Diviser pour régner

Le principe *diviser pour régner* ("divide and conquer") est un principe militaire et politique ancestral. Déjà mise en avant par Jules César. Elle est reprise également dans les écrits politiques de Nicolas Machiavel (L'art de la guerre).

> "Divide et impera" : "Divise et règne"

En informatique, *diviser pour régner* est une méthode de conception d'algorithmes **réduisant récursivement un problème en un ou plusieurs sous-problèmes du même type (ou de la même classe de problème)**.

Trois étapes de mise en oeuvre :

1. **DIVISER** : On découpe le problème initial en sous-problèmes de taille plus faible qui lui sont identiques, indépendants les uns des autres.
2. **REGNER** : On résout chacun des sous-problèmes récursivement ou directement s'ils sont élémentaires (cas de base)
3. **COMBINER** : On construit la solution du problème combinant les solutuons des sous-problèmes. *Cette 3ème étape n'est pas toujours présente*.

Exemple 1 : Recherche dichotomique d'une valeur dans une liste triée

Si nous reprenons cet exemple :

1. **DIVISER** : Découper la liste triée en deux.
2. **REGNER** : Chercher la valeur x dans le sous-tableau qui convient
3. **COMBINER** : Rien à faire !

Exemple 2 : TRI FUSION

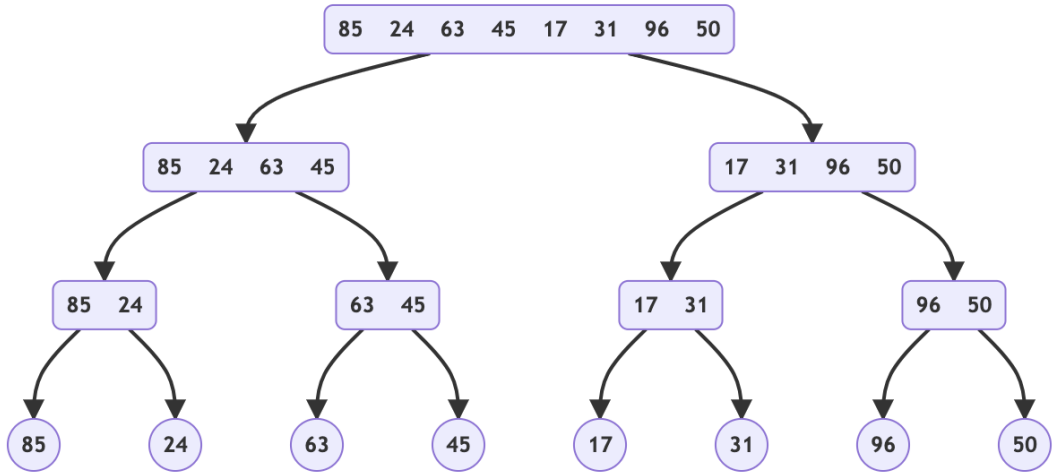
Nous avons vu en classe de première deux exemples de tri : par insertion et par sélection.

Le tri fusion est un autre exemple de tri, non intuitif mais très intéressant en terme de complexité.

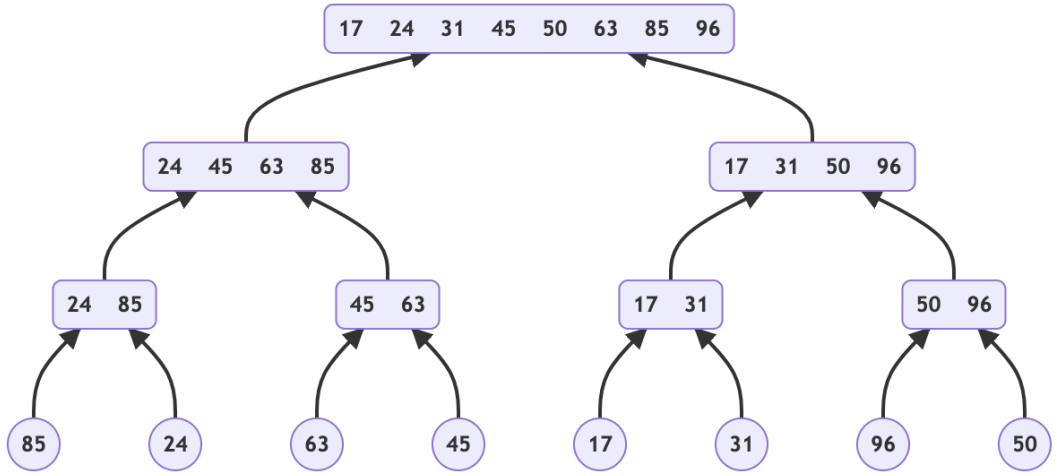
Principe

1. **DIVISER** : Découper la liste triée en deux.
2. **REGNER** : Trier récursivement les deux sous-tableaux
3. **COMBINER** : Fusionner les deux demi-tableaux triés obtenus

Exemple



> *Résultats des différents appels récursifs (Partie* ***Diviser****).*



> *Résultats progressifs après les étapes* ***Régner*** *et* ***Fusionner****.*

Algorithme

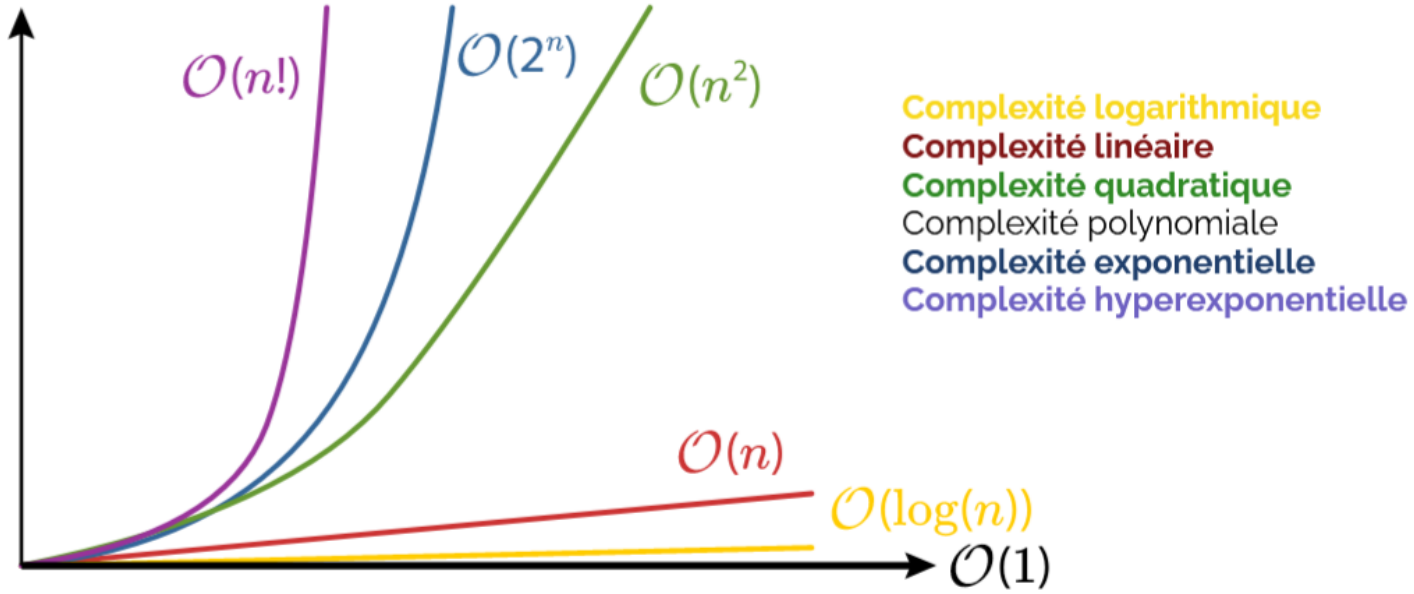
```
triFusion(t)
     n = len(t)
     c = n//2
     t1 = triFusion(t[0,c[)  {tri du sous-tableau t[0 :c]}
     t2 = triFusion(t[c,n[)  {tri du sous-tableau t[c :n]}
     fusionner(t1, t2) {fusion des deux sous-tableaux triés}

fusionner(l1,l2)
     si l1 = []
          l2
     si l2 = []
          l1
     si l1[0] < l2[0]
          l1[0] + fusionner(l1[1 :], l2)
     sinon l2[0] + fusionner(l1, l2[1 :])
```

A faire :

1. Implémenter l'algorithme en Python (*spécification, assertion -vérifier type liste- et jeu de test attendu*)
2. Compléter votre algorithme pour calculer le temps d'exécution de la fonction sur des listes de taille variable
3. Noter vos résultats dans un tableur. *Compléter si possible avec les valeurs trouvées pour les tris par insertion et sélection réalisés en 1ère*.
4. Réaliser un graphique : temps d'exécution en fonction de la taille de la liste à trier.

Complexité

Tri fusion : $\theta(n \log_2 n)$ Tri par insertion et tri par sélection $\theta(N^2)$



L'algorithme qui a la plus faible complexité est toujours le meilleur.

Compléments

Cours sur le manuel : p302 à 307
Exercices d'entraînement : 12.1p320, 12.2p320 et 12.3p321

Ce que je dois savoir-faire :

- Identifier les différentes étapes de la stratégie « diviser pour régner » dans un problème
- Écrire/implémenter un algorithme utilisant la méthode « diviser pour régner »
- Evaluer le coût **nlog(n)** d'un algorithme « diviser pour régner » et son avantage par rapport à d'autres algorithmes *(cas à connaître : tri fusion par rapport aux tris par sélection et par insertion vus en 1ère)*.