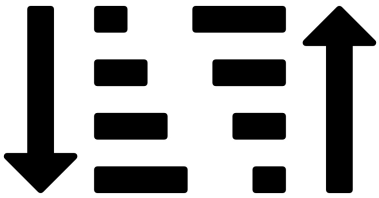
TRIER

Le tri est une activité fondamentale et omniprésente de l'informatique.



- On l'utilise régulièrement, par exemple :
  - Lorsque l'on consulte sur Internet des listes de produits que l'on souhaite afficher par prix croissant ou décroissant, par popularité...
  - Lorsque dans un tableur on souhaite trier des noms par ordre alphabétique
  - Lorsque l'on affiche ses photos par date
- Cela permet d'étudier des concepts algorithmiques puissants et efficaces
- **Pour simplifier, on cherche ici à trier des liste d'entiers dans l'ordre croissant. Le but d'un algorithme de tri est ainsi de calculer une nouvelle liste, ou de modifier la liste initiale, de manière à ce qu'elle contienne les mêmes nombres que la liste de départ, mais que ces éléments soient ordonnés.**

In [1]:

```
#Générer une liste d'entiers aléatoires
from random import *
L=[randint(-32,32) for i in range(5)]
print(L)
```

```
[12, -16, -21, -10, -9]
```

1. De quoi dispose un ordinateur pour trier ?

- Une fonction de comparaison ($<$, $>$), pour comparer deux valeurs.
- Des zones de stockage pour mémoriser des emplacements(à l'aide de l'index des éléments d'une liste), déplacer des valeurs...
- Il existe des dizaines d'algorithmes de tri, nous allons en étudier 2 cette année :
  - **Le tri par sélection.**
  - **Le tri par insertion.**

2. Le tri par sélection

8	5	2	6	9	3	1	4	0	7

Présentation

- On commence par chercher, parmi les nombres à trier, un élément plus petit que tous les autres. Cet élément sera le premier de la liste triée.
- On cherche ensuite, parmi ceux qui restent, un élément plus petit que tous les autres, qui sera le deuxième du tableau trié
- On recommence pour trouver le troisième élément trié et ainsi de suite jusqu'à ce que toute la liste soit triée.

Exemple :

On cherche à trier la liste suivante : `[29, -6, 12, -11, 10]`

- Le plus petit élement est -11, on le place au premier index 0
- Que fait-on du 29 ? Il prend la place de -11, la liste devient donc : `[-11,-6,12,29,10]`
- Le plus petit des élements restants à trier est -6, il est déjà bien placé à l'index 1.
- Le plus petit des élements restants à trier est 10, on le place en troisième position (index 2).
- Que fait-on du 12 ? Il prend la place de 10, la liste devient donc : `[-11,-6,10,29,12]`
- Le plus petit des élements restants à trier est 12, on le place en quatrième position (index 3).
- Que fait-on du 29 ? Il prend la place de 12, la liste devient donc : `[-11,-6,10,12,29]`
- Le dernier élément est nécéssairement le plus grand, la liste est triée.

Exercice 1 :

1. En indiquant les listes intermédiaires, trier la liste `[5, -27, -14, 10, 7]` par sélection

    -
    -
    -
    -

1. Même exercice avec la liste `[30, 24, -7, -19, 26, 21, -18]`

    -
    -
    -
    -
    -

Complexité

Exercice 2 :

On cherche à trier la liste suivante par sélection : `[12, 31, -2, -14, 8]`

1. Au premier parcours, la liste devient `[-14, 31, -2, 12, 8]` . Combien de comparaisons ont été nécéssaires pour identifier que -14 est le plus petit élement ?
2. Au deuxième parcours la liste devient `[-14, -2, 31, 12, 8]` . Combien de comparaisons ont été nécéssaires pour identifier que -2 est le plus petit des éléments restant à trier ?
3. Combien de comparaisons sont nécéssaires pour trier toute la liste ?
4. Est-ce vrai pour toutes les listes de cinq éléments ?

Réponse :

1.
2.
3.
4.

Exercice 3 :

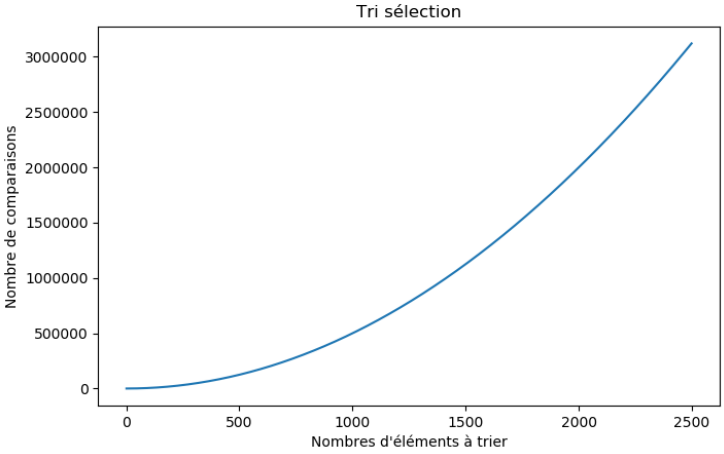
1. Combien de comparaisons sont nécéssaires pour trier par sélection une liste de 7 éléments ?
2. Et pour une liste de 10 éléments ?
3. Et pour n éléments ?

Réponse : En suivant le principe précédent

1. Pour 7 éléments :
2. Pour 10 éléments :
3. Pour n éléments :

A retenir

Quelque soit le nombre d'éléments n d'éléments à trier et dans tous les cas(par exemple même quand la liste est partiellement triée), le nombre de comparaisons est exactement $\frac{n\times(n-1)}{2} = \frac{n^2-n}{2}$, c'est à dire de l'ordre de n^2.



On dit que cet algorithme est de complexité quadratique en $\mathcal{O}(n^2)$.

Par exemple , une liste de 25000 éléments à trier nécéssite de l'ordre de $25000^2 = 625000000$ comparaisons.

Programmation

Recherche du minimum et échange

La première étape de l'algorithme est de rechercher le minimum de la liste et d'échanger sa place avec le premier élément:

(1) $L\ est\ une\ liste$
(2) $indexmin \leftarrow 0$
(3) $Pour\ j\ de\ 1\ à\ index\ du\ dernier\ élément\ de\ L$
(4) $\quad Si\ L[j] < L[indexmin]$
(5) $\quad\quad indexmin \leftarrow j$
(6) $L[0], L[indexmin] = L[indexmin], L[0]$

Exercice 4 :

1. Que représente la variable `indexmin` ?
2. Expliquer la ligne 6 de l'algorithme.
3. En remplaçant les commentaires, programmer cet algorithme.

Réponses :

1.
2.

In [ ]:

```
#3
L=[randint(-32,32) for i in range(5)]
print(L)

#ligne2
#ligne3
#ligne4
#ligne5
#ligne6

print(L)
```

Exercice 5 :

On suppose que le minimum de la liste est désormais le premier élément. L'étape suivante est d'identifier le plus petit élément parmi ceux restant à trier et de le placer en deuxième position.

1. Recopier et modifier l'algorithme précédent en ce sens. On utilisera la liste `[-29, 24, 31, -15, 8]`. l'instruction `print(L)` doit renvoyer `[-29, -15, 31, 24, 8]`.
2. Qu'a-t-on modifié précisement ?
3. Que doit-on modifier pour trouver le troisième élement de la liste triée ?

In [ ]:

```
#1.
L=[-29, 24, 31, -15, 8]

#ligne2
#ligne3
#ligne4
#ligne5

print(L)
```

1.

1. Pour trouver le troisième élément trié , il faut :
   -
   -
   -

Algorithme du tri par sélection

Exercice 6 :

La fonction `tri_select(L)` prend en paramètre une liste `L` d'entiers et doit renvoyer cette liste triée par sélection. En vous aidant des résultats précédents, compléter cette fonction en remplaçant les `?` .

In [1]:

```
#Algorithme de tri par sélection
def tri_select(L):
    for i in ?:
        imin = ?
        for j in range(?,len(L)):
            if L[j]<L[imin]:
                imin=j

        L[?],L[imin]=L[imin],L[?]

    return L

L=[randint(-32,32) for i in range(5)]
print (L)
print(tri_select(L))
```

```
  File "<ipython-input-1-679e15220855>", line 3
    for i in ?:
             ^
SyntaxError: invalid syntax
```

Remarques :

- On est sûr que l'algorithme s'arrête car on sort des boucles `for` une fois le parcours terminé.
- A la fin de l'étape i, les i premiers éléments du tableau sont triés, ce qui prouve que cet algorithme est correct (il fait ce que l'on attend de lui)

2. Le tri par insertion

Présentation

- C'est celui que l'on utilise intuitivement lorsque l'on trie des cartes à jouer dans sa main.
- Dans une liste, on commence par trier les deux premiers élements en insérant éventuellement le deuxième élément en première position.
- On insère tour à tour chaque élément non trié à sa place dans l'ensemble déjà trié précédemment.

Exemple :

On cherche à trier la liste suivante : `[29, -6, 12, -11, 10]` .

- On insère -6 en première position `[ -6, 29, 12, -11, 10]`
- On insère 12 en deuxième position `[ -6, 12, 29, -11, 10]`
- On insère -11 en première position `[ -11, -6, 12, 29, 10]`
- On insère 10 en troisième position `[ -11, -6, 10, 12, 29]`

Exercice 7:

1. En indiquant les listes intermédiaires obtenues, trier la liste `[5, -27, -14, 10, 7]` par insertion.
    -
    -
    -
2. Même exercice avec la liste `[30, 24, -7, -19, 26, 21, -18]`
    -
    -
    -
    -
    -
    -

Programmation

Insérer l'élément d'index `i`

- Sauvegarder `L[i]` dans une variable.
- Parcourir la partie de la liste déjà triée en commençant à l'index `i-1` jusqu'à trouver la position d'insertion en décalant dans le même temps les éléments pour pouvoir effectuer l'insertion.
- Insérer l'élément.

Exercice 8 :

On considère la liste `L=[2,15,23,10,5,1]` . Cette liste est triée jusqu'à l'index 2. En remplaçant les `?` , compléter le programme python qui doit permettre d'insérer l'élément d'index 3 à sa place.

L'instruction `print(L)` doit renvoyer `L=[2,10,15,23,5,1]` .

In [1]:

```
L=[2,15,23,10,5,1]

temp=L[3]
j=?
while L[j]>temp:
    L[?]=L[?]
    j=j-1
L[j+1]=?

print(L)
```

```
  File "<ipython-input-1-be5c9dd75216>", line 4
    j=?
      ^
SyntaxError: invalid syntax
```

Exercice 9 :

1. On considère la liste `L=[2,10,15,23,5,1]` . Cette liste est triée jusqu'à l'index 3. Recopier et modifier le programme python de l'exercice précédent pour permettre d'insérer l'élément d'index 4 à sa place. L'instruction `print(L)` doit renvoyer `L=[2,5,10,15,23,1]`

In [7]:

```
#1.
L=[2,10,15,23,5,1]
```

1. On considère la liste `L=[2,5,10,15,23,1]` . Le programme ci-dessous doit insérer le dernier élément à sa place,ici au début de la liste, mais il renvoie une erreur. Le modifier(on pourra faire afficher les différentes valeurs prises par `j` pour trouver l'erreur). L'instruction `print(L)` doit renvoyer `L=[1,2,5,10,15,23]` .

In [14]:

```
#2.
L=[2, 5, 10, 15, 23, 1]
temp=L[5]
j=4
while L[j]>temp:
    L[j+1]=L[j]
    j=j-1
L[j+1]=temp

print(L)
```

```
---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-14-c737d2a4290d> in <module>
      3 temp=L[5]
      4 j=4
----> 5 while L[j]>temp:
      6     L[j+1]=L[j]
      7     j=j-1

IndexError: list index out of range
```

Exercice 10 :

La fonction `tri_insert(L)` prend en paramètre une liste `L` d'entiers et doit renvoyer cette liste triée par sélection. En vous aidant des résultats précédents, compléter cette fonction en remplaçant les `?` .

In [ ]:

```
#Algorithme du tri par insertion
def tri_insert(L):
    for i  in range(?) :
        tmp = L[?]
        j=?
        while j>=0 and L[j]>tmp:
            L[j+1] = L[j]
            j = j-1
        L[j+1] = tmp
    return L

L=[randint(-32,32) for i in range(10)]
print (L)
print(tri_insert(L))
```

Remarques

- La complexité de cet algorithme est quadratique, en $\mathcal{O}(n^2)$.
- Cependant , cet algorithme est plus efficace que le tri sélection lorsqu'une partie de la liste est triée.
- On est sur que cet algorithme se termine car :
  - On sort de la boucle `for` une fois le parcours terminé.
  - Le booléen `j>=0` et l'instruction `j=j-1` font que la boucle `while` finit par se terminer.

3. Conclusion

Python sait trier...

La méthode `.sort()`

In [7]:

```
L=[randint(-32,32) for i in range(20)]
print(L)

# Tri ascendant :
L.sort()
print(L)

# Tri descendant
L.sort(reverse=True)
print(L)
```

```
[2, 13, 10, 31, 4, -8, 25, 9, 3, -20, 23, -23, -18, -26, 2, -21, 11, 24, -
26, -9]
[-26, -26, -23, -21, -20, -18, -9, -8, 2, 2, 3, 4, 9, 10, 11, 13, 23, 24,
25, 31]
[31, 25, 24, 23, 13, 11, 10, 9, 4, 3, 2, 2, -8, -9, -18, -20, -21, -23, -2
6, -26]
```

La fonction `sorted()`

In [6]:

```
L=[randint(-32,32) for i in range(20)]
print(L)

# Tri ascendant :
L1 = sorted(L)
print(L1)

# Tri descendant
L2 = sorted(L,reverse=True)
print(L2)
```

```
[-29, 21, 24, -7, 29, -19, -29, 26, -28, 11, -32, -15, 18, -4, -10, -29, 1
7, -9, 32, 12]
[-32, -29, -29, -29, -28, -19, -15, -10, -9, -7, -4, 11, 12, 17, 18, 21, 2
4, 26, 29, 32]
[32, 29, 26, 24, 21, 18, 17, 12, 11, -4, -7, -9, -10, -15, -19, -28, -29,
-29, -29, -32]
```

Exercice 11

Le fichier `mots.txt` contient une liste de mots séparés par un retour à la ligne. On souhaite :

- Classer ces mots selon leur première lettre dans des fichiers nommés `a`, `b`, ... , `z`.
- Que les mots contenus dans chacun de ces fichiers soit classés par ordre alphabétique

Compléter la fonction `insere(mot)`.

In [3]:

```
def insere(mot):
    nomfichier=mot[0]
    f=open(nomfichier,'a+')
    f.seek(0)
    if mot not in f.readlines():
        f.write(mot)
    #à compléter



def classe(fichier) :
    f=open(fichier,'r')
    for mot in f.readlines():
        insere(mot)
    f.close()

classe('mots.txt')
```

Algorythmics

1. Vidéo 1 : De quel tri s'agit-il ? (https://www.youtube.com/watch?v=ROalU379l3U)
2. Vidéo 2 : De quel tri s'agit-il ? (https://www.youtube.com/watch?v=Ns4TPTC8whw)

En savoir plus

- Il existe des dizaines d'algorithmes de tri, nous en étudierons d'autres en terminale, bien plus efficaces que le tri sélection ou insertion!
- Visualiser et comparer 24 algorithmes de tri en 2 minutes (https://www.youtube.com/watch?v=BeoCbJPuvSE)
- Pour en savoir plus : http://lwh.free.fr/pages/algo/tri/tri.htm (http://lwh.free.fr/pages/algo/tri/tri.htm)

FIN