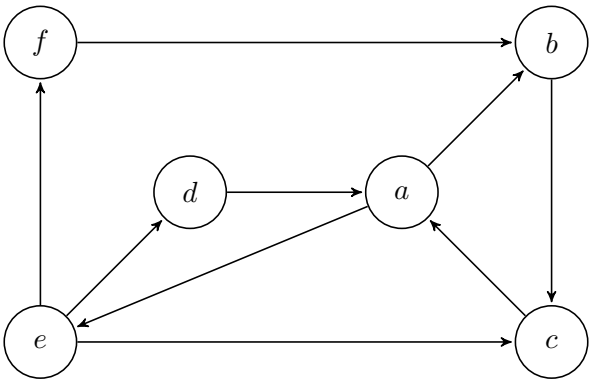
On s'intéresse ici à l'algorithme PageRank qui a fait le succès du moteur de recherche de Google.
Pour se faire, on modélise le web par une collection de N pages dont certaines incluent des liens hypertextes vers d'autres pages. On pense que dans la réalité $N = 10^9$ est un bon ordre de grandeur mais on se contentera ici de $N = 6$ (les pages étant nommées a, b, c, d, e et f). Voici le graphe qui décrit les liens hypertexte pointant entre les différentes pages .



L'objectif est de donner à chaque page un score qui mesure sa popularité.
Pour cela, on considère un surfeur allant de page en page, cliquant aléatoirement sur les liens hypertexte qu'il rencontre. (ainsi, s'il se trouve à l'état initial sur la page a, il a une probabilité de $1/2$ de se retrouver en b et une probabilité de $1/2$ de se retrouver en d à l'étape suivante). Le score d'une page sera alors la proportion de visites sur cette page après un nombre significatif de clics du surfeur.

A. Un peu de manipulation...

1. Simuler une marche aléatoire sur le graphe. Estimer le score de chaque page. Le résultat dépend-il de la page de départ ?

On note maintenant a_n (respectivement b_n, c_n, ...) la probabilité que le surfeur se trouve sur la page a (respectivement b, c, *etc.*) au n-ème clic. On note également $V_n = \begin{pmatrix} a_n \\ b_n \\ \vdots \\ f_n \end{pmatrix}$.

Nb : on dira que $W = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$ est un **vecteur de probabilités** s'il vérifie $\forall k$, $x_k \geqslant 0$ et $\sum_k x_k = 1$.

2. En utilisant la formule des probabilités totales, exprimer a_{n+1} en fonction de $(a_n, b_n, c_n, d_n, e_n, f_n)$.
3. Donner une matrice M de $\mathcal{M}_6(\mathbb{R})$ telle que, pour tout $n \in \mathbb{N}$, $V_{n+1} = MV_n$.
4. Que peut-on dire de la somme des coefficients de chaque colonne de M ? Une telle matrice est appelée **matrice stochastique**.
5. Montrer par récurrence que pour tout $n \in \mathbb{N}$, $V_n = M^n V_0$.
6. A l'aide d'une calculatrice ou d'un logiciel adapté, calculer le vecteur V_n pour différentes valeurs de n et de V_0. Que peut-on conjecturer ?
7. Bien que la notion de limite de vecteur n'ait pas été définie, on suppose ici que la suite (V_n) **converge vers un vecteur** L. En remarquant que $M^{n+1}V_0 = MM^nV_0$, montrer que L est vecteur propre de M. Donner la valeur propre associée.

B. Un peu de théorie...

Soit M une matrice stochastique de $\mathcal{M}_N(\mathbb{R})$, l'objectif de cette partie est de montrer que M admet un vecteur propre $L = {}^t\begin{pmatrix}\ell_1 & \ell_2 & \dots & \ell_N\end{pmatrix}$ associé à la valeur propre 1 qui soit un vecteur de probabilités.

8. Avant de prouver son existence, montrer que si un tel vecteur L existe, alors il est bien limite de M^nX pour un certain vecteur de probabilité X à préciser. Est-ce le cas, à priori pour un vecteur de probabilité X quelconque ?
9. Calculer $\begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix} \times M$. En déduire que le vecteur ${}^t\begin{pmatrix}1 & \dots & 1\end{pmatrix}$ est vecteur propre de ${}^tM$ associé à la valeur propre 1. Montrer alors que 1 est également valeur propre de M (on pourra montrer que $M - I_N$ n'est pas inversible).
10. Soit $L = {}^t\begin{pmatrix}\ell_1 & \ell_2 & \dots & \ell_N\end{pmatrix}$ *un* vecteur propre de M associé à la valeur propre 1. On note $L' = {}^t\begin{pmatrix}|\ell_1| & |\ell_2| & \dots & |\ell_N|\end{pmatrix}$. Montrer que le vecteur $ML' - L'$ n'a que des composantes positives (on pensera à utiliser l'inégalité triangulaire).
11. En sommant les coordonnées de $ML' - L'$, montrer que L' est vecteur propre de M.
12. Conclure sur cette partie.

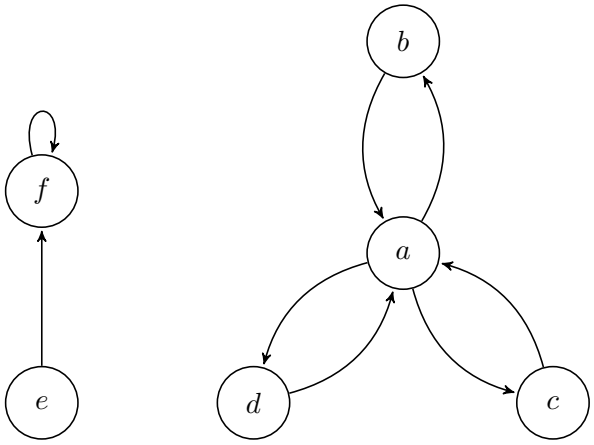
C. Limitations...

Nous venons de voir qu'une matrice stochastique admet toujours un vecteur propre pour la valeur propre 1. Dans la première partie, nous l'avions « trouvé » comme limite de la suite de vecteurs (M^nV_0)... Le calcul itéré de M^nV_0 présente cependant un certain nombre d'inconvénients :

- Rien n'assure la convergence de M^nV_0.
- Si le graphe est en plusieurs parties non reliées entre elles (on dit qu'il est **non-connexe**) alors la distribution de probabilité initiale peut influer sur le résultat.

Notez qu'on peut parfois aussi s'échapper d'une partie du graphe sans y revenir. Les nœuds de cette partie auront alors une probabilité nulle dans le vecteur propre L.

13. Mettre en évidence ces limitations sur le graphe suivant. Quel est la dimension du sous-espace propre associé à la valeur propre 1 ? :



Tous ces problèmes peuvent contournés en remplaçant la matrice M par la matrice

$$M' = \delta M + \frac{1-\delta}{N}\begin{pmatrix}1 & \dots & 1\\ \vdots & & \vdots\\ 1 & \dots & 1\end{pmatrix}$$

qui revient à introduire la probabilité $1-\delta$ de « sauter » sur une page prise au hasard au lieu d'emprunter les liens hypertextes.

Un résultat mathématique (le théorème de **Perron-Frobenius** †) affirme alors que le sous espace propre associé à la valeur propre 1 est maintenant de dimension 1.

14. Simuler ce nouvel algorithme sur le graphe précédent pour $\delta = 0.85$ (valeur habituellement utilisée). On commencera déjà par montrer que M' est encore stochastique.

†. théorème de **Perron-Frobenius** : `https://ljk.imag.fr/membres/Bernard.Ycart/mel/re/node17.html`

15. Que peut-on dire de l'influence de δ sur la vitesse de convergence ?
16. Que peut-on dire du choix de δ sur la précision du résultat final ?

D. Programme utile

```
import numpy as np
import numpy.linalg as alg
import random

A=np.array([ # Matrice stochastique déduite du graphe
    [0   ,  0,  1,  1, 0       , 0],
    [0.5 ,  0,  0,  0, 0       , 1],
    [0   ,  1,  0,  0, 0.33333 , 0],
    [0   ,  0,  0,  0, 0.33333 , 0],
    [0.5 ,  0,  0,  0, 0       , 0],
    [0   ,  0,  0,  0, 0.33333 , 0]])

def clic(Sto,npage):
    '''Simule un clic aleatoire sur une page
    et renvoie le numero de la page cliquee
    Sto=matrice stochastique du graphe
    npage=page initiale
    '''
    r=random.randint(0,9999)
    t=0
    for i in range(len(Sto)):
        t=t+Sto[i][npage]
        if r<=t*10000 :
            return i

def surf(Sto,npage,nbclics):
    '''Simule un surf sur le web
    et retourne un vecteur contenant
    les frequences des visites de chaque page.
    Sto = matrice stochastique du graphe
    npage=page initiale
    nbclics = nombre de fois ou l'on clique sur un lien'''
    V=np.array([0.0,0.0,0.0,0.0,0.0,0.0])
    for i in range(nbclics):
        npage=clic(Sto,npage)
        V[npage]=V[npage]+1.0
    return V/nbclics

print(surf(A,4,5)) # Test de la fonction surf
V=np.array([0,0,0,0,0,1]) # Creation d'un vecteur de probabilites
V=np.transpose(V) # Transposition du meme vecteur
Mn=alg.matrix_power(A,5) # Mise a la puissance de A
print(np.dot(Mn,V)) # Produit de A par V
```