Tri fusion (principe)

On dispose d'une liste de nombre non triée

0	1	2	3	4	…	…	N-2	N-1	N
12	2	29	102	38	…	…	6	50	26

L'idée du tri-fusion est de :

- diviser la liste en 2 listes

0	1	2	3	4	…	N/2
12	2	29	102	38	…	…

N/2 +1	…	…	N-2	N-1	N
…	…	…	6	50	26

- Trier les 2 listes

0	1	2	3	4	…	N/2
2	12	29	38	102	…	…

N/2 +1	…	…	N-2	N-1	N
…	…	…	6	26	50

- Fusionner les 2 listes triées

0	1	2	3	4	…	…	N-2	N-1	N
2	6	12	26	29	38	50	…	…	…

L’appel récursif

Chaque demi-liste sera traitée de la même manière :

- Diviser
- Trier
- Fusionner

Jusqu’à ce que la demi-liste ne contienne plus qu’un élément (celui-ci constitue donc une liste triée)

Algorithme

Rappel de la diapo 1 :
L'idée du tri-fusion est de :

- diviser la liste en 2 listes
- Trier les 2 listes
- Fusionner les 2 listes triées

On considère les fonctions :

diviser(L) :

renvoie 2 listes (1ère moitié de L et 2ème moitié de L)

trier(L) :

Renvoie une liste triée avec les éléments de L

fusionner(L1,L2)

Renvoie une liste issue de la fusion des listes triées L1 et L2

Algorithme

Rappel de la diapo 1 :
L'idée du tri-fusion est de :

- diviser la liste en 2 listes
- Trier les 2 listes
- Fusionner les 2 listes triées

```
Fonction trier (L):
        si longueur(L)<>1 alors
                L1,L2 = diviser(L)
                L3 = trier(L1)
                L4 = trier(L2)
                retourner fusionner(L3,L4)
        sinon retourner L
```

Appels récursifs

Cas de base, arrêt des appel récursifs

Algorithme

Rappel de la diapo 1 :
L'idée du tri-fusion est de :

- diviser la liste en 2 listes
- Trier les 2 listes
- Fusionner les 2 listes triées

```
Fonction trier (L):
        si longueur(L)<>1 alors
                L1,L2 =
diviser(L)
                L3 = trier(L1)
                L4 = trier(L2)
                retourner
fusionner(L3,L4)
         sinon retourner L
```

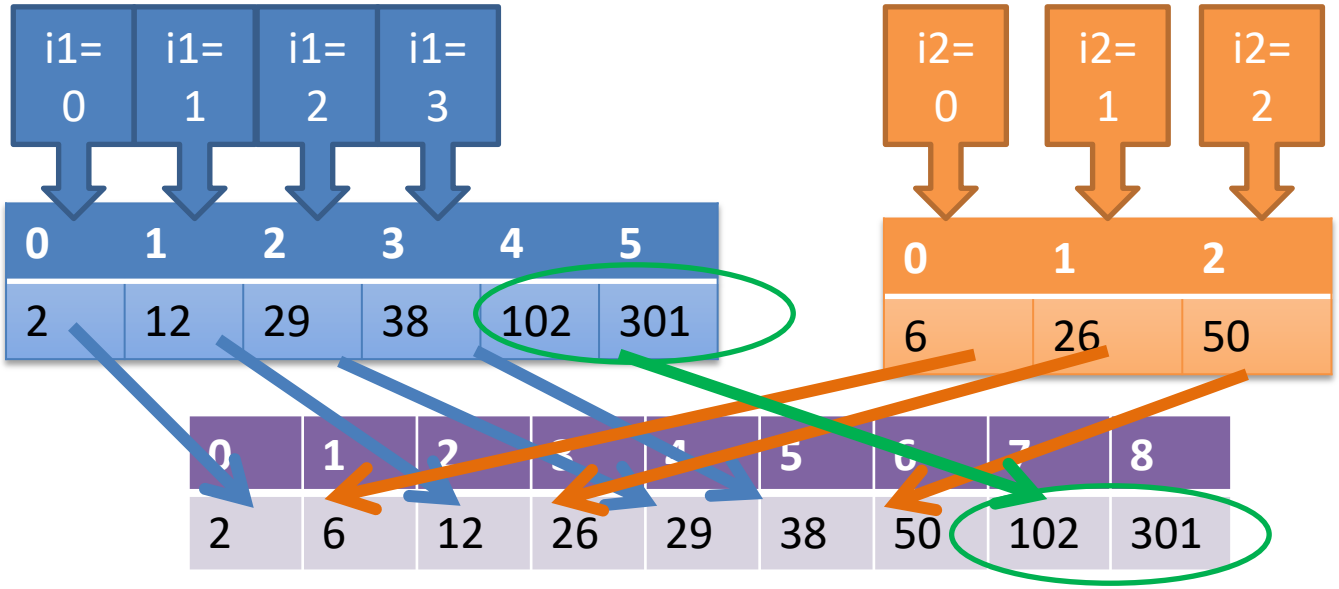
Version condensée en python :

```
def trier(L):
    if len(L)!=1:
        return fusionner(trier(L[:len(L)//2]),trier(L[len(L)//2:]))
    return L
```

Et l'interclassement alors ?

L'interclassement se gère assez facilement en parcourant les 2 listes :

- tant qu'il reste des éléments dans les 2 listes,
- On ajoute l'éléments de gauche le plus petits parmi les 2 listes,
- Quand une liste n'a plus d'éléments on ajoute la fin de la 2ème.



En python…

En parcourant les 2 listes :

- tant qu'il reste des éléments dans les 2 listes,
- On ajoute l'éléments de gauche le plus petits parmi les 2 listes,
- Quand une liste n'a plus d'éléments on ajoute la fin de la 2ème.

```
def interclassement2 (L1 , L2):
    lst = [ ]
    i1 = 0
    i2 = 0
    while i1<len(L1) and i2<len(L2):
        if L1[i1]<L2[i2]:
            lst.append(L1[i1])
            i1+=1
        else:
            lst.append(L2[i2])
            i2+=1
    return lst+L1[i1:]+L2[i2:]
```

Visualisation (et écoute) du tri

Sound of sorting

Téléchargement :
https://panthema.net/2013/sound-of-sorting/