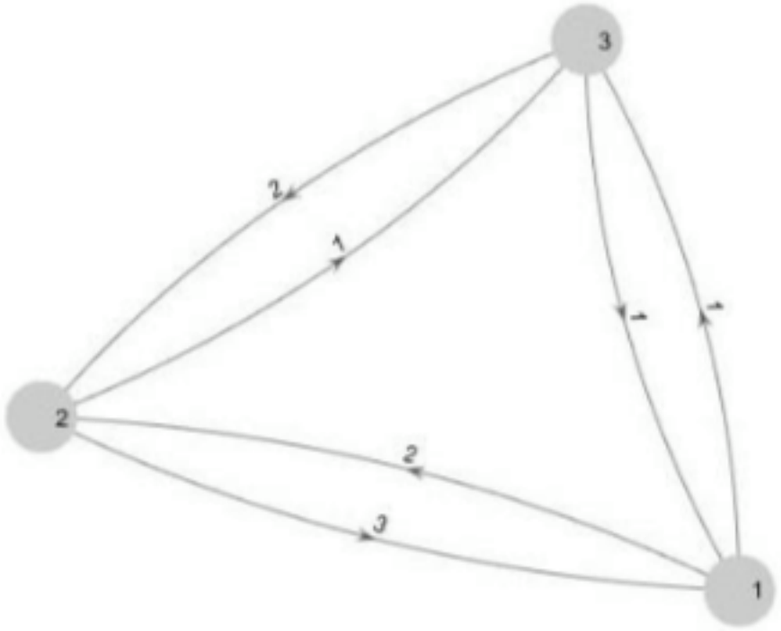
Dans cette question, nous allons vous demander d'implémenter l'algorithme PageRank « de base ». Cet algorithme a été introduit avec la venue du moteur de recherche Google pour classer les pages web. Le PageRank permet de calculer la « popularité » ou le « score » d'une page web à l'aide des liens qu'elle possède avec les autres pages web. C'est en partie grâce à cet algorithme que Google peut fournir à un utilisateur une liste de sites web classés (par ordre décroissant) selon leur popularité PageRank répondant aux critères de sa recherche : les sites obtenant les scores PageRank les plus élevés sont présentés en premier.

Avant d'aller plus loin, nous allons d'abord introduire certains concepts de théorie des graphes qui vous seront nécessaires pour implémenter l'algorithme. Afin de modéliser les pages web et les liens qu'elles possèdent entre elles, vous aurez besoin de la notion de graphe. Un graphe est un ensemble de n nœuds qui sont reliés entre eux par des (hyper)liens. A chacun de ces liens, on peut associer un poids non-négatif qui permet de représenter le degré d'affinité entre les deux nœuds de part et d'autre du lien. Les nœuds du graphe (représentant les page web) sont numérotés séquentiellement (nœud 1, 2, …, n). Voici un exemple de graphe à trois nœuds.



$$\mathbf{A} = \begin{bmatrix} 0 & 2 & 1 \\ 3 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 0 & 2/3 & 1/3 \\ 3/4 & 0 & 1/4 \\ 1/3 & 2/3 & 0 \end{bmatrix}$$

$$d_{in} = \begin{bmatrix} 4 \\ 4 \\ 2 \end{bmatrix} \qquad d_{out} = \begin{bmatrix} 3 \\ 4 \\ 3 \end{bmatrix}$$

Ce graphe peut être représenté sous forme matricielle à l'aide d'une *matrice d'adjacence* **A**, de taille n x n, où l'élément a_{ij} sera égal au degré d'affinité (par exemple le nombre d'hyperliens présents sur la page i et pointant vers la page j) entre le nœud numéro i et le nœud numéro j si ce lien existe, ou 0 si ce lien n'existe pas (0 hyperliens). Dans notre exemple, la page 2 possède trois liens vers la page 1. Sur base de cette matrice, nous pouvons définir le vecteur de degré *entrant* $\mathbf{d}_{in}$ comme la somme des éléments de chaque colonne de la matrice **A**. Les éléments de ce vecteur représentent le nombre de liens qui entrent dans chaque nœud. De manière analogue, nous pouvons définir le vecteur de degré *sortant* $\mathbf{d}_{out}$ comme la somme des éléments de chaque ligne de la matrice **A.** Les éléments de ce vecteur représentent le nombre de liens qui sortent de chaque nœud. De plus, en divisant chaque ligne de la matrice d'adjacence **A** par la ligne correspondante du vecteur de degré sortant $\mathbf{d}_{out}$, vous obtiendrez la

matrice de *probabilités de transition* **P** dont chaque élément p_{ij} correspond à la probabilité de passer du nœud *i* au nœud *j* lorsqu'on « surfe » sur le web et que l'on choisit la prochaine page à consulter au hasard parmi les liens disponibles sur cette page.

Afin d'obtenir les scores PageRank contenus dans le tableau à une dimension **score**, vous devez utiliser un algorithme itératif nommé « power method ». Cet algorithme consiste à répéter l'équation ci-dessous jusqu'à convergence du vecteur de scores, en d'autres termes jusqu'à ce que les scores à l'itération *t* soient (approximativement) les même que les scores à l'itération *t*–1.

$$\mathbf{score}(t) = \mathbf{P}^{\mathrm{T}} * \mathbf{score}(t-1)$$

où $*$ est la multiplication matrice-vecteur. Cette équation consiste à simplement multiplier la transposée (T) de la matrice de probabilités de transition $\mathbf{P}^{\mathrm{T}}$ avec le vecteur de scores obtenu à l'itération précédente. Pour rappel, la transposée de la matrice **P** est la matrice $\mathbf{P}^{\mathrm{T}}$ dont les lignes contiennent les colonnes de **P**. Afin d'initialiser l'algorithme à l'itération $t = 0$, vous allez devoir utiliser le vecteur de degré entrant $\mathbf{d}_{in}$ *normalisé* (la somme de ses éléments doit être égale à un) comme premier score, comme vous pouvez le voir dans l'exemple ci-dessous :

Itération *t* = 1

$$\begin{bmatrix} 0.3667 \\ 0.4000 \\ 0.2333 \end{bmatrix} = \begin{bmatrix} 0 & 3/4 & 1/3 \\ 2/3 & 0 & 2/3 \\ 1/3 & 1/4 & 0 \end{bmatrix} * \begin{bmatrix} 4/10 \\ 4/10 \\ 2/10 \end{bmatrix}$$

Itération *t* = 2

$$\begin{bmatrix} 0.3741 \\ 0.4000 \\ 0.2259 \end{bmatrix} = \begin{bmatrix} 0 & 3/4 & 1/3 \\ 2/3 & 0 & 2/3 \\ 1/3 & 1/4 & 0 \end{bmatrix} * \begin{bmatrix} 0.3667 \\ 0.4000 \\ 0.2333 \end{bmatrix}$$

Cette méthode itérative doit être répétée et converge vers des scores stables qui représentent la popularité de chaque page : on peut montrer qu'une page web aura une popularité élevée si elle est référencée par de nombreuses pages qui sont elle-même populaires. Nous vous demandons d'implémenter les méthodes suivantes sur base de la matrice d'adjacence **A** :

Signature de la méthode 1 :

```
public static double[][] ProbabilityMatrix(int A[][])
```

Cette méthode permet de calculer la matrice de probabilités de transition sur base de la matrice d'adjacence et du vecteur de degré sortant.