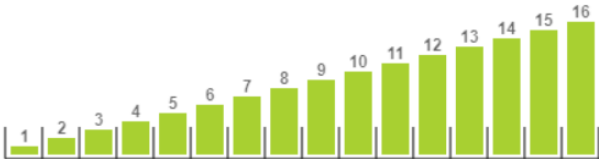
Liste d'entiers dans l'ordre croissant.

```
[6, 8, 16, 20, 23, 41, 49, 49, 51, 51, 56, 57, 64, 75, 75, 76, 83, 85, 92, 95]
```

2. Le tri par sélection

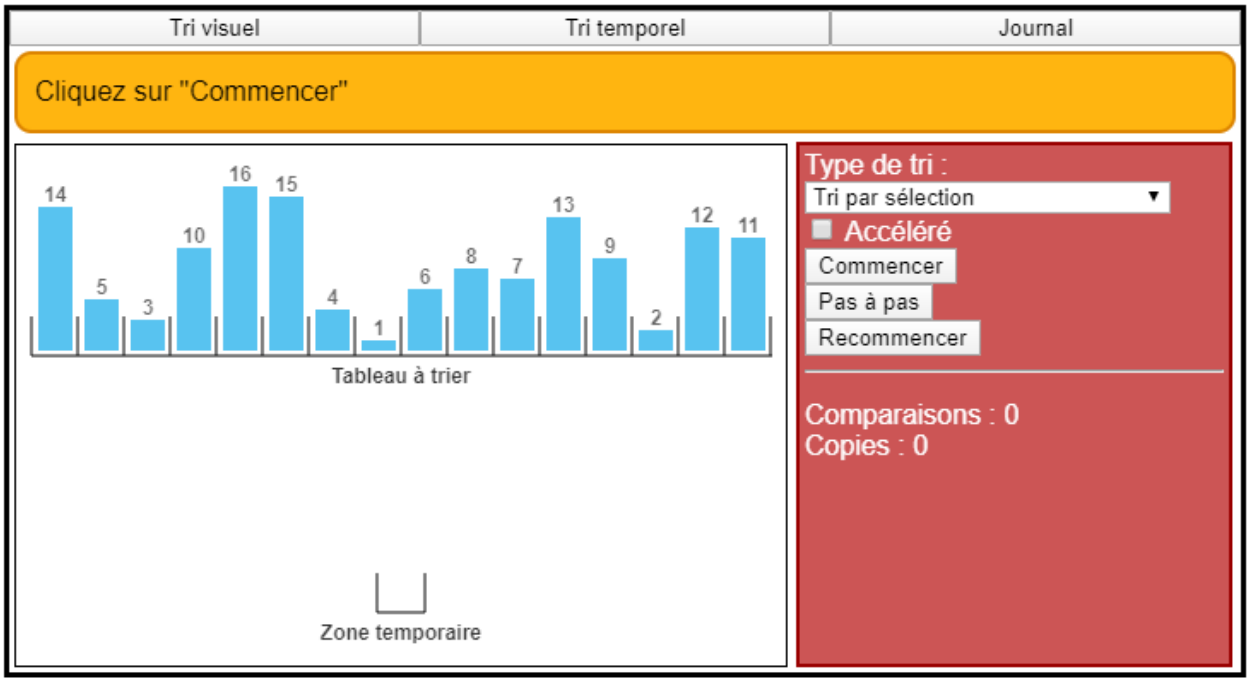
Cette méthode de tri s’apparente à celle utilisée pour trier des copies suivant l’ordre décroissant des notes par exemple.





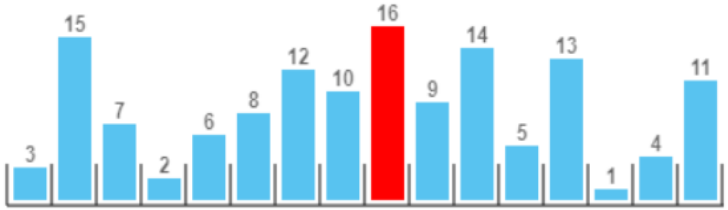
)i(interstices.info

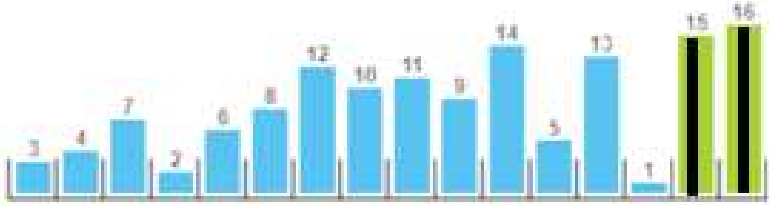
https://interstices.info/les-algorithmes-de-tri/



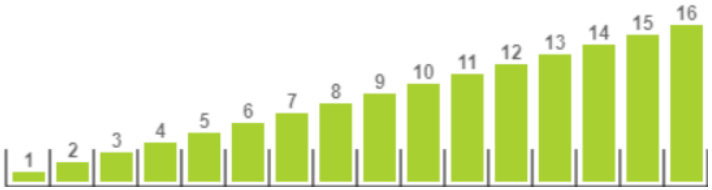
Interstices est une revue de culture scientifique en ligne, créée par des chercheurs pour nous inviter à explorer les sciences du numérique.

On cherche le plus grand entier puis on permute. Le plus grand est bien placé.
On recommence .
Comparaisons.



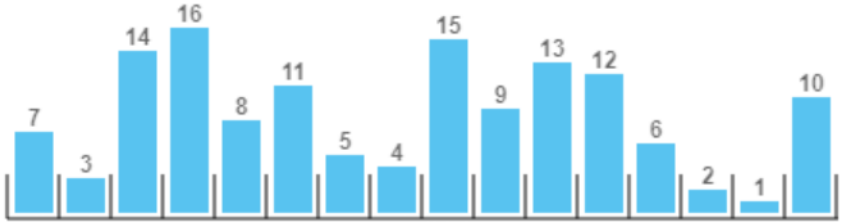


.....



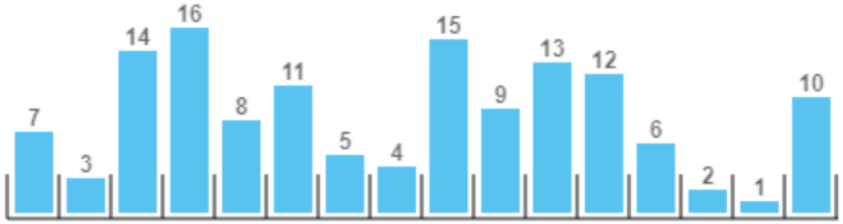
Calcul du nombre de comparaisons

Le tri par sélection




```
def tri_selection(t):
    for i in range(len(t)):
        ind = IndiceMax(t,len(t)-1-i)
        t[len(t)-1-i],t[ind] = t[ind],t[len(t)-1-i]
    return t
```

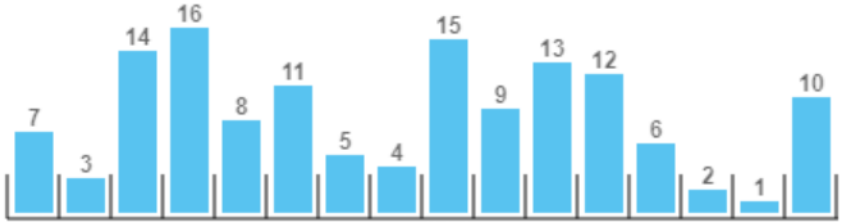
Le tri par sélection




```
def tri_selection(t):
    for i in range(len(t)):
        ind = IndiceMax(t,len(t)-1-i)
        t[len(t)-1-i],t[ind] = t[ind],t[len(t)-1-i]
    return t
```

```
ind = IndiceMax(t,15)
t[15],t[ind] = t[ind],t[15]
```

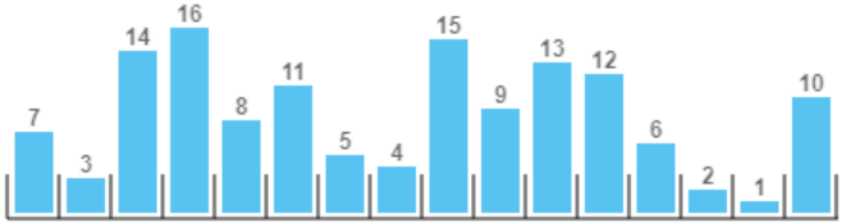
Le tri par sélection




```
def tri_selection(t):
    for i in range(len(t)):
        ind = IndiceMax(t,len(t)-1-i)
        t[len(t)-1-i],t[ind] = t[ind],t[len(t)-1-i]
    return t
```

```
ind = IndiceMax(t,15)
t[15],t[ind] = t[ind],t[15]
ind = IndiceMax(t,14)
t[14],t[ind] = t[ind],t[14]
```

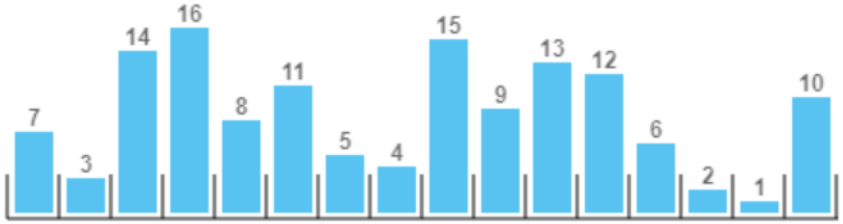
Le tri par sélection




```
def tri_selection(t):
    for i in range(len(t)):
        ind = IndiceMax(t,len(t)-1-i)
        t[len(t)-1-i],t[ind] = t[ind],t[len(t)-1-i]
    return t
```

```
ind = IndiceMax(t,15)
t[15],t[ind] = t[ind],t[15]
ind = IndiceMax(t,14)
t[14],t[ind] = t[ind],t[14]
ind = IndiceMax(t,13)
t[13],t[ind] = t[ind],t[13]
```

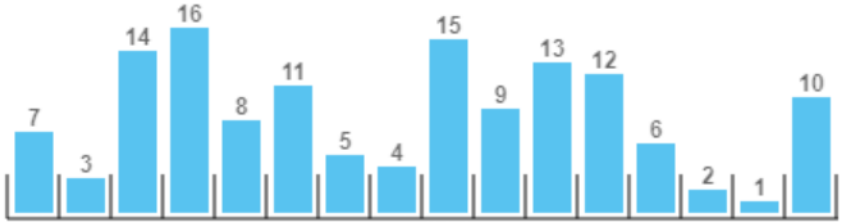
Le tri par sélection




```
def tri_selection(t):
    for i in range(len(t)):
        ind = IndiceMax(t,len(t)-1-i)
        t[len(t)-1-i],t[ind] = t[ind],t[len(t)-1-i]
    return t
```

```
ind = IndiceMax(t,15)
t[15],t[ind] = t[ind],t[15]
ind = IndiceMax(t,14)
t[14],t[ind] = t[ind],t[14]
ind = IndiceMax(t,13)
t[13],t[ind] = t[ind],t[13]
ind = IndiceMax(t,12)
t[12],t[ind] = t[ind],t[12]
```

Le tri par sélection




```
def tri_selection(t):
    for i in range(len(t)):
        ind = IndiceMax(t,len(t)-1-i)
        t[len(t)-1-i],t[ind] = t[ind],t[len(t)-1-i]
    return t
```

```
ind = IndiceMax(t,15)
t[15],t[ind] = t[ind],t[15]
ind = IndiceMax(t,14)
t[14],t[ind] = t[ind],t[14]
ind = IndiceMax(t,13)
t[13],t[ind] = t[ind],t[13]
ind = IndiceMax(t,12)
t[12],t[ind] = t[ind],t[12]
 …
ind = IndiceMax(t,0)
t[0],t[ind] = t[ind],t[0]
```

Le tri par sélection

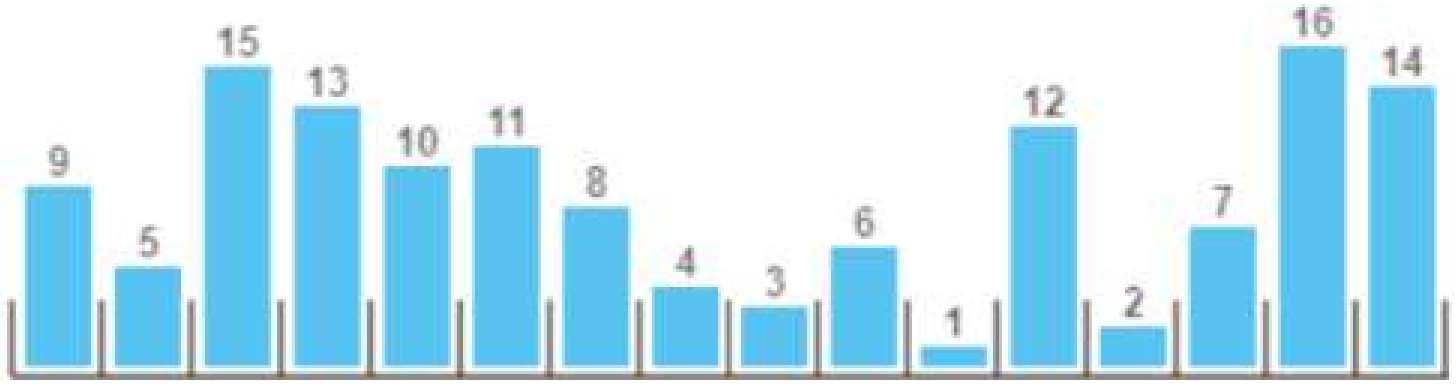
```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

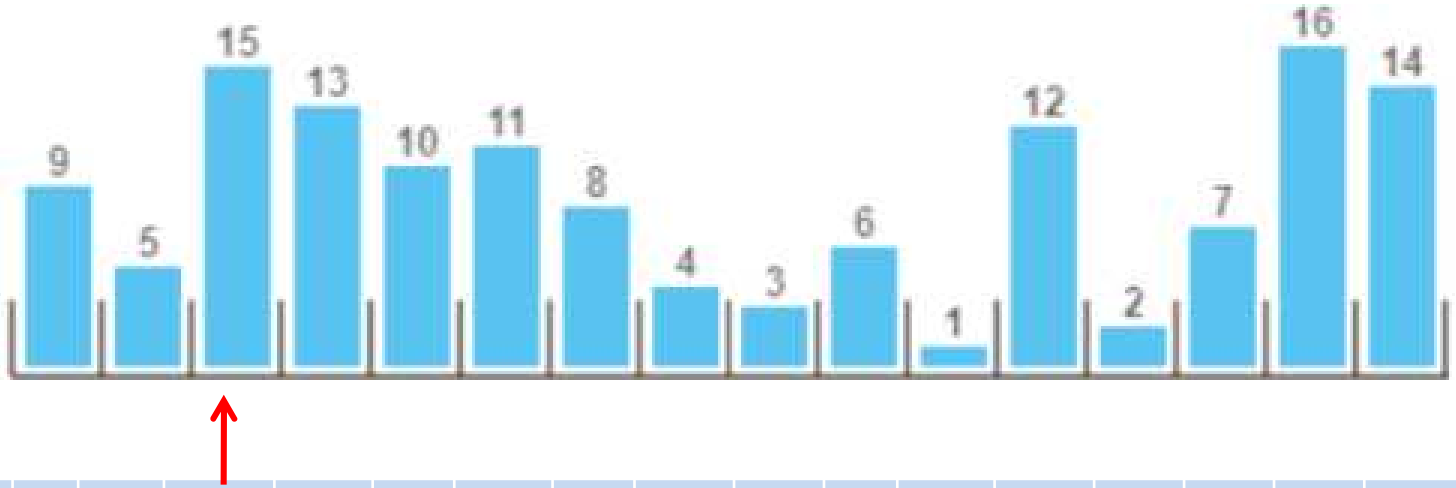


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0															

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

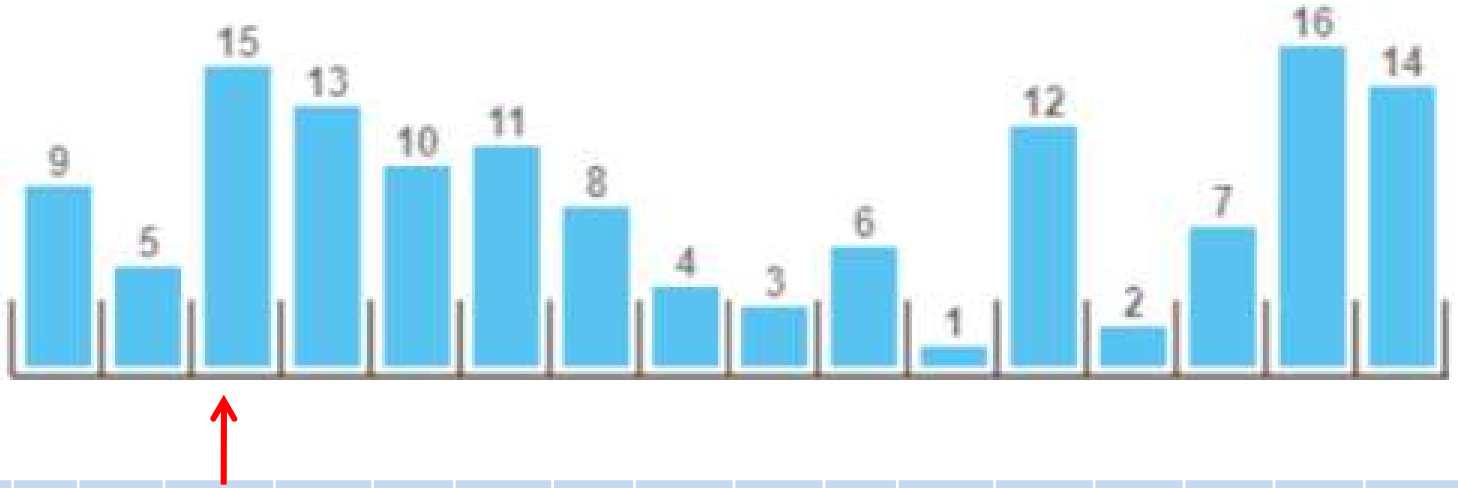


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0															

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

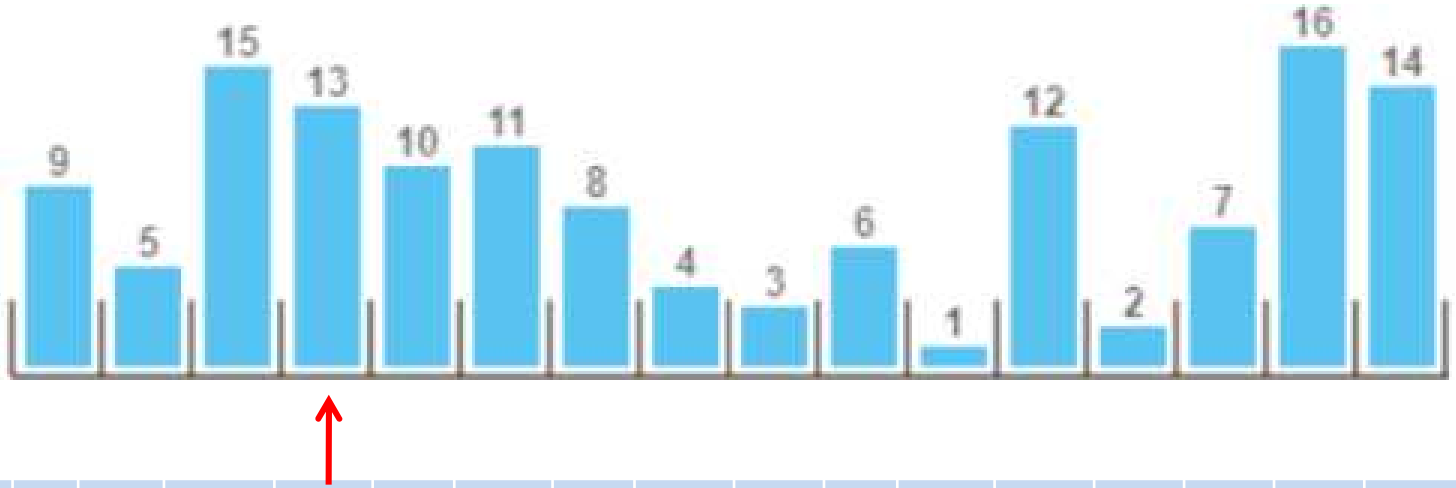


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0		2													

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

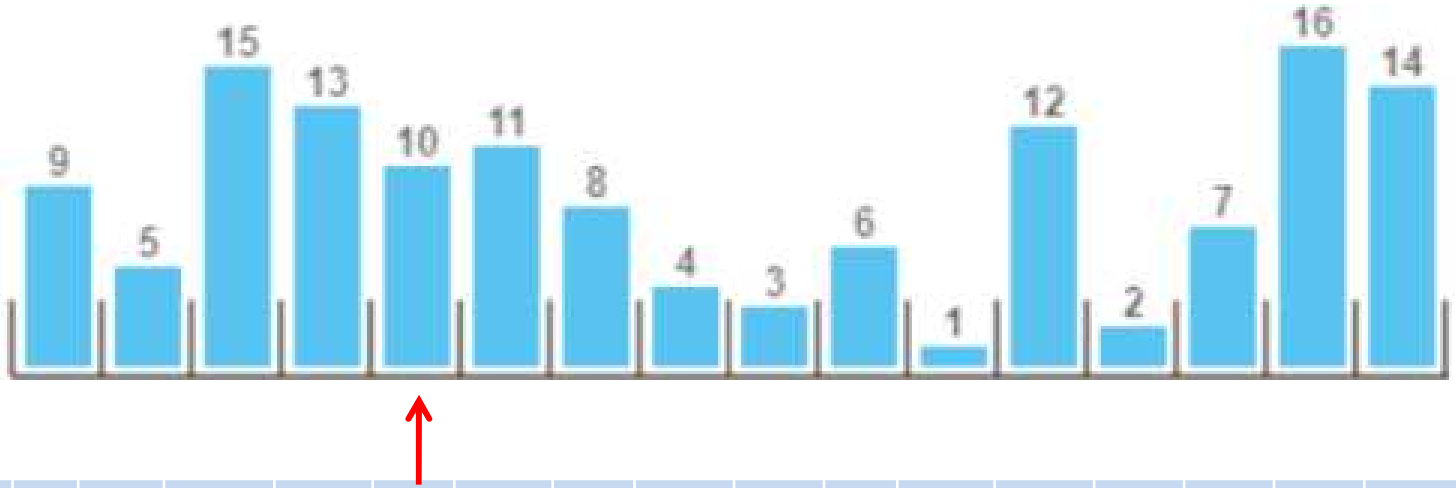


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0		2													

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

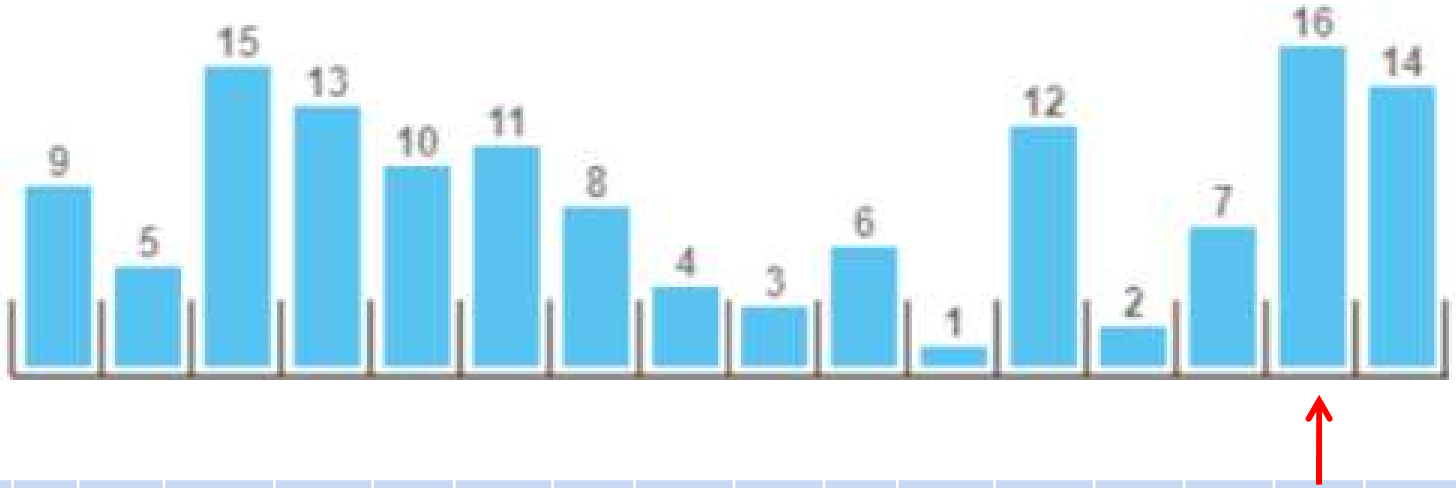


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0		2													

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

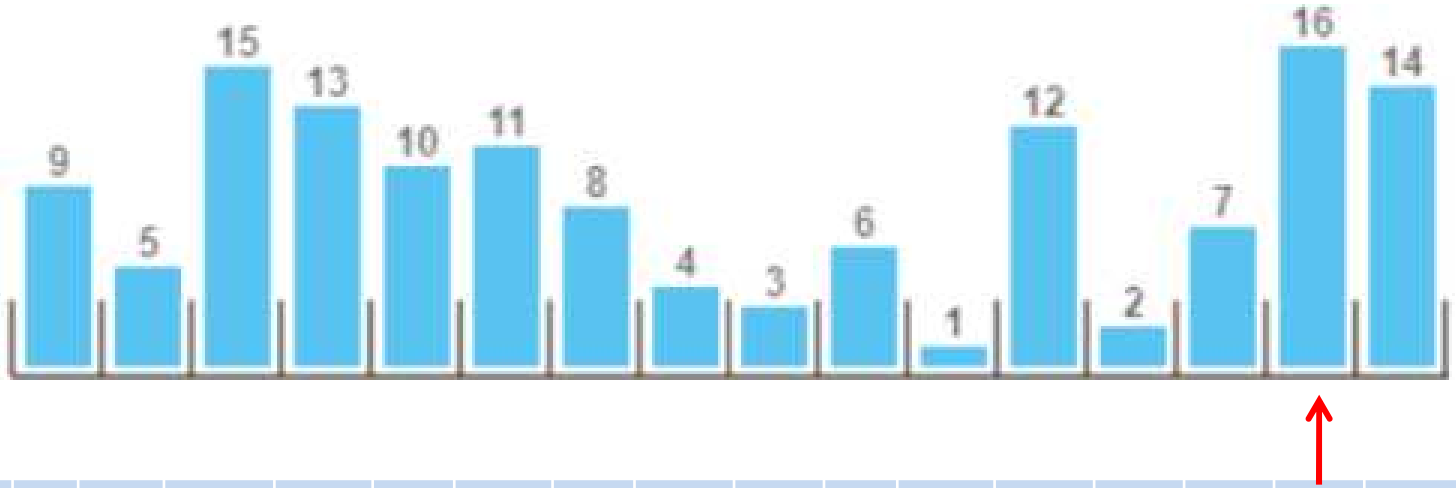


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0		2													

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

```
ind = IndiceMax(t,15)
```

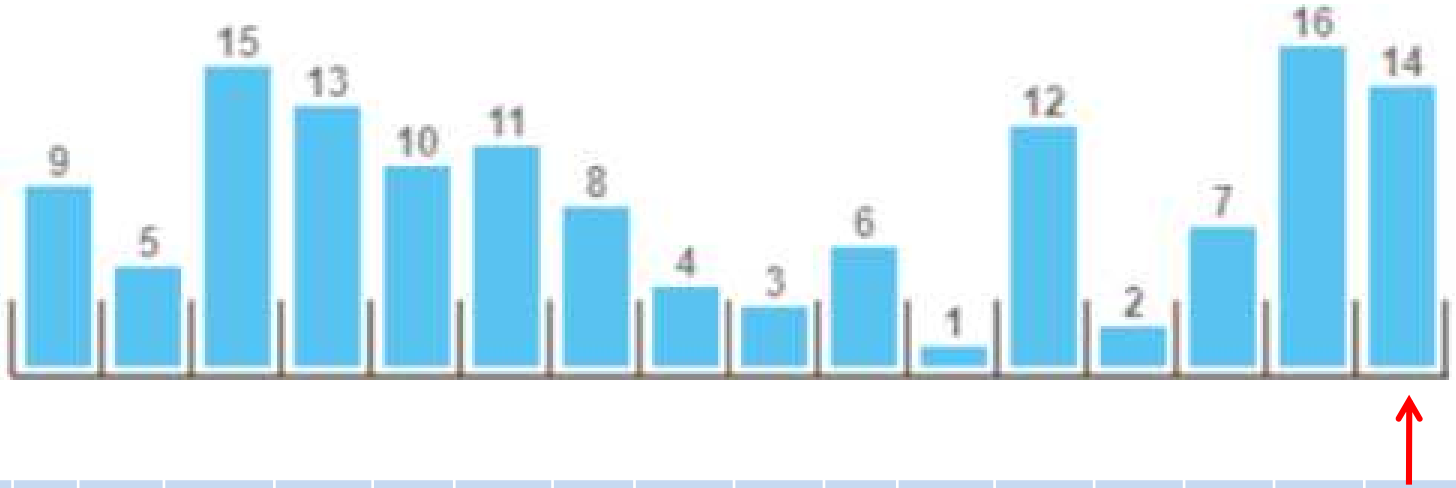


k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0		2												14	

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```

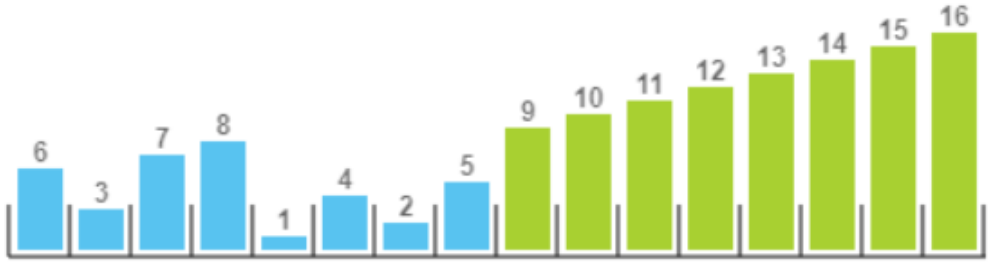
```
ind = IndiceMax(t,15)
```



k		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iMax	0		2												14	

Le tri par sélection

```
def IndiceMax(t,i):
    iMax = 0
    for k in range(1,i+1):
        if t[k]>t[iMax]:
            iMax = k
    return iMax
```



Le tri par sélection

Tri par sélection appliqué à 1 tableau(x) de 1000 élément(s).

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500

Tri par sélection appliqué à 1 tableau(x) de 2000 élément(s).

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Tri par sélection appliqué à 1 tableau(x) de 4000 élément(s).

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

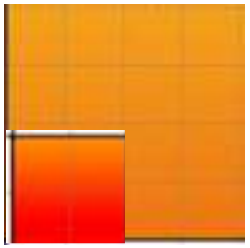
Le tri par sélection

Tri par sélection appliqué à 1 tableau(x) de 1000 élément(s).

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500



Tri par sélection appliqué à 1 tableau(x) de 2000 élément(s).

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Tri par sélection appliqué à 1 tableau(x) de 4000 élément(s).

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

Le tri par sélection

Tri par sélection appliqué à 1 tableau(x) de 1000 élément(s).

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500

Tri par sélection appliqué à 1 tableau(x) de 2000 élément(s).

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Tri par sélection appliqué à 1 tableau(x) de 4000 élément(s).

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

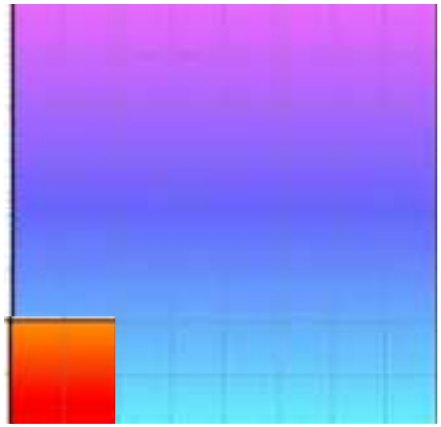
Le tri par sélection

Tri par sélection appliqué à 1 tableau(x) de 1000 élément(s).

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500

Nombre de comparaisons : 499 500



Tri par sélection appliqué à 1 tableau(x) de 2000 élément(s).

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Nombre de comparaisons : 1 999 000

Tri par sélection appliqué à 1 tableau(x) de 4000 élément(s).

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

Nombre de comparaisons : 7 998 000

Questionnaire à Choix Multiples sur le tri par sélection

Pour chaque question, donner l'unique bonne réponse

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

1. Pour localiser la valeur maximale de cette liste, combien faut-il faire de comparaisons ?

   a. 2

   b. 5

   c. 0

   d. 4

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

1. Pour localiser la valeur maximale de cette liste, combien faut-il faire de comparaisons ?

   a. 2

   b. 5

   c. 0

   d. 4

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

2. Quand on applique l'algorithme de tri par sélection à la liste L, à la fin de la première étape on échange de place :

   a. les valeurs 4 et 1

   b. les valeurs 2 et 5

   c. les valeurs 5 et 1

   d. les valeurs 4 et 5

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

2. Quand on applique l'algorithme de tri par sélection à la liste L, à la fin de la première étape on échange de place :

   a. les valeurs 4 et 1

   b. les valeurs 2 et 5

   c. les valeurs 5 et 1

   d. les valeurs 4 et 5

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

3. A la fin de la troisième étape de l'algorithme de tri par sélection, la liste L vaut :

   a. `[4, 5, 2, 3, 1]`

   b. `[2, 1, 3, 4, 5]`

   c. `[1, 2, 3, 4, 5]`

   d. `[5, 4, 3, 2, 1]`

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

3. A la fin de la troisième étape de l'algorithme de tri par sélection, la liste L vaut :

   a. `[4, 5, 2, 3, 1]`

   b. `[2, 1, 3, 4, 5]`

   c. `[1, 2, 3, 4, 5]`

   d. `[5, 4, 3, 2, 1]`

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

4. Pour trier cette liste avec l'algorithme de tri par sélection on effectue au total :

   a. 5 comparaisons

   b. 10 comparaisons

   c. 15 comparaisons

   d. 20 comparaisons

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

4. Pour trier cette liste avec l'algorithme de tri par sélection on effectue au total :

   a. 5 comparaisons

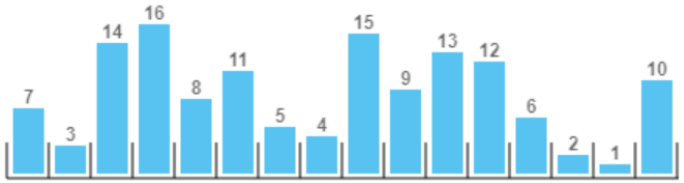
   b. 10 comparaisons

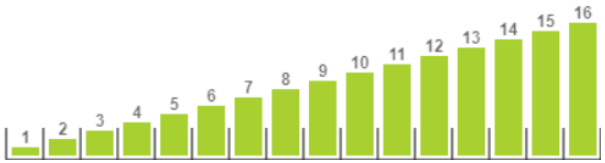
   c. 15 comparaisons

   d. 20 comparaisons

3. Le tri par insertion

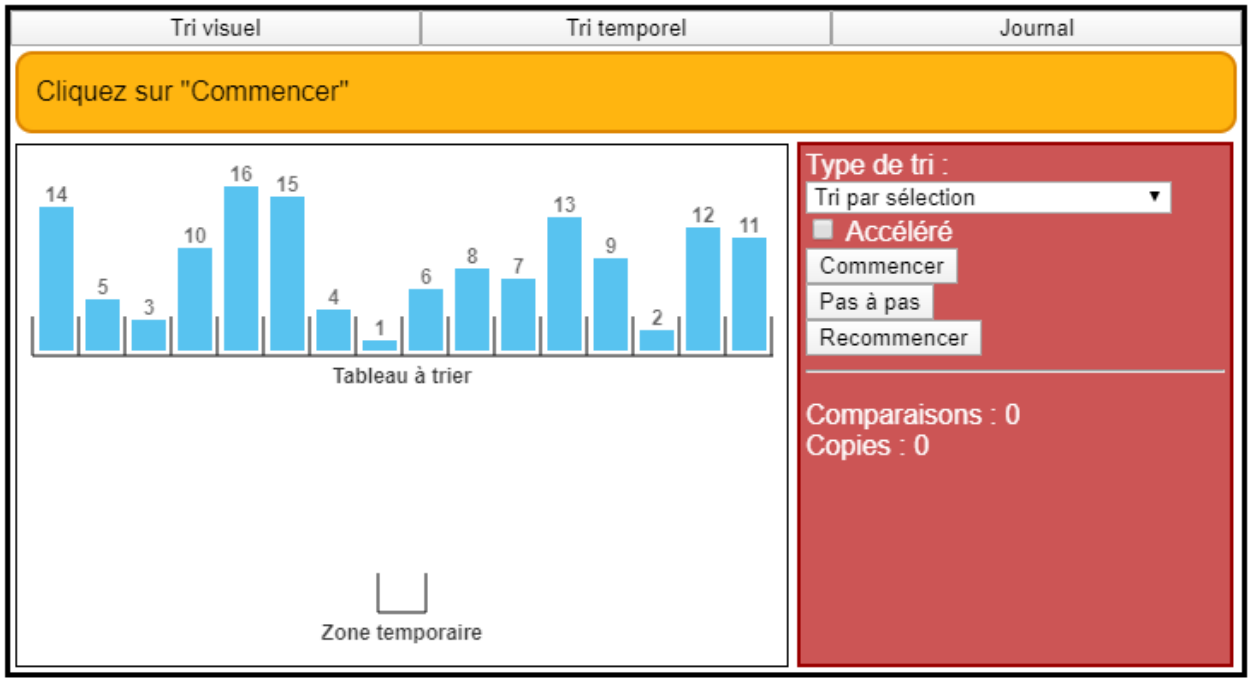
Cette méthode de tri est très différente de la méthode de tri par sélection et s’apparente à celle utilisée pour trier ses cartes dans un jeu.





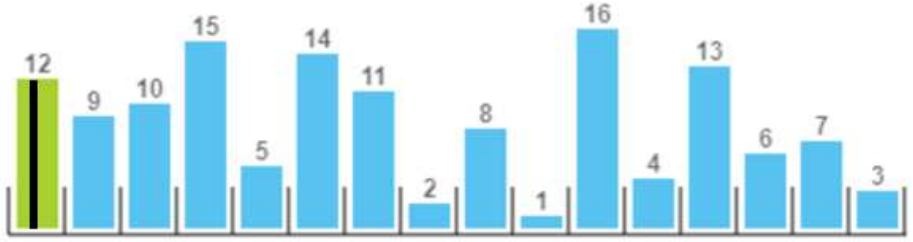
)i(interstices info

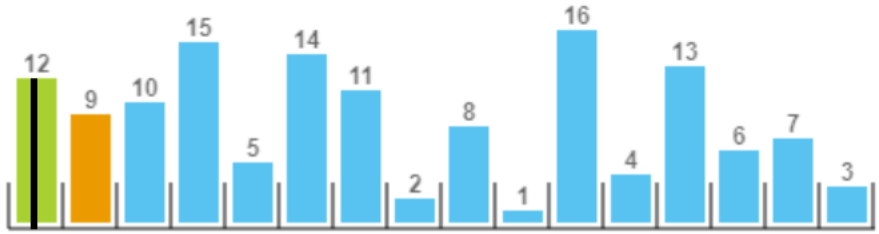
https://interstices.info/les-algorithmes-de-tri/

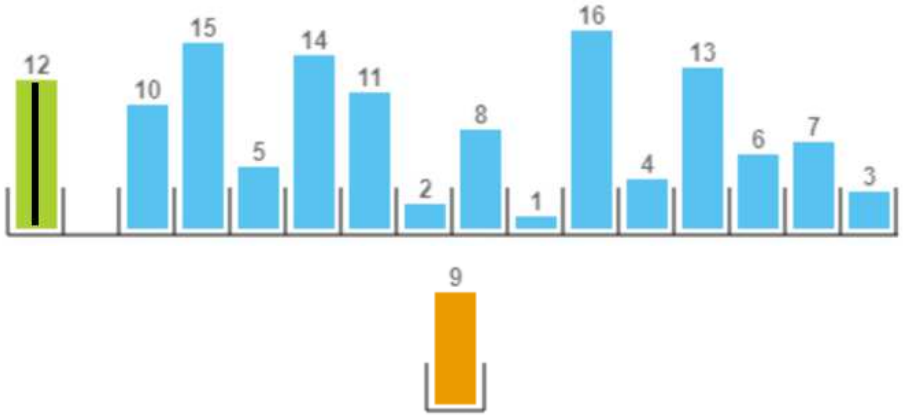


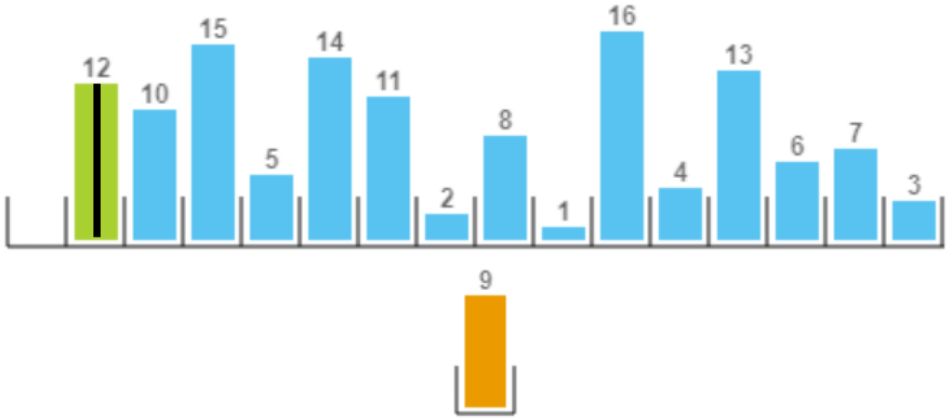
Interstices est une revue de culture scientifique en ligne, créée par des chercheurs pour vous inviter à explorer les sciences du numérique.

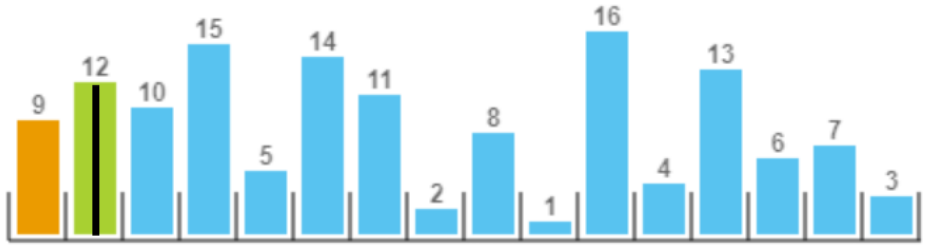
A chaque étape on insère un nouvel élément dans la partie triée

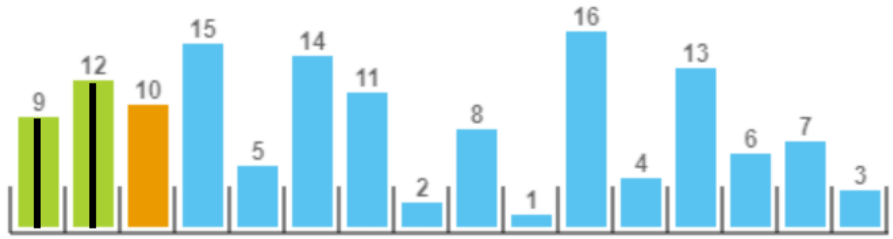


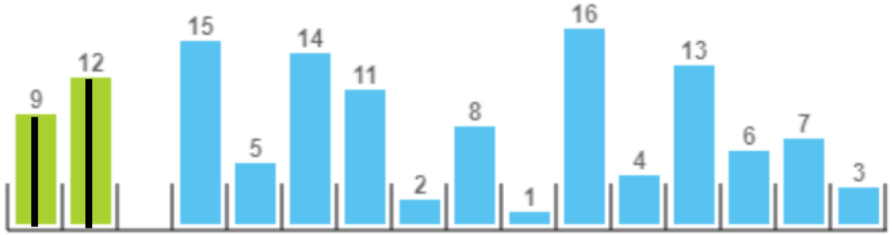


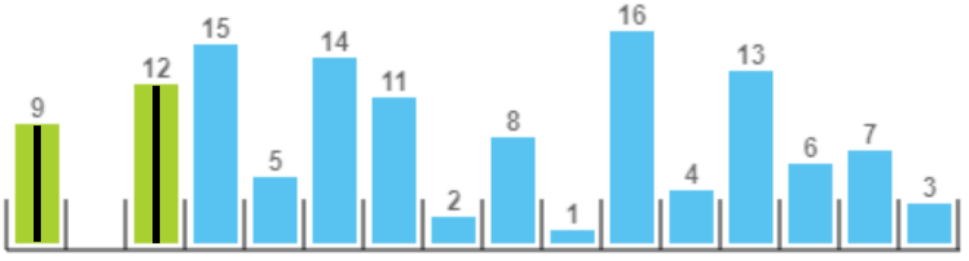






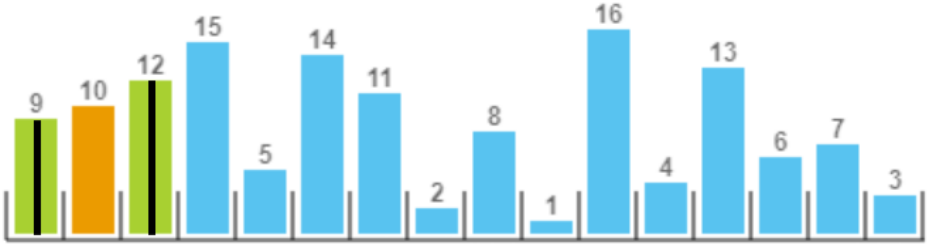


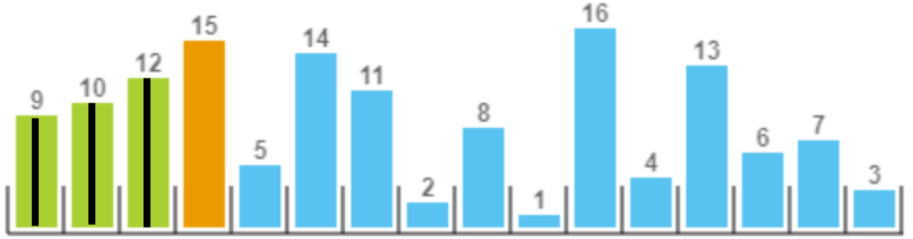

9 12 15 5 14 11 2 8 1 16 4 13 6 7 3


9 12 15 5 14 11 2 8 1 16 4 13 6 7 3

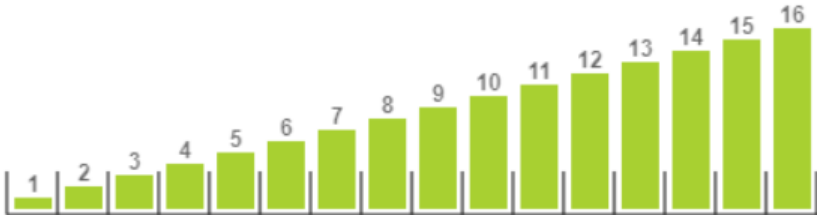

10


10


9 10 12 15 5 14 11 2 8 1 16 4 13 6 7 3


9 10 12 15 5 14 11 2 8 1 16 4 13 6 7 3

.....


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Le tri par insertion

```
def tri_insertion(t):
    for j in range(1,len(t)):
        insere(t, j)
```

Le tri par insertion

```
def tri_insertion(t):
    for j in range(1,len(t)):
        insere(t, j)
```

```
insere(t,1)
```

Le tri par insertion

```
def tri_insertion(t):
    for j in range(1,len(t)):
        insere(t, j)
```

```
insere(t,1)
insere(t,2)
```

Le tri par insertion

```
def tri_insertion(t):
    for j in range(1,len(t)):
        insere(t, j)
```

```
insere(t,1)
insere(t,2)
insere(t,3)
```

Le tri par insertion

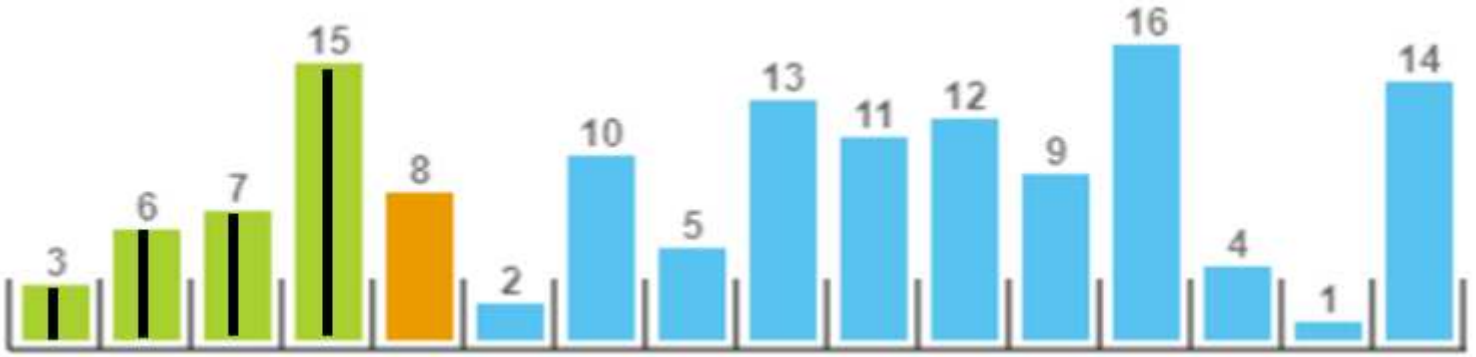
```
def tri_insertion(t):
    for j in range(1,len(t)):
        insere(t, j)
```

```
insere(t,1)
insere(t,2)
insere(t,3)
…
insere(t,15)
```

Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

```
insere(t,4)
```

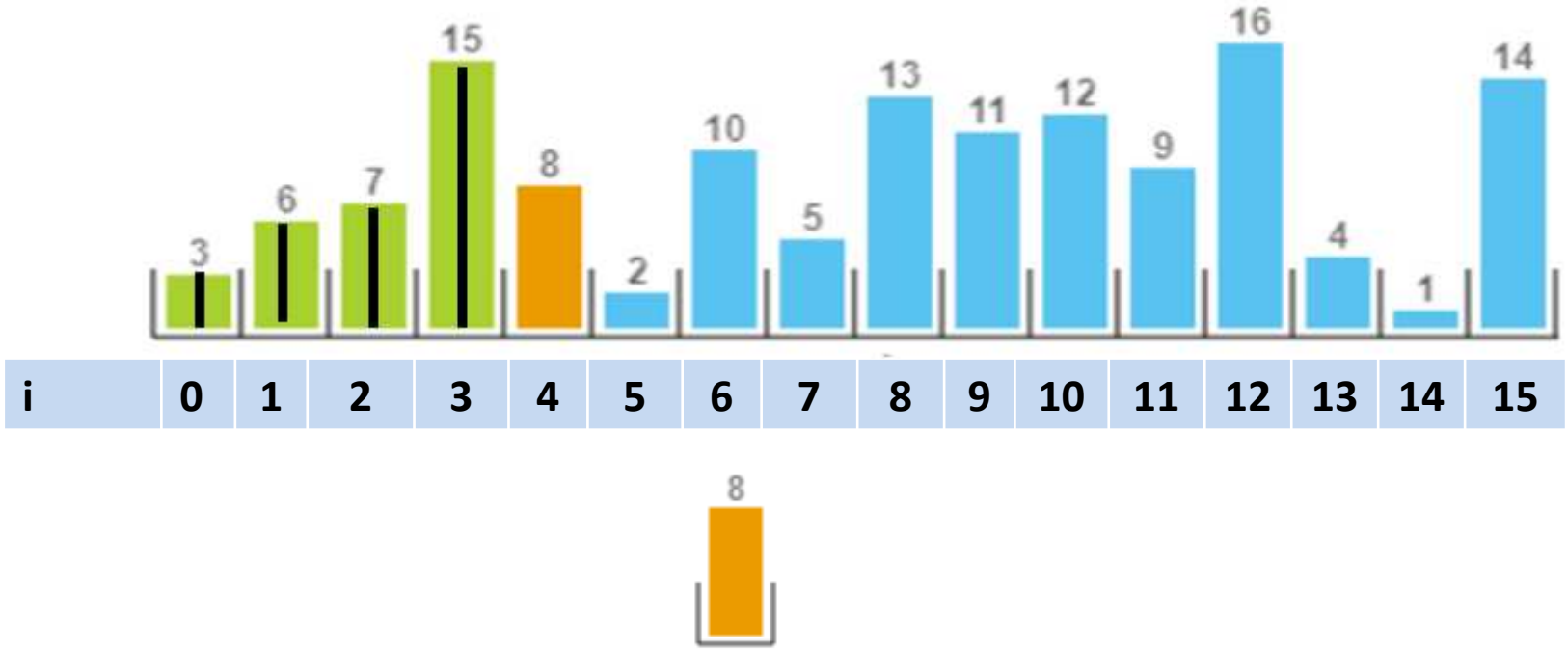


i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

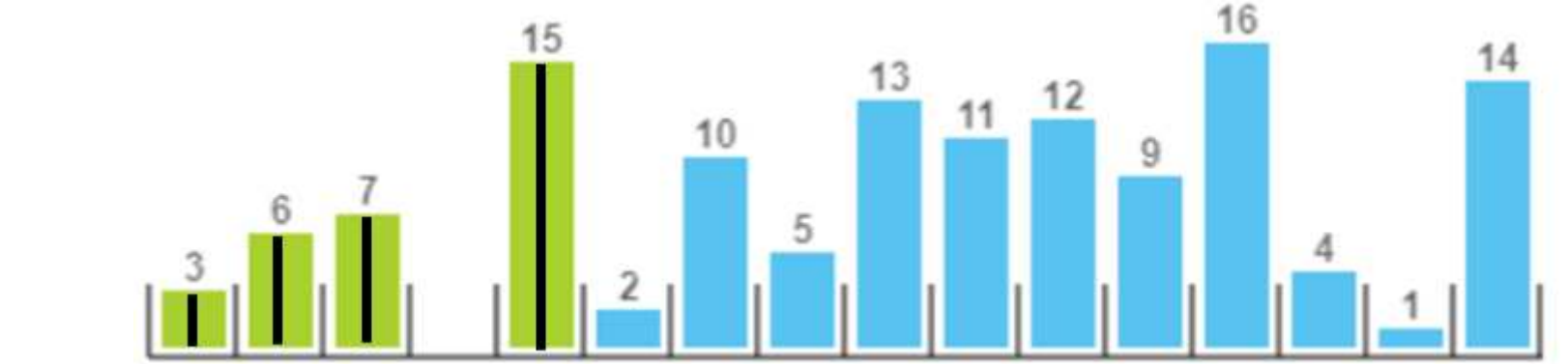
```
insere(t,4)
```



Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

```
insere(t,4)
```

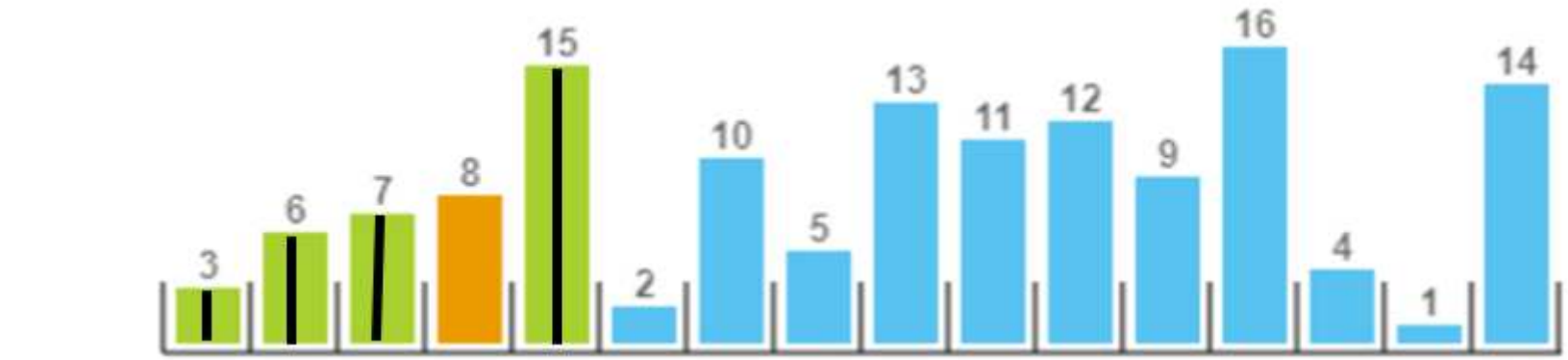


i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

```
insere(t,4)
```

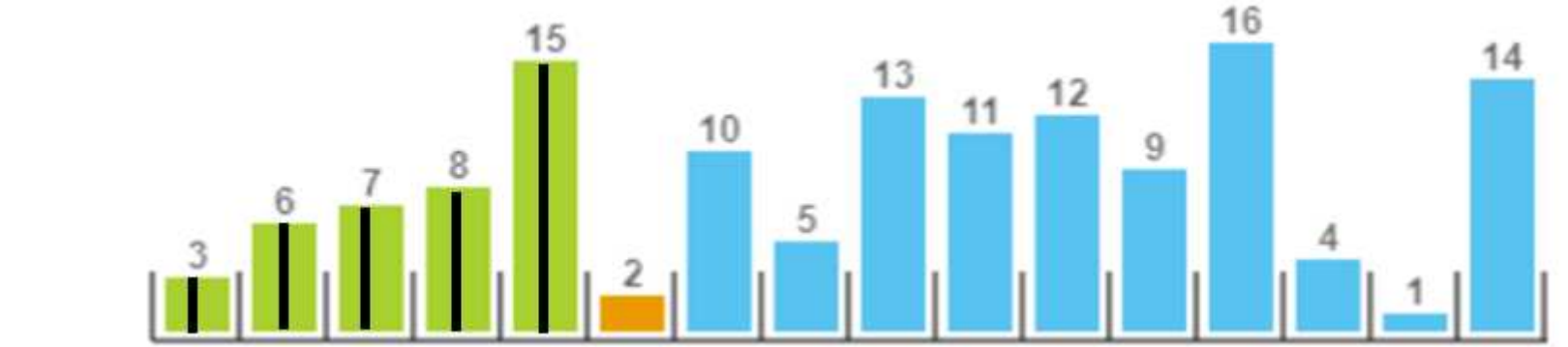


i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

```
insere(t,5)
```

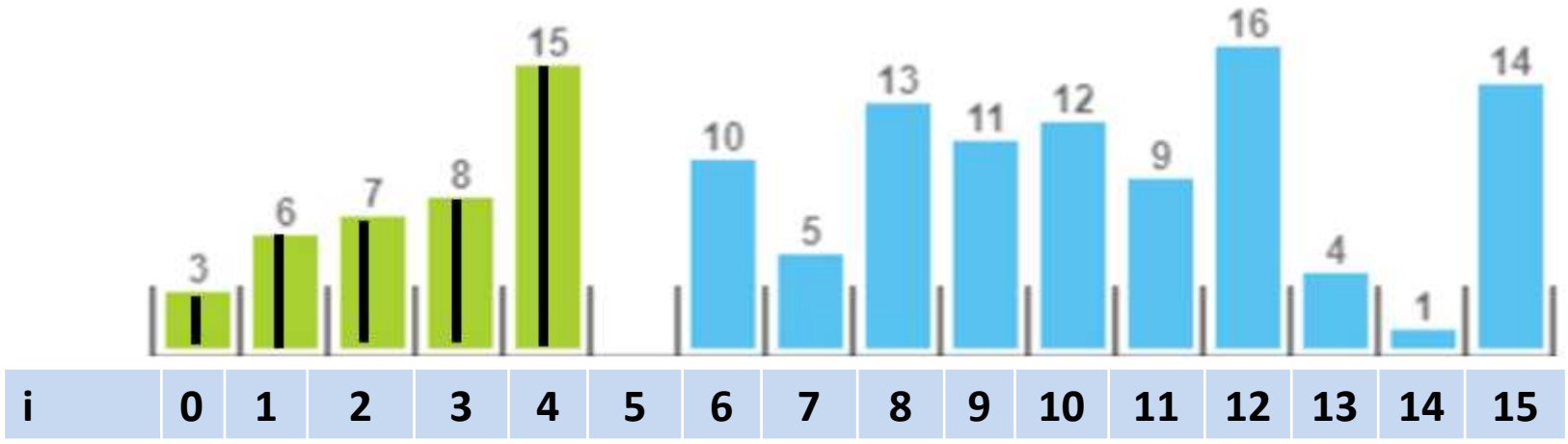


i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

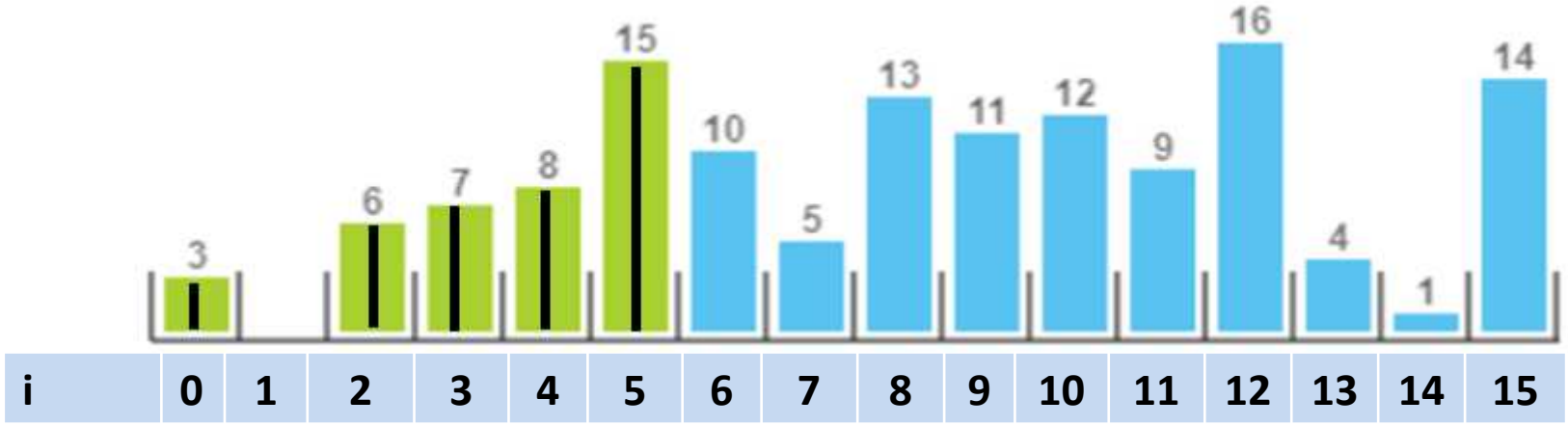
insere(t,5)



Le tri par insertion

```
def insere(t,j):
    val = t[j]
    i = j-1
    while i>=0 and t[i]>val:
        t[i+1] = t[i]
        i = i-1
    t[i+1] = val
```

```
insere(t,5)
```



Tri par insertion appliqué à 1 tableau(x) de 1000 élément(s).

Nombre de comparaisons : 255 771

Nombre de comparaisons : 250 798

Nombre de comparaisons : 242 425

Ordre de grandeur 250 000

Tri par insertion appliqué à 1 tableau(x) de 2000 élément(s).

Nombre de comparaisons : 995 770

Nombre de comparaisons : 976 313

Nombre de comparaisons : 989 262

Ordre de grandeur 1 000 000

Tri par insertion appliqué à 1 tableau(x) de 4000 élément(s).

Nombre de comparaisons : 4 084 713

Nombre de comparaisons : 4 078 119

Nombre de comparaisons : 4 051 259

Ordre de grandeur 4 000 000

Dans certains cas comme dans un tableau trié il y a beaucoup moins de comparaisons.

Questionnaire à Choix Multiples sur le tri par insertion

Pour chaque question, donner l'unique bonne réponse

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

1. Si on applique l'algorithme de tri par insertion à la liste L, alors à la première étape on échange :

   a. les valeurs `4` et `5`

   b. les valeurs `3` et `1`

   c. aucune valeur

   d. les valeurs `2` et `3`

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

1. Si on applique l'algorithme de tri par insertion à la liste L, alors à la première étape on échange :

   a. les valeurs 4 et 5

   b. les valeurs 3 et 1

   c. aucune valeur

   d. les valeurs 2 et 3

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

2. A la fin de la deuxième étape de l'algorithme de tri par insertion la liste L vaut :

   a. `[2, 4, 5, 3, 1]`

   b. `[4, 5, 1, 2, 3]`

   c. `[5, 4, 3, 2, 1]`

   d. `[1, 5, 4, 3, 2]`

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

2. A la fin de la deuxième étape de l'algorithme de tri par insertion la liste L vaut :

   a. **`[2, 4, 5, 3, 1]`**

   b. `[4, 5, 1, 2, 3]`

   c. `[5, 4, 3, 2, 1]`

   d. `[1, 5, 4, 3, 2]`

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

3. Avec l'algorithme de tri par insertion, la liste est triée en :

   a. 2 étapes

   b. 3 étapes

   c. 4 étapes

   d. 5 étapes

On donne la liste suivante

```
L = [4, 5, 2, 3, 1]
```

3. Avec l'algorithme de tri par insertion, la liste est triée en :

   a. 2 étapes

   b. 3 étapes

   c. 4 étapes

   d. 5 étapes


Maintenant je comprends !