# Informatique tronc commun
# Devoir n° 03

À rendre le 29 février 2016

Ce devoir est à réaliser *individuellement* pendant les vacances d'hiver. Les points suivants seront particulièrement évalués.

— Clarté des programmes : vous commenterez vos fonctions. Chacune commencera par une *docstring* et chaque création de variable sera accompagnée d'un commentaire.

— Correction des programmes : vous justifierez la correction de vos fonctions. Chaque boucle sera accompagnée d'un invariant (et d'un variant pour les boucles `while`). Vous pourrez écrire ces invariants hors du corps de la fonction, pour plus de lisibilité. Vous démontrerez la correction de chaque invariant (ou variant), puis que chaque fonction donne le résultat attendu.

— Efficacité des programmes : on veillera à donner des programmes utilisant le moins d'opérations possible. Notamment, on veillera à obtenir la meilleur complexité temporelle asymptotique possible. Garder une complexité spatiale (occupation de la mémoire vive par les objets créés) raisonnable sera vivement apprécié, mais ce n'est pas l'objet du devoir.

Vous écrirez les fonctions demandées dans le langage Python (version 3) et rendrez une version manuscrite [1] de votre travail. Veillez à bien organiser l'écriture vos fonctions, notamment au niveau de l'indentation. Par souci de clarté, il est vivement recommandé de numéroter les lignes d'une fonction Python et de se référencer à un numéro de ligne quand vous écrivez une justification.

Vous rédigerez les justifications demandées hors du corps des fonctions.

## 1 Addition de matrices creuses

Un enjeu scientifique et technologique actuel est de savoir traiter des problèmes mettant en jeu un nombre très important de données. Un point crucial est souvent de pouvoir manipuler des matrices de très grandes dimensions, ce qui est *a priori* très coûteux, en

1. Bien entendu, il est impératif de faire tourner vos fonctions sur ordinateur, ne serait-ce que pour vérifier expérimentalement vos réponses.

temps de calcul et en mémoire. On peut cependant souvent considérer que les matrices manipulées ne contiennent que « peu » d'éléments non nuls : c'est ce que l'on appelle les matrices *creuses*.

Nous nous intéressons ici à l'implémentation de deux algorithmes d'addition de matrices : l'un pour une représentation classique des matrices, l'autre pour une représentation des matrices creuses.

Soit $n \in \mathbb{N}^*$, on note $\mathscr{M}_n(\mathbb{R})$ l'ensemble des matrices carrées d'ordre $n$, à coefficients dans $\mathbb{R}$.

On représentera classiquement une matrice $M \in \mathscr{M}_n(\mathbb{R})$ par un tableau à double entrées. En `Python`, cela sera un tableau (type `list`) de longueur $n$, chaque élément de ce tableau représentant une ligne de $M$. Chaque élément de ce tableau est donc un tableau de longeur $n$, dont tous les éléments sont des nombres (types `int` ou `float`).

Cette même matrice $M$ sera représentée de manière creuse en ne décrivant que ses cases non vides par un tableau de triplets `(i,j,x)`, où `x` est l'élément de $M$ situé sur la $i^{\text{e}}$ ligne et la $j^{\text{e}}$ colonne. On pourra supposer que les éléments non nuls de $M$ sont ainsi décrits ligne par ligne.

**Exemple 1.0.1.** La matrice $\begin{pmatrix} 1 & 0 & 5 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ sera représentée classiquement par le tableau

```
[ [1,0,5] , [0,-2,0] , [0,0,0] ]
```

et de manière creuse par le tableau

```
[ (0,0,1) , (0,2,5) , (1,1,-2) ].
```

**Q1** Écrire une fonction `add(M,N)` prenant en argument deux représentations classiques `M` et `N` de deux matrices $M$ et $N$ (carrées, de même ordre) et renvoyant la représentation classique de $M + N$. On prendra soin d'écrire une fonction « optimale » en terme de complexité, spatiale et temporelle.

**Q2** Écrire une fonction `add_creuse(M,N)` prenant en argument deux représentations creuses `M` et `N` de deux matrices $M$ et $N$ (carrées, de même ordre) et renvoyant la représentation creuse de $M + N$. On prendra soin d'écrire une fonction « optimale » en terme de complexité, spatiale et temporelle.

Pour simplifier, on ne justifiera pas que les éléments obtenus sont disposés dans le bon ordre.

**Q3** On suppose que $M$ et $N$ sont carrées, d'ordre $n$, représentées classiquement par `M` et `N`. Évaluer asymptotiquement la complexité temporelle de la fonction `add(M,N)`.

**Q4** On suppose que $M$ et $N$ sont carrées et contient chacune au plus $p$ élements non nuls, représentées de manière creuse par `M` et `N`. Évaluer asymptotiquement la complexité temporelle de la fonction `add_creuse(M,N)`.

**Q5** Discuter du choix de la représentation pertinente à utiliser pour additionner deux matrices.