

# Algorithmique et Structures de données 1

LIV 2022-2023  
Travaux Dirigés 4

---

Site du cours : <https://defelice.up8.site/algo-struct.html>

Les exercices marqués de (@) sont à faire dans un second temps.

---

## Exercice 1. Esclavage

Ranger les suites suivantes de la plus dominée à la plus dominante. Indiquer les suites égales.

- |                            |                               |                        |                           |
|----------------------------|-------------------------------|------------------------|---------------------------|
| — $n \mapsto (1,000001)^n$ | — $n \mapsto n!$              | — $n \mapsto 3$        | — $n \mapsto n^4$         |
| — $n \mapsto n \cdot n$    | — $n \mapsto n^1$             | — $n \mapsto 2^n$      | — $n \mapsto \ln(n)$      |
| — $n \mapsto n^n$          | — $n \mapsto n^{\frac{1}{5}}$ | — $n \mapsto n^{0,25}$ | — $n \mapsto \sqrt[5]{n}$ |
| — $n \mapsto n^{0,5}$      | — $n \mapsto n^{1000}$        | — $n \mapsto n$        | — $n \mapsto \log_2(n)$   |
| — $n \mapsto \sqrt{n}$     | — $n \mapsto n^{0,2}$         | — $n \mapsto n^2$      | — $n \mapsto 3^n$         |

## Exercice 2. Segregation

Pour chaque suite trouver un équivalent plus simple de sa classe de domination  $\Theta$ , Puis ranger les suites suivantes par ordre de domination en indiquant les suites qui sont dans la même classe.

- |                              |                                |                                   |                               |
|------------------------------|--------------------------------|-----------------------------------|-------------------------------|
| — $n \mapsto 2^n + n^{1000}$ | — $n^n$                        | — $n \mapsto n\sqrt{n}$           | — $n \mapsto 45n^4 + n^2 + 1$ |
| — $n \mapsto 3^n$            | — $n!$                         | — $n \mapsto n(n^{1,5} + 3n + 1)$ |                               |
| — $n \mapsto \log_2^2(n)$    | — $n \mapsto n^{1,5} + 3n + 1$ | — $n \mapsto n \log n$            |                               |

## Exercice 3. Recherche dichotomique

1. Écrire une fonction `int recherche(int n, int* tab, int val)`. La fonction doit renvoyer l'indice d'une case contenant la valeur `val` ou renvoyer -1 si aucune case ne contient `val`. Le tableau `tab` est supposé être trié de taille `n`. La fonction doit utiliser environ  $\Theta(\log_2(n))$  opérations dans le pire des cas.

## Exercice 4. Fusion

L'algorithme de << tri fusion >> est un algorithme de tri récursif qui procède en plusieurs étapes dont voici une idée.

1. On découpe le tableau à trier en deux parties de taille à peu près égales puis on utilise l'algorithme pour trier séparément les deux parties.
2. Ensuite on fusionne les deux tableaux pour produire un tableau trié en utilisant le moins d'opérations possibles.

On souhaite implanter l'algorithme en C. On suppose que l'on a à disposition une fonction `void copie(int n, int* dest, int* src)` qui recopie le tableau `src` de longueur `n` dans `dest`.

1. Écrire en C une fonction `void fusion(int n1, int* t1, int n2, int* t2, int* t)` qui fusionne les deux tableaux `t1` et `t2`, supposés triés dans le tableau `t`. Les tableaux `t1` et `t2` sont respectivement de taille `n1` et `n2` et on suppose que la place allouée pour `t` est au moins de taille `n1+n2`.
2. Écrire en C l'algorithme de "tri fusion" `void triFusion(int n, int* tab, int* tabT)` en utilisant la fonction `fusion` précédente. Le tableau à trier est à l'adresse `tab` ayant `n` cases et `tabT` est un tableau temporaire supposé déjà réservé (qui ne chevauche pas `tab`) de taille `n` que l'on peut utiliser.
3. Quelle est la complexité de l'algorithme en temps ? en espace ?