Partie 4 : Tri itératif de tableau

Hypothèses de travail :

- On manipule des **entiers**.

On choisit des entiers par simplicité mais on pourrait trier des tableaux contenant un autre type de données tant que ces données sont ordonnables (c'est-à-dire qu'on peut les classer).

- On cherche à ordonner une séquences de **N** entiers.

- On considère que ces entiers sont déjà dans un tableau.

On va donc modifier le tableau initial pour qu'il soit trié à la fin. Dans la mesure du possible on va tenter de ne pas utiliser de second tableau pour éviter d'encombrer la mémoire.

- On considère que l'ordre recherché est l'ordre croissant.

On peut facilement modifier les algorithmes pour obtenir l'ordre décroissant donc ce choix n'est pas limitatif.

Tri itératif de tableau

Critères d'analyses des algorithmes :

Chaque algorithme sera étudié selon les 3 critères suivant pour nous permettre de les comparer :

- Complexité en temps (voir cours précédent)
- Complexité en mémoire (voir diapo précédente)
- Stabilité (voir diapo suivante)

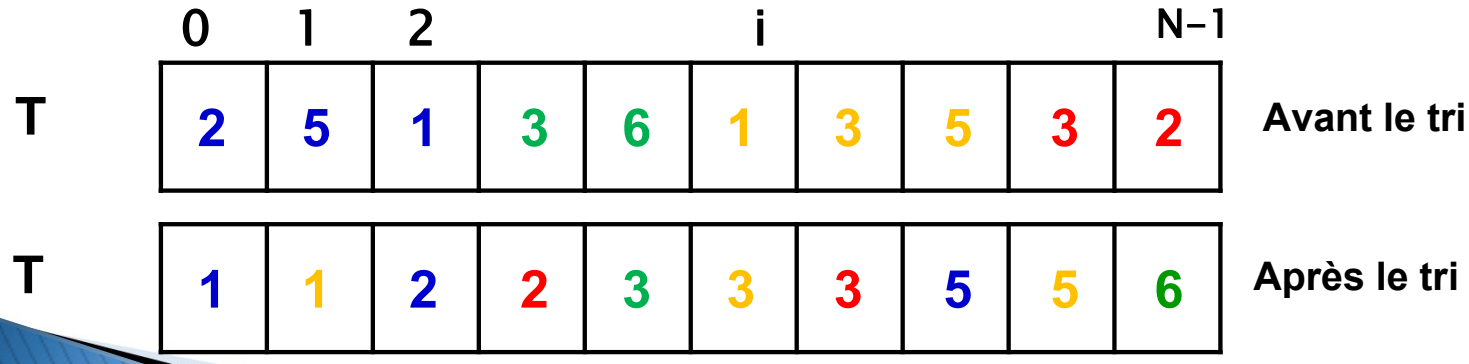
Tri itératif de tableau

Critères d'analyses des algorithmes : **Stabilité** (1/2)

Un algorithme de tri de tableau sera dit stable si, quand on l'applique, il garantit la conservation d'un tri déjà existant dans le tableau.

Si on considère par exemple que le tableau T ci-dessous est trié selon l'ordre des couleurs : d'abord le bleu puis le vert puis le jaune puis le rouge. **Un algorithme de tri stable triera les entiers par ordre croissant, mais si il existe plusieurs occurrences d'un même entier dans T l'ordre initial des couleurs sera conservé.** Par exemple ici le après le tri on a

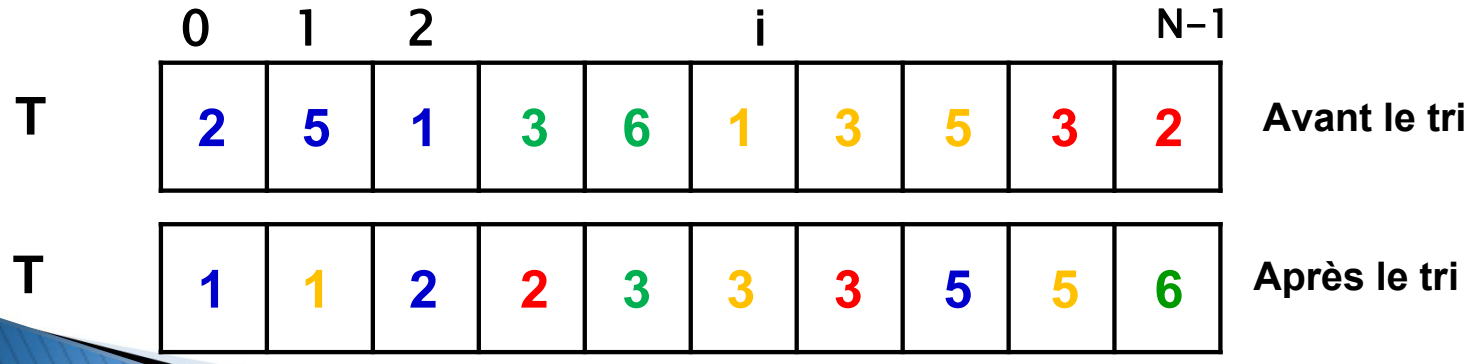
3 3 3



Tri itératif de tableau

Critères d'analyses des algorithmes : **Stabilité** (2/2)

- Tous les algorithmes que nous allons étudier ne seront pas stables. Vous devez comprendre la définition et comprendre pourquoi un algorithme est stable ou pas. On y réfléchira dans la suite.
- Dans le futurs si vous avez besoin de cette propriété pour trier un tableau il faudra savoir quel algorithme choisir.
- De façon évident cette propriété n'est pas utile si vos données sont toutes différentes dans le tableau que vous voulez trier puisqu'elle s'applique aux éléments de valeurs identiques.

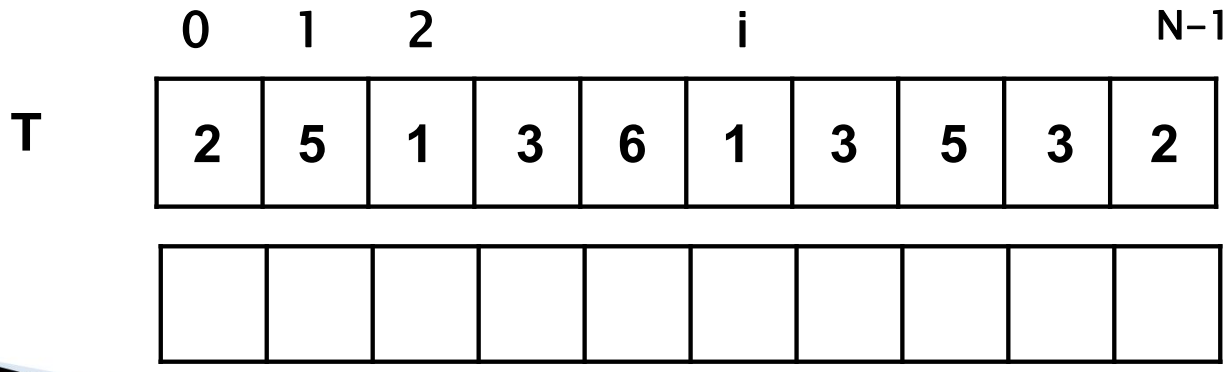


Tri itératif de tableau

Ici l'objectif est que **vous réfléchissiez** quelques minutes à des idées pour trier un tableau T en ordre croissant.

Pour cela vous pouvez imaginer que vous avez devant vous une boite avec des boules numérotées et que vous voulez les trier en ordre croissant.

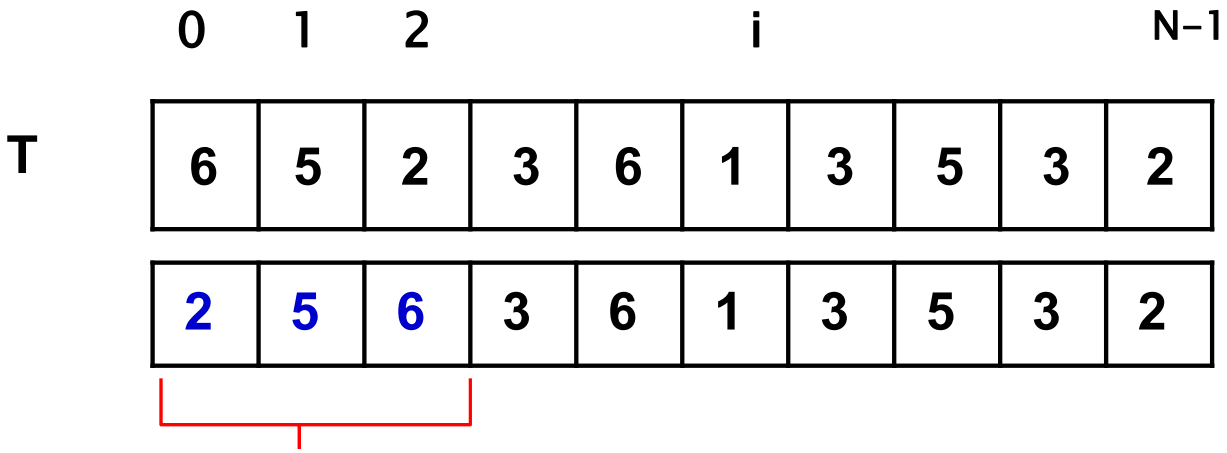
Quelles idées proposez-vous ? Prenez le temps d'essayer.



Tri itératif de tableau

Idées : Ici l'objectif est le même que celui de la diapo précédente mais en supposant qu'un algorithme de tri à déjà commencé son travail et que les premiers éléments du tableau sont déjà triés entre eux.

Quelles idées proposez-vous pour continuer ? Peut-être n'avez-vous pas imaginé une méthode de ce type sur le problème précédent.



Les trois premiers éléments sont déjà triés entre eux comment continue-t-on ?

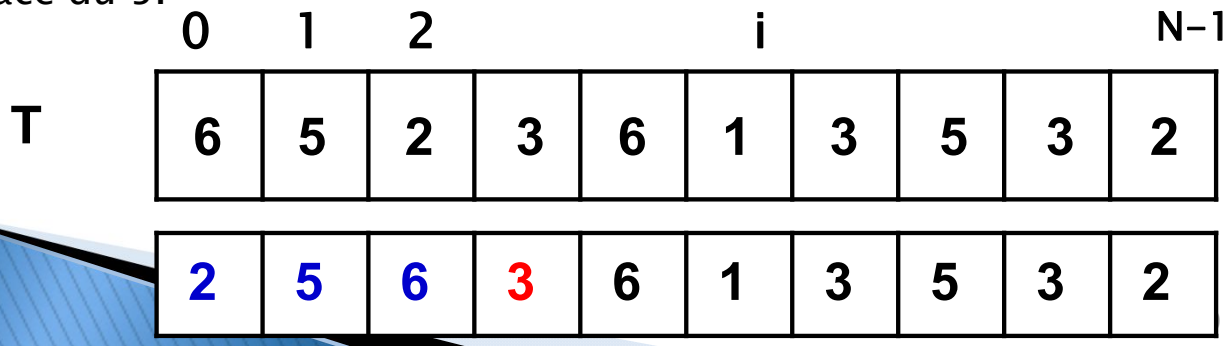
Tri itératif de tableau

```
def diapoSuivante(diapo,assezReflechi):
  if assezReflechi :
    return (diapo+1)
  else :
    print(`` bien essayé'')
    return(diapo-2)
```

Tri itératif de tableau

Les idées qui sont en général discutées en amphi à ce moment là (après avoir réfléchi sur les 2 diapos précédentes) sont les suivantes :

1. On compare les éléments 2 à 2 et si ils ne sont pas dans l'ordre choisi on les échange et on recommence…
2. On cherche le minimum quand on le trouve on l'échange avec l'élément dans la première case et on recommence avec le second minimum.
3. Si on considère le contexte de la diapo précédente on prend le premier élément qui n'est pas dans la zone des éléments triés entre eux et on le met à sa place par échanges successifs vers la gauche. C'est le tri du joueur de carte. Dans la vie courante on le pratique souvent mais la plupart du temps l'idée de vient pas quand on réfléchit aux algorithmes de tri sur un tableau. Ici il s'agira de placer le 3 à la place du 6 puis à la place du 5.



176

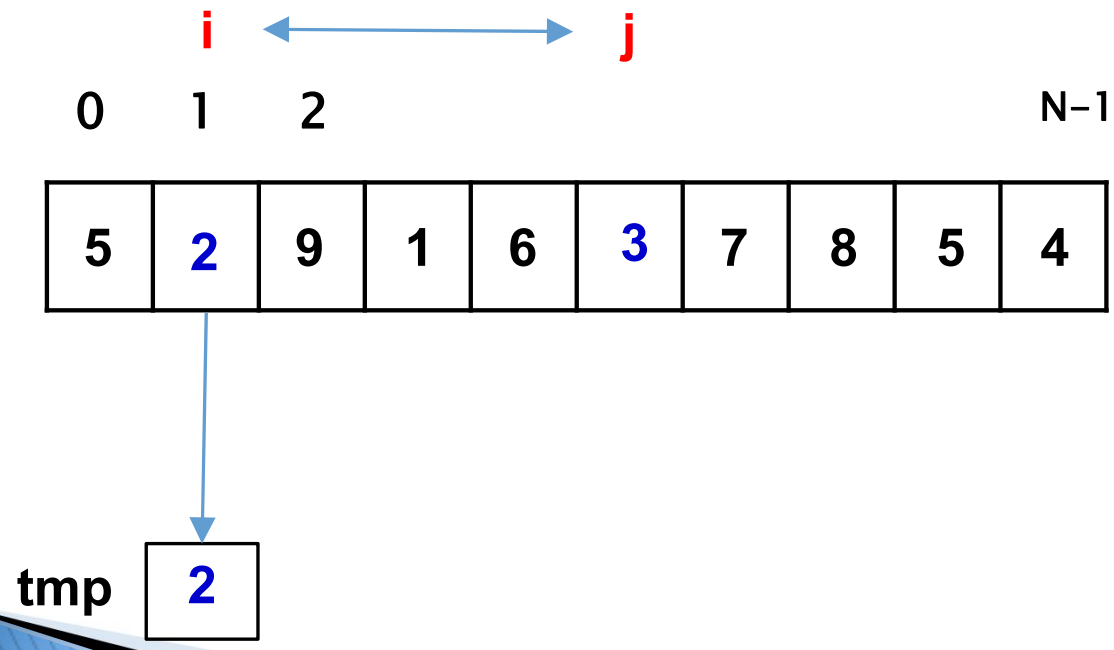
Tri par sélection

- Le tri par sélection est celui dont l'idée principale est décrite au point 2 de la page précédente. Nous allons maintenant le détailler et écrire l'algorithme.
- Pour écrire l'algorithme nous allons avoir besoin de 2 autres fonctions :
  - La fonction **`echange`** qui permet d'échanger la place de 2 éléments dans un tableau.
  - La fonction **`indiceMin`** qui permet de trouver l'indice de l'élément le plus petit entre les cases i et N-1 du tableau.

Tri par sélection : fonction échange

Pour la suite nous allons avoir besoin de la fonction :

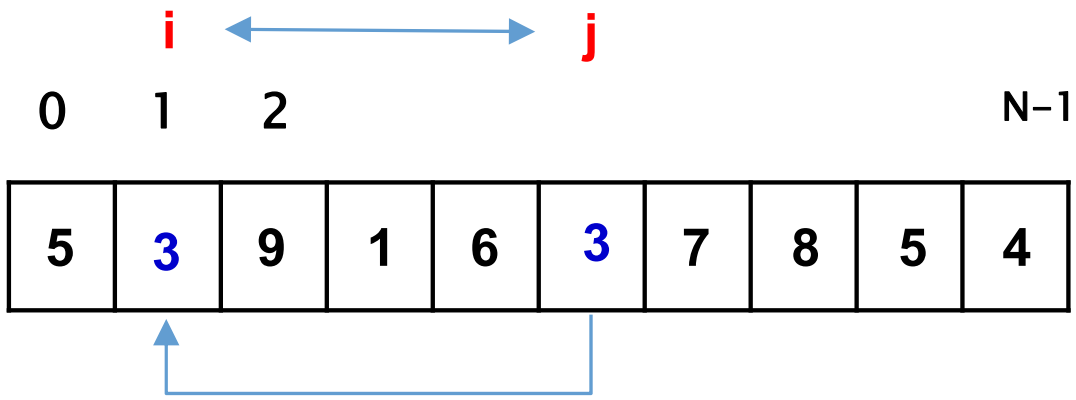
```
echanger (t, i, j)
```



Tri par sélection

Pour la suite nous allons avoir besoin de la fonction :

```
echanger (t, i, j)
```

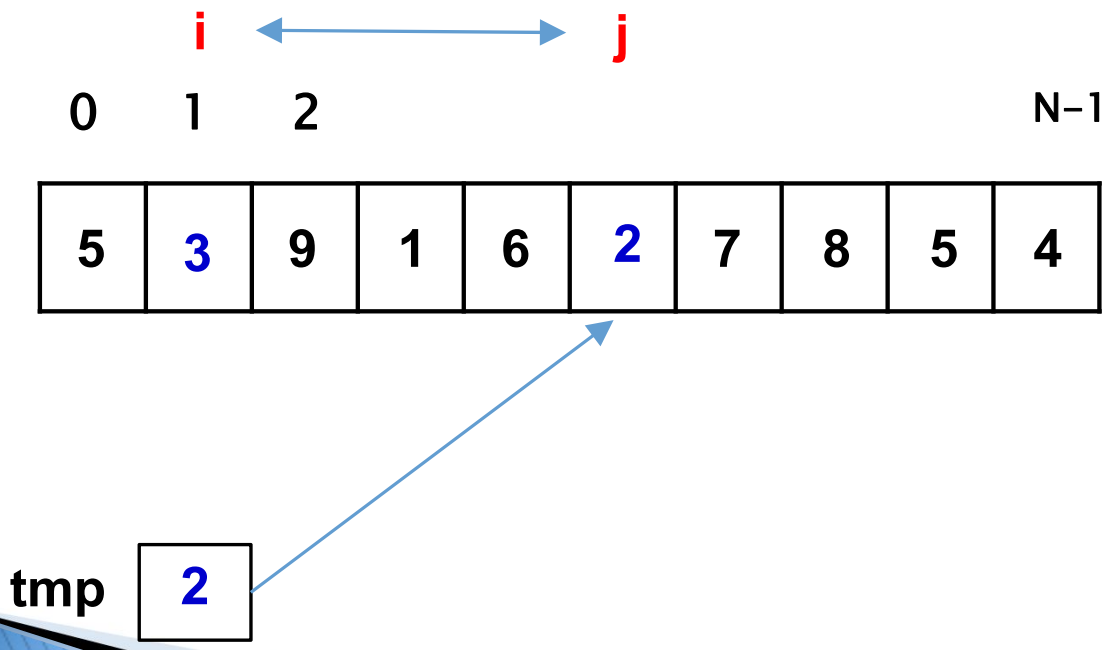


tmp | 2

Tri par sélection

Pour la suite nous allons avoir besoin de la fonction :

```
echanger (t, i, j)
```



Tri par sélection

Utilise la fonction

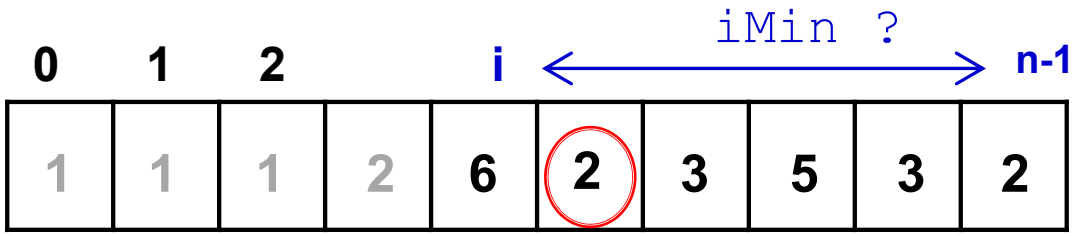
```
def echanger (t, i, j):
    tmp=t[i]
    t[i]=t[j]
    t[j]=tmp
```

Temps de calcul :
$\Theta(1)$

Tri par sélection : fonction `indiceMin`

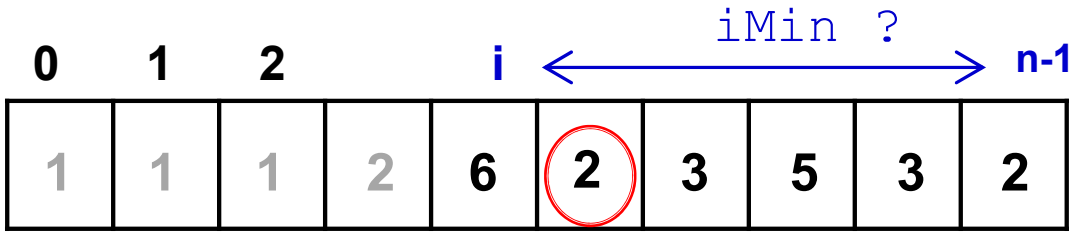
Cette fonction est basée sur la fonction qui permet de trouver l'indice de l'élément minimum dans un tableau.

- On doit adapter l'algorithme pour l'utiliser sur un sous tableau (entre les indices i et n-1).
- Elle sera utilisée en supposant que le début du tableau (entre les indices 0 et i-1) est déjà trié.



Tri par sélection : fonction `indiceMin`

Le code de cette fonction est :




```
def indiceMin(t, n, i):
    iMin=i
    for j in range (i+1, n):
      if(t[j]<t[iMin]):
        iMin=j
    return(iMin)
```

Temps de calcul ?

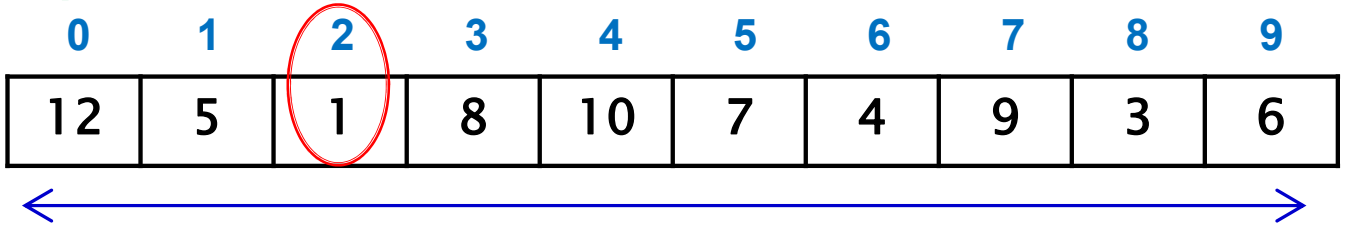
$\Theta(n-i)$

Car on doit parcourir tous les éléments entre les indices i et n-1 pour trouver le minimum et son indice (les constantes ne sont pas retenues)

Tri par sélection : déroulement

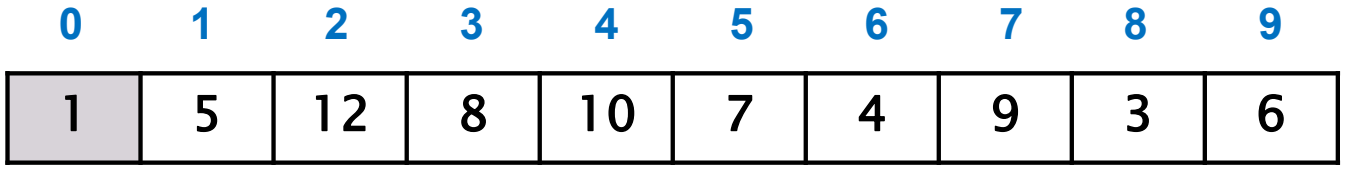
3 : Eléments à échanger

2: Minimum trouvé après exploration



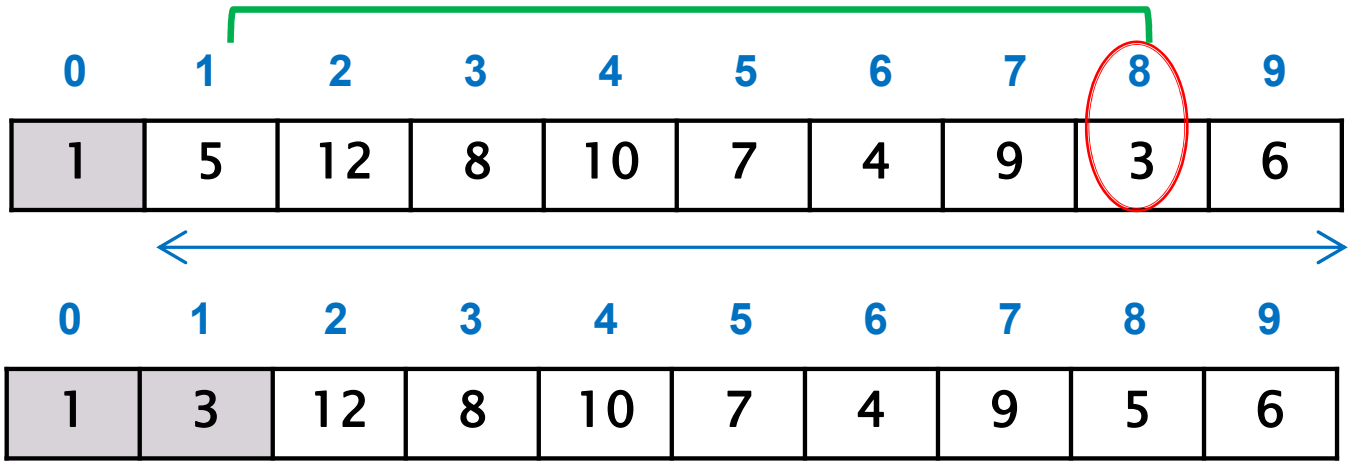
1 : Zone explorée

On obtient :



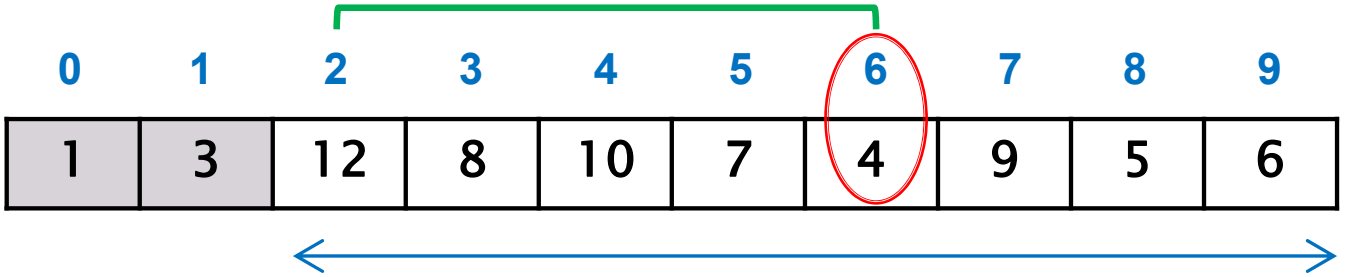
Tri par sélection : déroulement

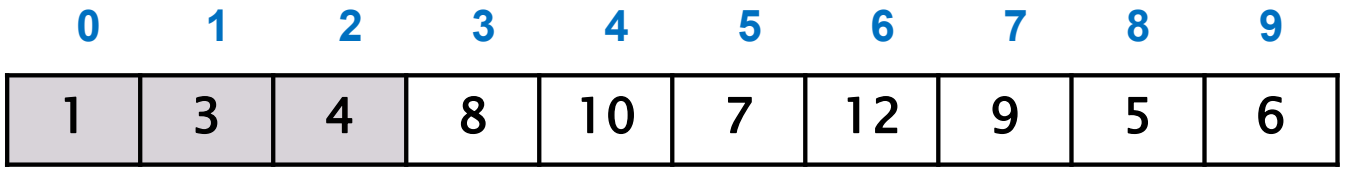
Pour la seconde étape, en utilisant le même code couleur et le même ordre des étapes que sur la diapo précédente on obtient :



Tri par sélection : déroulement

3ème étape





On continue selon ce principe jusqu'à ce que la zone à traiter soit réduite à une case

Tri par sélection : algorithme

```
def triSelection (t,n):
  for  i in range(n) :
    iMin=i
    for j in range(i+1,n):
      if(t[j]<t[iMin]):
        iMin=j
    if(i != iMin):
      echanger(t,i,iMin)
```

En rouge on reconnait le code de la fonction **indiceMin**

Comme pour tous les algorithmes de tri suivants on se pose les questions suivantes :

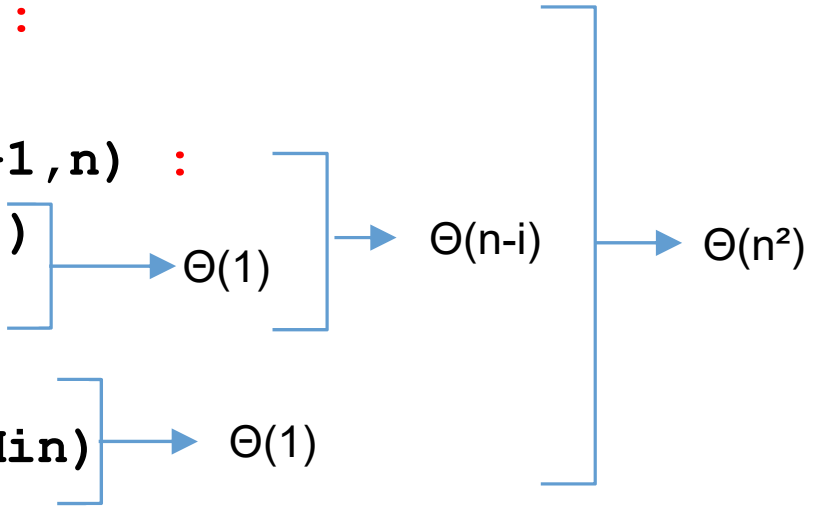
Complexité en temps ?

Complexité en mémoire ?

Stabilité ?

Tri par sélection : analyse

```
def triSelection (t,n):
  for  i in range(n) :
    iMin=i
    for j in range(i+1,n) :
      if(t[j]<t[iMin])
        iMin=j
    if(i != iMin):
      echanger(t,i,iMin)
```



Complexité en temps : Θ(n²) car à chaque étape la zone à explorer diminue d'une case. On fait donc successivement n-1 puis n-2 puis n-3 puis…1 comparaisons. Au total on fait n*(n-1)/2 comparaisons (somme des n-1 premiers entiers) donc la complexité est de l'ordre de n² (les constantes ne sont pas retenues). Si vous n est grand cela prendra beaucoup de temps (chapitre précédent)

Tri par sélection : analyse

```
def triSelection (t,n):
  for  i in range(n) :
    iMin=i
    for j in range(i+1,n) :
      if(t[j]<t[iMin])
        iMin=j
    if(i != iMin):
      echanger(t,i,iMin)
```

Complexité en mémoire : Θ(1) nous trions directement dans le tableau t sans avoir besoin d'un second tableau

Tri par sélection : analyse

```
def triSelection (t,n):
  for  i in range(n) :
    iMin=i
    for j in range(i+1,n) :
      if(t[j]<t[iMin])
        iMin=j
    if(i != iMin):
      echanger(t,i,iMin)
```

Stabilité : A votre avis ? J'attends vos réponses dans le forum du moodle ou dans le live j'arrive à l'organiser. Suivez vos mails

Tri par sélection

Vous trouverez beaucoup d'illustrations du déroulement de ce tri sur youtube par exemple.

Celui-ci est surement le plus étrange… :
https://www.youtube.com/watch?v=Ns4TPTC8whw

Posez toutes vos questions concernant ce cours ou les TD dans le forum sur moodle.