LE TRI FUSION (MERGE SORT)

Méthode « Diviser pour mieux régner »

1. Présentation générale du tri fusion

Le **tri fusion** est un algorithme de tri basé sur la méthode **« diviser pour mieux régner »**.

Son principe est le suivant :

1. Diviser le tableau en deux parties.
2. Trier chaque partie.
3. Fusionner les deux parties triées pour obtenir un tableau trié.

Le tri fusion est :

- **efficace** (complexité en $O(n \log\ n)$),
- **stable**,
- naturellement adapté à une **implémentation récursive**.

2. Étape essentielle : la fusion de deux tableaux triés

Avant de trier récursivement, il faut savoir **fusionner deux tableaux déjà triés**.

Principe de la fusion

- On compare les premiers éléments des deux tableaux.
- On place le plus petit dans le tableau résultat.
- On recommence jusqu'à épuisement d'un des tableaux.
- On ajoute les éléments restants.

3. Version classique (itérative) du tri fusion

3.1 Fonction de fusion (itérative)

```
def fusion(t1, t2):
    resultat = []
    i = 0
    j = 0
    while i < len(t1) and j < len(t2):
        if t1[i] <= t2[j]:
            resultat.append(t1[i])
            i += 1
        else:
            resultat.append(t2[j])
            j += 1

    resultat.extend(t1[i:])
    resultat.extend(t2[j:])
    return resultat
```

Explication

- i et j parcourent respectivement t1 et t2.
- On ajoute à resultat le plus petit élément disponible.
- La boucle s'arrête quand un tableau est vide.
- Les éléments restants sont ajoutés à la fin.

3.2 Tri fusion itératif (approche classique)

```
def tri_fusion_iteratif(t):
    listes = [[x] for x in t]
    while len(listes) > 1:
        nouvelles_listes = []
        for i in range(0, len(listes), 2):
            if i + 1 < len(listes):
                nouvelles_listes.append(fusion(listes[i],
listes[i + 1]))
            else:
                nouvelles_listes.append(listes[i])

        listes = nouvelles_listes

    return listes[0] if listes else []
```

Explication

- Chaque élément est d'abord considéré comme un tableau trié de taille 1.
- On fusionne les tableaux deux à deux.
- À chaque étape, le nombre de tableaux diminue.
- On répète jusqu'à obtenir un seul tableau trié.

👉 Cette version **n'utilise pas la récursivité**, mais applique bien la stratégie « diviser pour régner ».

4. Version récursive du tri fusion

4.1 Principe récursif

- **Cas de base** :
  Un tableau de taille 0 ou 1 est déjà trié.
- **Cas récursif** :
  - on coupe le tableau en deux,
  - on trie récursivement chaque moitié,
  - on fusionne les deux tableaux triés.

4.2 Code du tri fusion récursif

```
def tri_fusion(t):
    if len(t) <= 1:
        return t

    milieu = len(t) // 2
    gauche = tri_fusion(t[:milieu])
    droite = tri_fusion(t[milieu:])

    return fusion(gauche, droite)
```

4.3 Explication détaillée

- La fonction s'appelle elle-même sur des tableaux de plus en plus petits.
- Chaque appel génère **deux appels récursifs** → récursivité multiple.
- La récursion s'arrête lorsque la taille du tableau vaut 0 ou 1.
- Les résultats sont combinés grâce à la fonction fusion.

5. Comparaison itératif / récursif

Version itérative	**Version récursive**
Plus longue	Plus concise
Moins intuitive	Correspond à la définition
Pas de pile d'appels	Utilise la pile
Plus complexe à écrire	Plus naturelle

6. À retenir (à faire copier aux élèves)

Le tri fusion est un algorithme « diviser pour mieux régner ».
Il divise le tableau en sous-tableaux, les trie récursivement, puis fusionne les résultats.
La version récursive est la plus naturelle pour implémenter cet algorithme.

7. Conclusion pédagogique

- Le tri fusion illustre parfaitement la **récursivité multiple**.
- Il montre que la récursivité est une **stratégie algorithmique**, pas seulement une technique de programmation.
- La version itérative et la version récursive résolvent le même problème, mais de manière différente.