Algorithmique

DIVISER POUR REGNER (Divide and conquer)

Le diviser pour régner est une méthode algorithmique basée sur le principe suivant :
On prend un problème (généralement complexe à résoudre), **on divise ce problème en une multitude de petits problèmes**, l'idée étant que les "petits problèmes" seront plus simples à résoudre que le problème original. Une fois les petits problèmes résolus, on recombine les "petits problèmes résolus" afin d'obtenir la solution du problème de départ.

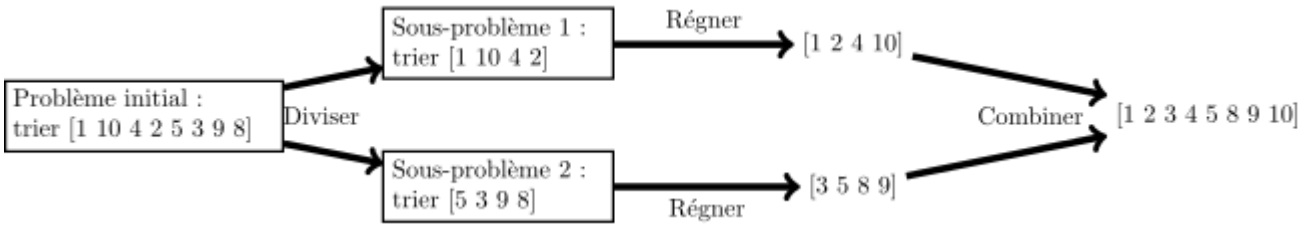
Le paradigme "diviser pour régner" repose donc sur 3 étapes :

- **DIVISER** : le problème d'origine est divisé en un certain nombre de sous-problèmes
- **RÉGNER** : on résout les sous-problèmes (les sous-problèmes sont plus faciles à résoudre que le problème d'origine)
- **COMBINER** : les solutions des sous-problèmes sont combinées afin d'obtenir la solution du problème d'origine.

Exemple : le tri fusion (merge sort)

On connait déjà 2 algorithmes de tri : par insertion et par sélection. On se souvient que ces deux tris ont une complexité quadratique (en n^2).

La méthode algorithmique « diviser pour régner » peut s'illustrer de la manière suivante pour le tri :



(on peut imaginer que les deux sous-problème 1 et 2 seront eux aussi résolus par la méthode « diviser pour régner ». Ca sent très fort la récursivité…)

Complexité

La faible complexité des algorithmes *diviser pour régner* est l'un de leurs principaux intérêts. La table suivante compare la complexité d'un algorithme naïf et de l'algorithme diviser pour régner pour quelques problèmes:

	Complexité avec l'algorithme naïf	Complexité avec l'algorithme diviser pour régner
Recherche dans un tableau trié de taille n	$O(n)$	$O(\log n)$ avec la recherche dichotomique
Tri d'un tableau de n éléments	$O(n^2)$ (tri par insertion et tri par sélection)	$O(n \log n)$ avec le tri fusion $O(n \log n)$ en moyenne avec le tri rapide
Multiplication de deux nombres avec n chiffres	$O(n^2)$	$O(n^{1,585})$ avec l'algorithme de Karatsuba

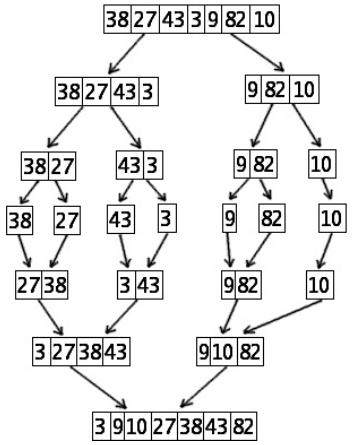
Parallélisme

Les algorithmes *diviser pour régner* sont souvent adaptés pour être exécutés sur des machines avec plusieurs processeurs.

Le TRI FUSION en détail :

Pour comprendre le principe, assez simple, regardez cette animation :
http://ressource.elec.free.fr/docs/NSI/diviser_pour_regner/Merge-sort-example-300px.gif

Et ce graphique :



Identifiez les étapes de « division » et les étapes De « fusion »

Puisqu'il faut diviser puis fusionner, l'algorithme se décomposera en deux : un algorithme de fusion et un algorithme de division :

FUSION :

entrée : *deux listes triées A et B*
sortie : une liste triée qui contient exactement les éléments des listes A et B

```
fonction fusion(A, B)
    si A est le tableau vide
        renvoyer B
    si B est le tableau vide
        renvoyer A
    si A[1] ≤ B[1]
        renvoyer A[1] ⊕ fusion(A[2, …], B)
    sinon
        renvoyer B[1] ⊕ fusion(A, B[2, …])
```

Aide :
⊕ -> concaténation

Rappel en python :
concaténer des listes :
[1] + [12] donne [1,12]

Utiliser juste une partie d'une liste :
L[1 :] -> toute la liste à partir de l'indice 1 inclus
L[:4] -> toute la liste jusqu'a l'indice 4 exclu
L[1:4] -> la liste entre l'indice 1 et l'indice 4

TRI FUSION (qui appelle *fusion()*) :

entrée : *un tableau T*
sortie : *une permutation triée de T*

```
fonction triFusion(T[1, …, n])
    si n ≤ 1
        renvoyer T
    sinon
        p = partie entière(n/2)
        renvoyer fusion(triFusion(T[1, … p]), triFusion(T[p + 1, ….]))
```

Implantez cet algorithme en Python