# TD7. Structure de données

Bases de la Programmation Impérative – Chapitre 2 : iterations
version du 7 octobre 2022

Ce TD à pour objectif d'illustrer l'importance du choix d'une structure de données sur la complexité des algorithmes associés.

On cherche ici à analyser un ensemble de données très volumineux. On dispose d'un tableau de plus d'un million de tuples contenant chacun une note et le nom d'un étudiant. Chaque note est un entier entre `0` et `100`.

L'objectif est de répondre aux questions suivantes :

- Comment sélectionner aléatoirement un étudiant ayant une certaine note ?
- Quelle est la note moyenne ?
- Combien d'étudiants ont au moins une certaine note ?
- Qui est le `nième` étudiant en partant du plus faible ?
- Quelle est la note médiane ?

## Exercice 1 : conversion des données

Afin de pouvoir répondre à toutes ces questions de manière efficace, on se propose de stocker les données dans un tableau dynamique `donnees`. `donnees` est indicé entre `0` et `100` (donc de longueur 101) et chaque case `donnees[note]` contient elle-même un tableau de 2 cases :

- `donnees[note][0]` est le nombre total d'étudiants ayant une note strictement inférieure à `note` ;
- `donnees[note][1]` est un tableau dynamique des noms des étudiants dont la note est exactement `note`, trié par ordre alphabétique.

## Question 1

> Représenter `donnees` de façon schématique sur papier puis écrire une fonction `conversion` qui convertit le grand tableau d'origine en `donnees`.

## Question 2

> Quelle est la complexité temporelle de l'algorithme de conversion ?

## Exercice 2 : analyse

Maintenant que notre structure de données est construite, nous pouvons réaliser les algorithmes d'analyse qui la parcourent. Nous allons donc implémenter les opérations décrites dans l'introduction du TD en utilisant `donnees`. Pour chacune des analyses, nous indiquerons quelle est la **complexité temporelle** et quelle est la **complexité spatiale** dans le pire cas.

## Question 1

> Écrire une fonction qui sélectionne aléatoirement un étudiant ayant une certaine note.

## Question 2

Écrire une fonction qui calcule la note moyenne.

## Question 3

Écrire une fonction qui calcule combien d'étudiants ont au moins une certaine note.

## Question 4

Écrire une fonction qui récupère le `nième` étudiant en partant du plus faible.

## Question 5

Écrire une fonction qui calcule la note médiane. Il faut noter que celle-ci n'est pas nécessairement unique si le nombre d'étudiants est pair. En effet la médiane est définie comme un nombre tel que la moitié de la population est en-dessous au sens large (inférieur ou égal) et la moitié de la population est au-dessus au sens large. Si le nombre d'étudiants est pair, il peut s'agir de n'importe quel nombre situé entre les notes des deux étudiants du milieu. Nous utiliserons la *plus petite* médiane.

# Exercice 3 : analyse++ (pour aller plus loin)

## Question 1

Écrire une fonction qui calcule la note médiane en choisissant la *médiane moyenne* plutôt que la plus petite médiane si le nombre d'étudiants est pair.

## Question 2

Modifier la fonction qui calcule la note moyenne afin de minimiser le nombre de multiplications.