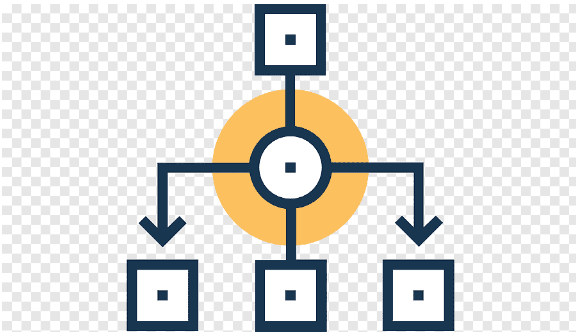
# Algorithmique Avancée

1.0



Noureddine Amraoui
Maître assistant classe B
Université Mohamed Boudiaf de M'Sila

24 avril 2022

# Table des matières

# Objectifs

A l'issu de cet enseignement, l'apprenant sera capable de :

- En terme de Savoir :
  a. Maîtriser les bases de l'analyse algorithmique.
  b. Identifier les stratégies de résolution de problèmes.
  c. Connaître les classes de problèmes.
- En terme de Savoir-faire :
  a. Analyser et classer les problèmes de différents domaines.
  b. Construire la ou les solutions.
  c. Évaluer les différentes solutions en terme de calcul de complexité.
  d. Choisir la meilleure solution.
- En terme de Savoir-être :
  a. Vous sensibiliser à la résolution des problèmes.

# I.Chapitre 2 : La méthode Diviser Pour Régner (DPR)

## 1. Pré-requis

L'étudiant doit connaître :

- Les algorithmes itératifs et récursifs.
- Les structures de données fondamentales : tableaux, fichiers, listes, piles, et files.

## 2. Pre-test

### 2.1. Exercice : Maîtriser-vous la récursivité ?

[Solution n°1 p 10]

*La récursivité est une méthode dans laquelle la solution d'un problème dépend de :*

- ○ Instances plus importantes de problèmes différents.
- ○ Instances plus importantes du même problème.
- ○ Petites instances du même problème.
- ○ Petites instances de différents problèmes.

### 2.2. Exercice : Pouvez-vous différencier entre les problème de récursivité ?

[Solution n°2 p 10]

*Lequel des problèmes suivants ne peut pas être résolu à l'aide de la récursivité ?*

- ○ Factorielle d'un nombre.
- ○ Suite de Fibonacci.
- ○ Longueur d'une chaîne.
- ○ Problèmes sans cas de base.

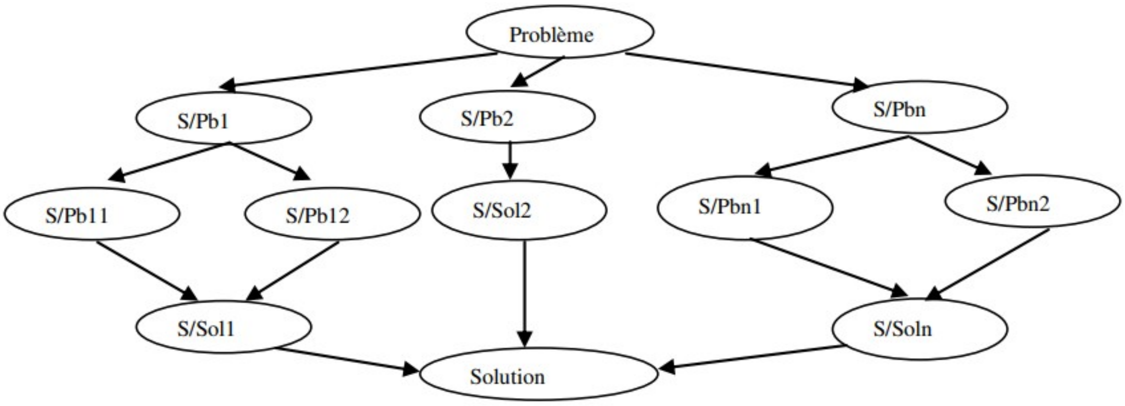
## 3. Principe

*Définition*

Diviser pour régner (**divide and conquer**), on dit aussi « diviser pour résoudre », fut une célèbre stratégie militaire avant de devenir une technique de conception algorithmique. Les généraux remarquèrent qu'il était plus facile de vaincre une armée de 50.000 hommes puis une autre de la même taille que de directement affronter une seule armée de 100.000 hommes.

Cette technique appliquée à l'informatique a souvent permis pour un problème donné de construire des algorithmes de plus basse complexité.

Diviser pour régner est une technique algorithmique consistant à diviser un problème de grande taille en plusieurs sous-problèmes analogues. L'étape de subdivision est appliquée récursivement[2].




*Principe de diviser pour régner*

# 4. Schéma général

## *Conseil*

Trois décisions à prendre pour avoir un algorithme diviser-pour-régner efficace :[3]

- **Quand est x « petit ou simple ? »** Détermination d'un seuil adéquat. Bien décider quand utiliser l'algorithme simple sur de petites instances plutôt que les appels récursifs.
- **Comment décomposer x ?** Nombre de morceaux, de taille équilibrée. Les sous-instances doivent être, autant que possible, environ de la même taille. La décomposition d'une instance en sous-instances doit être efficace.
- **Comment recombiner les solutions des sous-exemplaires ?** une façon efficace de profiter du travail accompli. La recombinaison des sous-solutions doit être efficace.

L'algorithme générique de diviser pour régner peut être résumé comme suit :

```
Fonction Diviser_Régner (P : Problème) : Solution
Variables S : Solution ;
Début
 Si (||P|| est petite) Alors S ← CasBase(P)
 Sinon
 (P1, P2, ..., Pk ) ← Diviser (P) ;
 Pour i ← 1 à k Faire Si ← Regner (Pi) ; FinPour
 S ← Combiner (S1, S2, ..., Sk ) ;
 FinSi
 Retourner S ;
Fin
```

# 5. Exemples classiques

## 5.1. Tri par fusion (MergeSort)

### *Définition*

Ce tri est basé sur la technique algorithmique diviser pour régner. L'opération principale de l'algorithme est la fusion, qui consiste à réunir deux listes triées en une seule. L'efficacité de l'algorithme vient du fait que deux listes triées peuvent être fusionnées en temps linéaire.

L'algorithme de tri par fusion est construit suivant le paradigme « diviser pour régner » :[4]

1. **Diviser** : on divise la séquence de $n$ éléments à trier en deux sous-séquences de $n$ éléments ($T[1, . . n2]\ et\ T[n2+ 1, . . n]$),
2. **Régner** :
   - On trie les deux sous-séquences à l'aide du tri par fusion (appel récursif),
   - Toute séquence de longueur 1 est triée (donc on ne fait rien).
3. **Combiner** : on combine, en les fusionnant, les deux sous-séquences triées pour produire la séquence complète triée.

**Algorithme :** il nous faut donc :

- une fonction qui fusionne deux sections contiguës du tableau.
- une fonction de tri proprement dit qui lance des appels récursifs sur les sous-parties puis appelle la fonction précédente

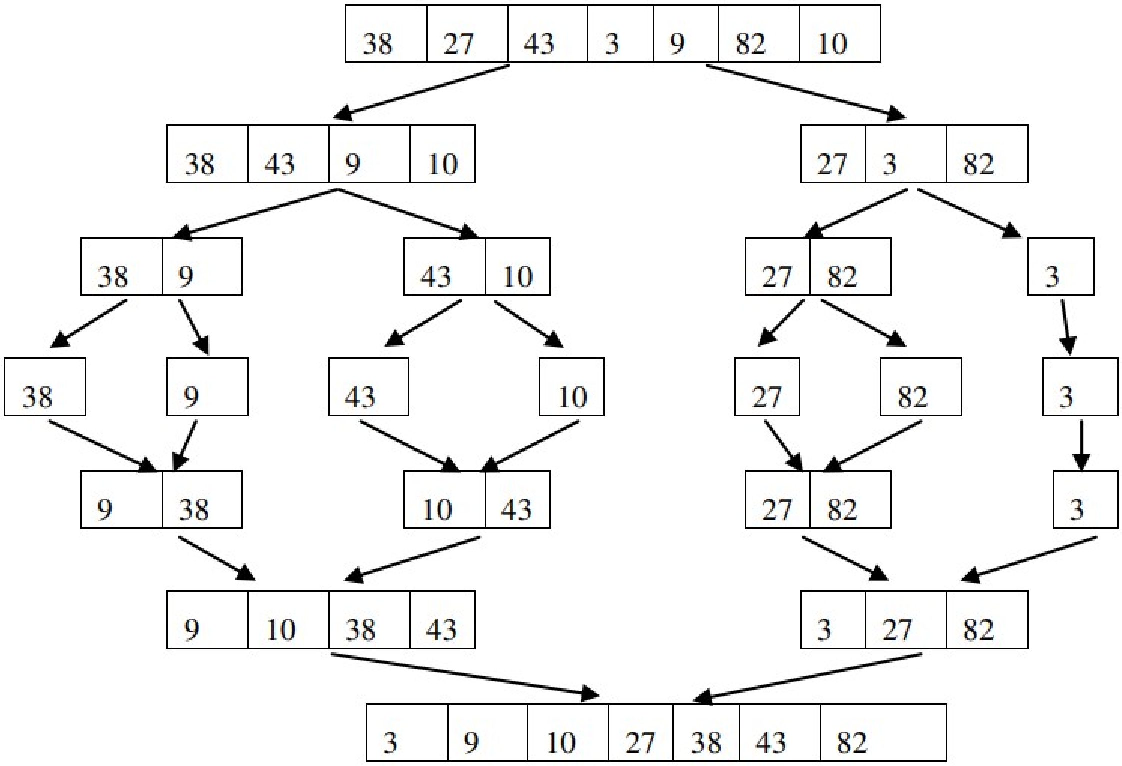
En pseudo-code, l'algorithme pourrait s'écrire ainsi :

```
fonction scinder(liste0) :
 si longueur(liste0) <= 1, renvoyer le couple (liste0, liste_vide)
 sinon
 soient e1 et e2 les deux premiers éléments de liste0, et reste le reste de
 liste0
 soit (liste1, liste2) = scinder(reste)
 renvoyer le couple de listes (liste de tête : e1 et de queue : liste1, liste de
 tête : e2 et de queue : liste2)

fonction fusionner(liste1, liste2) :
 si la liste1 est vide, renvoyer liste2
sinon si la liste2 est vide, renvoyer liste1
 sinon
 si tête(liste1) <= tête(liste2), renvoyer la liste de tête : tête(liste1) et de
 queue :
fusionner(queue(liste1),liste2)
 sinon, renvoyer la liste de tête : tête(liste2) et de queue :
 fusionner(liste1,queue(liste2))

fonction triFusion(liste0) :
 si longueur(liste0) <= 1, renvoyer liste0
 sinon,
 soit (liste1, liste2) = scinder(liste0)
 renvoyer fusionner(triFusion(liste1), triFusion(liste2))
```

**Exemple :** $T$ = [38,27,43,3,9,82,10]. L'arbre de récursion explicite le principe de division (D) et de fusionnement (F) de l'algorithme.




*. Exemple de Tri par fusion appliqué à un tableau de 7 éléments.*

## 5.2. Tri rapide (Quicksort)

### *Définition*

Proposé par Tony Hoare en 1960, le tri rapide est un exemple de récursivité sur le résultat. La donnée initiale et le résultat sont les mêmes que pour le tri par fusion.

Cette opération s'appelle le partitionnement. Pour chacun des sous-tableaux, on définit un nouveau pivot et on répète l'opération de partitionnement. Ce processus est répété récursivement, jusqu'à ce que l'ensemble des éléments soit trié.

1. **Diviser** :
   - $T[p..r]$ est divisé en 2 sous-tableaux non vide $T[p..q]$ et $T[q+1..r]$ tel que : chaque élément de $T[p..q]$ soit inférieur à chaque élément de $T[q+1..r]$.
   - fonction Partitionner
2. **Régner** :
   - sous-tableaux triés grâce à la récursivité.
   - fonction $TriRapide$
3. **Combiner** : 2 sous-tableaux triés sur place : Rien à faire.

En pseudo-code, l'algorithme pourrait s'écrire ainsi (le pivot est l'élément le plus à gauche du tableau) :

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2 | 6 | 9 | 1 | 3 | 8 | 4 | 5 | 9 |

- Deux pointeurs $i$, initialisé à 1, $j$ initialisé à $taille(tab)$
- Bouger $i$ vers la droite jusqu'à un élément $supérieur\ au$ pivot
- Bouger $j$ vers la gauche jusqu'à un élément $inférieur\ ou\ égal$ pivot
- Échanger $tab[i]$ et $tab[j]$ et répéter tant que i < $j$
- Échanger $pivot$ et $tab[j]$

```
procedure tri_rapide (c_cle[] tab, indice g, indice d)
      // trie la partie g..d du tableau
      indice i,j ;
      c_cle pivot;
debut
      si g<d alors
            pivot = tab(g);
            i = g;
            j = d;
            tant que (i<j) faire
                  //invariant les elem de g a i sont <= p
                  //les elem de j+1 a d sont > p
                  i := i + 1;
                  tant que((tab(i)<=pivot) et (i<j))faire i= i+1;fait
                  tant que (tab(j) > pivot) faire j = j - 1; fait
                  si (i < j) alors echanger (tab, i, j); fin si
            fait
            echanger (tab, g, j); //place definitive du pivot
            tri_rapide (tab, g, j - 1);
            tri_rapide (tab, j + 1, d);
      fin si
fin
```

*Tri rapide (Quicksort)*

### 5.3. Algorithme de Strassen pour la multiplication de matrices

#### *Méthode*

Soient $A$, $B$ deux matrices carrées d'ordre $n$ et $C$ leur produit. La façon usuelle de calculer $C$,

$$C = AB = (c_{ij}) ; 1 \leqslant i , j \leqslant n \text{ avec } c_{ij} = \sum a_{ik} b_{kj}$$

**Algorithme naïf**

L'algorithme classique est le suivant :

```
MULTIPLIER-MATRICES(A, B)
Soit n la taille des matrices carrés A et B
Soit C une matrice carré de taille n
Pour i← 1 à n faire
 Pour j← 1 à n faire
 ci ; j ← 0
 Pour k←1 à n faire
 ci; j ← ci; j + ai; k * bk; j
 Fin pour
 Fin pour
Fin pour
Renvoyer C
```

## 6. Exercices

### 6.1. Exercice

[Solution n°3 p 10]

*Laquelle des techniques de conception d'algorithmes suivantes est utilisée dans l'algorithme de tri rapide ?*

- ○ Programmation dynamique.
- ○ Retour en arrière.
- ○ Diviser pour régner.
- ○ Méthode gourmande.

# Solution des exercices

## > Solution n°1 (exercice p. 4)

*La récursivité est une méthode dans laquelle la solution d'un problème dépend de :*

- ○ Instances plus importantes de problèmes différents.
- ○ Instances plus importantes du même problème.
- ◉ Petites instances du même problème.
- ○ Petites instances de différents problèmes.

## > Solution n°2 (exercice p. 4)

*Lequel des problèmes suivants ne peut pas être résolu à l'aide de la récursivité ?*

- ○ Factorielle d'un nombre.
- ○ Suite de Fibonacci.
- ○ Longueur d'une chaîne.
- ◉ Problèmes sans cas de base.

## > Solution n°3 (exercice p. 8)

*Laquelle des techniques de conception d'algorithmes suivantes est utilisée dans l'algorithme de tri rapide ?*

- ○ Programmation dynamique.
- ○ Retour en arrière.
- ◉ Diviser pour régner.
- ○ Méthode gourmande.

# Références

[2] Cedric Chauve, « Diviser-pour-régner (Résumé du chapitre 2 du manuel) », INF7440-Conception et analyse d'algorithmes, Université du Québec, Montréal, 2005

http://www.lacim.uqam.ca/~chauve/Enseignement/INF7440/H05/BASE/INF7440-diviser-regner.pdf

[3] Gérard Sookahet , « Algorithme de V. Strassen pour la multiplication rapide de matrices », Août 1997

http://gersoo.free.fr/Download/docs/stras.pdf

[4] Thomas Cluzeau, « Calcul Formel, Notes de Cours », Master 1, Université de Limoges – CNRS, 2016

http://www.unilim.fr/pages_perso/thomas.cluzeau/Enseignement/PolyCalculFormel.pdf