L’algorithme PageRank et les chaînes de Markov
ou les Mathématiques expertes de Terminale en prolongement des activités sur le WEB en SNT de seconde

Éléments du programme de Mathématiques expertes auxquels fait référence ce document :

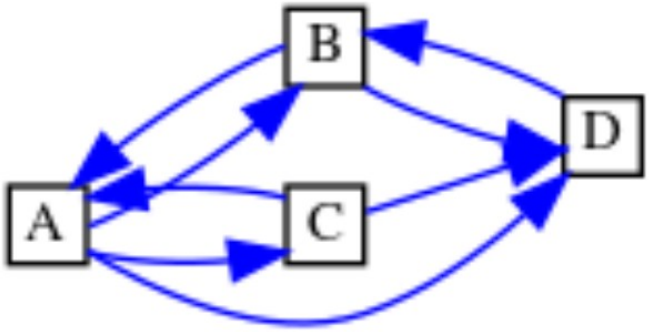
Contenus	- Graphe orienté pondéré associé à une chaîne de Markov à deux ou trois états. - Chaîne de Markov à deux ou trois états. Distribution initiale, représentée par une matrice ligne π_0. Matrice de transition, graphe pondéré associé. - Pour une chaîne de Markov à deux ou trois états de matrice P, interprétation du coefficient (i,j) de P_n. Distribution après n transitions, représentée comme la matrice ligne $\pi_0 P^n$. - Distributions invariantes d’une chaîne de Markov à deux ou trois états.
Capacités attendues	Dans le cadre de la résolution de problèmes, utiliser le calcul matriciel, notamment l’inverse et les puissances d’une matrice carrée, pour résoudre un système linéaire, étudier une suite récurrente linéaire, calculer le nombre de chemins de longueur donnée entre deux sommets d’un graphe, étudier une chaîne de Markov à deux ou trois états (calculer des probabilités, déterminer une probabilité invariante).
Démonstration	Pour une chaîne de Markov, expression de la probabilité de passer de l’état i à l'état j en n transitions, de la matrice ligne représentant la distribution après n transitions.

Première question : qu’est ce que le Web ?
- D’abord : ce n’est pas Internet, qui est un réseau à l’échelle mondiale, permettant de fournir de nombreux services. Parmi ces services, on peut citer :
. Le Web (protocole HTTP) ;
. Les Emails (protocole SMTP);
. La voix sur IP ;
. Le transfert de fichiers (protocole FTP);
. La messagerie instantanée ;
. etc.

Le Web a été inventé en 1989 par Tim Berners-Lee (Britannique) et Robert Caillau (Belge).
Ces deux ingénieurs du CERN ont mis au point le système hypertexte, qui permet, à partir d’un document, de consulter d’autres documents par un clic sur un mot-clé, appelé hyperlien ou lien hypertexte. Le Web est passé dans le domaine public en 1991 et a commencé à être popularisé en 1993.
Le Web est donc un service fourni par le réseau Internet, contenant des milliards de pages HTML (Hyper Text Markup Language), reliées entre elles par des milliards d’hyperliens.

On représentera le Web (ou plutôt une toute petite partie du Web) par un graphe orienté dans lequel :
- un sommet représentera une page web ;
- un arc représentera un hyperlien d’une page web vers une autre.

Exemple :



Deuxième question : Quel sont les rôles d'un moteur de recherche ?

Après que l'utilisateur a exprimé sa requête sous la forme de mots-clés recherchés dans un formulaire de recherche, le moteur de recherche :
- détermine une liste des pages web contenant les mots-clés (ou dont le titre contient les mots-clés) ;
- effectue un tri de cette liste selon leur « pertinence » ;
- affiche la liste triée (tout ceci en quelques millisecondes).

La question sous-jacente est donc : comment définir la pertinence d'une page web ?

On peut penser à plusieurs solutions naturelles :
Solution 1 : pour chaque domaine présent sur le web, demander à un expert (ou à un groupe d'experts) de donner un score à chaque page web traitant du domaine.
Plusieurs difficultés émergent immédiatement :
- il faut un nombre d'experts considérables pour traiter les millions de pages web et il faudra actualiser ces millions de scores régulièrement.
- on peut toujours douter de la neutralité des experts et des manipulations sont possibles.

Solution 2 : demander aux internautes d'attribuer un score à chaque site visité, bref une sorte de vote. Là encore, la neutralité du classement sera immédiatement remise en cause : on peut voter pour soi-même un grand nombre de fois ou payer des gens pour le faire etc.

Ces deux solutions naïves reposent sur des interventions humaines, sujettes à caution, d'où l'idée d'obtenir un classement automatisé basé sur un algorithme simple.

Troisième question : Pourquoi, malgré la présence de moteurs de recherche alternatifs, Google reste-t-il le moteur de recherche plébiscité ?

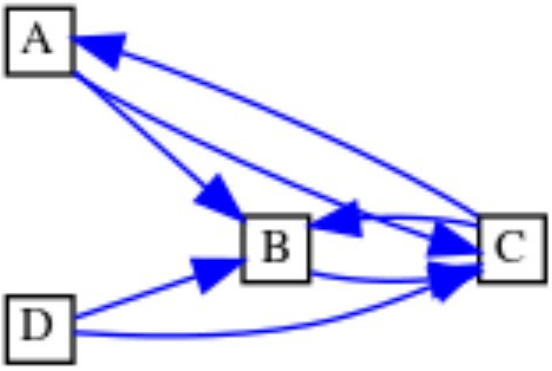
Il semble bien que Google arrive à répondre de manière la plus satisfaisante aux requêtes des utilisateurs… d'où son « presque monopole » des requêtes de recherche.

Les deux fondateurs de Google, Larry Page et Sergey Brin, ont défini un score pour chaque page web issue de la recherche, entre 0 et 1, appelé le PageRank (la somme des PageRanks vaut 1).
Le PageRank est calculé à partir des deux règles, très simples, suivantes :

Règle 1 : Le PageRank (= score) attribué à une page web doit être d'autant plus élevé que celle-ci est référencée par une page faisant autorité (= ayant elle-même un PageRank élevé).

Règle 2 : Le PageRank attribué à une page web doit être d'autant moins élevé que celle-ci est référencée par une page ayant peu de crédit, c'est-à-dire multipliant les liens vers d'autres pages.

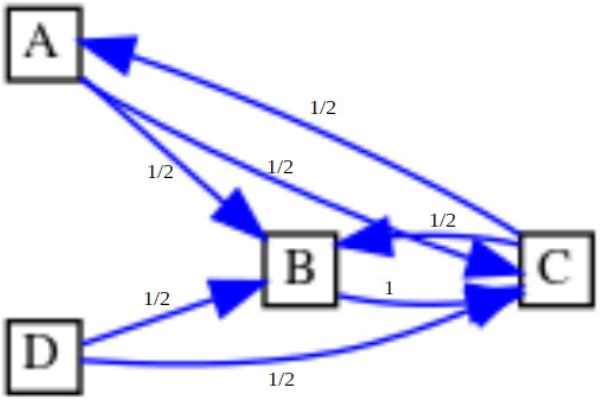
On modélise par un graphe orienté l'ensemble des pages web issues de la requête d'une recherche.
- un sommet représentera une page web ;
- un arc représentera un hyperlien d'une page web vers une autre.



On note a, b, c et d les PageRanks respectifs des pages A, B, C et D.

On va ensuite modéliser les deux règles précédentes de la façon suivante :
- Le PageRank de chaque page sera la somme des PageRanks des pages pointant vers elle (respect de la première règle) ;
- mais les PageRanks de chaque page sont pondérés par l'inverse du nombre de pages qu'elle référence (respect de la deuxième règle).

On aura donc un graphe probabiliste :



On obtient le système suivant, que l'on peut résoudre à la main :

$$\begin{cases} a = \frac{c}{2} \\ b = \frac{a}{2} + \frac{c}{2} + \frac{d}{2} \\ c = \frac{a}{2} + b + \frac{d}{2} \\ d = 0 \\ a + b + c + d = 1 \end{cases}$$

Et on trouve aisément :

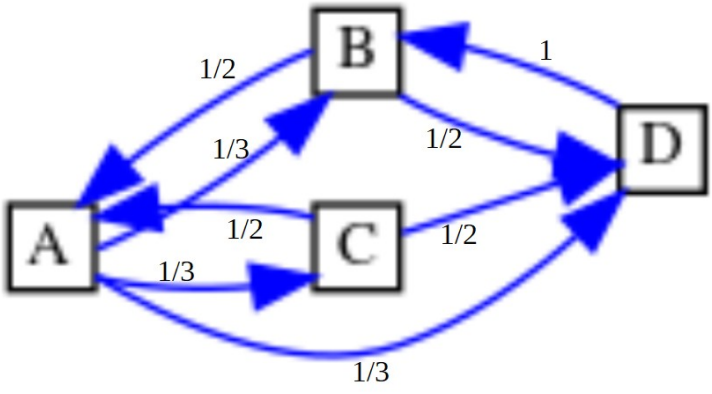
$a = \frac{2}{9}$, $b = \frac{1}{3}$, $c = \frac{4}{9}$, $d = 0$

$a \simeq 0{,}222222222$, $b \simeq 0{,}333333333$, $c \simeq 0{,}444444444$, $d = 0$

Prenons le résultat d'une seconde recherche :

Le système sera :

$$\begin{cases} a = \frac{b}{2} + \frac{c}{2} \\ b = \frac{a}{3} + d \\ c = \frac{a}{3} \\ d = \frac{a}{3} + \frac{b}{2} + \frac{c}{2} \\ a + b + c + d = 1 \end{cases}$$



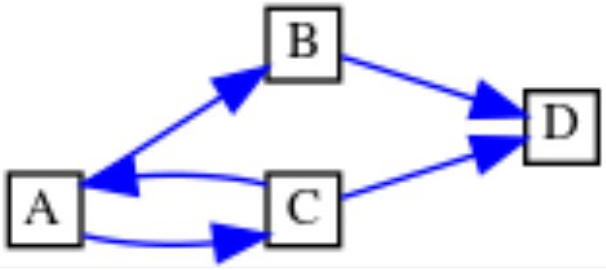
On peut le résoudre avec un logiciel de calcul formel :

```
linsolve([a=b/2+c/2,b=a/3+d,c=a/3,d=a/3+b/2+c/2,a+b+c+d=1],[a,b,c,d])
```

$$\left[\frac{3}{13}, \frac{5}{13}, \frac{1}{13}, \frac{4}{13}\right]$$

$a \simeq 0{,}23076923$, $b \simeq 0{,}38461538$, $c \simeq 0{,}07692307$, $d \simeq 0{,}30769230$

- Un premier problème : **le trou noir**.
Prenons le résultat d'une troisième recherche :




```
linsolve([a=c/2,b=a/2,c=a/2,d=b+c/2,a+b+c+d=1],[a,b,c,d])
```

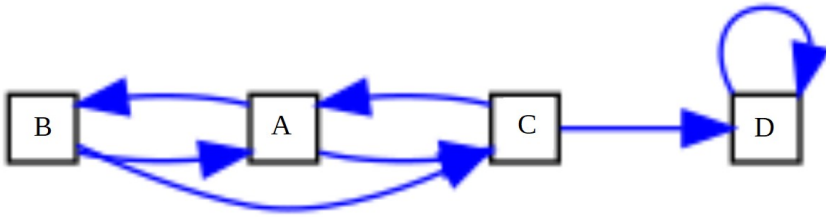
[]

```
linsolve([a=c/2,b=a/2,c=a/2,d=b+c/2],[a,b,c,d])
```

[0, 0, 0, 0]

Là, on ne peut pas dire que notre modèle soit satisfaisant… Pas vraiment de solution !

- Un deuxième problème : **le puits**.
Prenons le résultat d'une quatrième recherche :

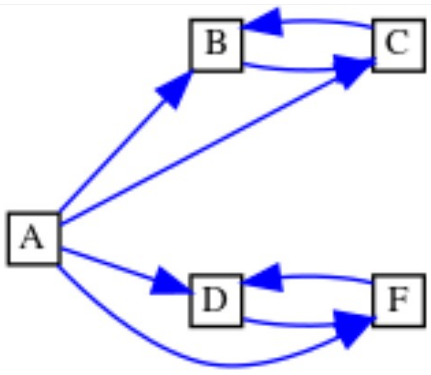


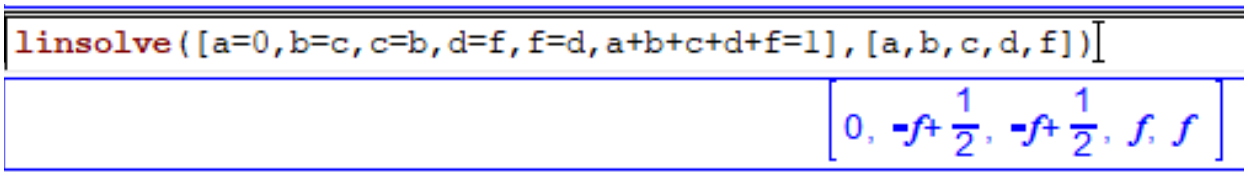

```
linsolve([a=b/2+c/2,b=a/2,c=a/2+b/2,d=c/2+d,a+b+c+d=1],[a,b,c,d])
```

[0, 0, 0, 1]

Ici, il y a bien une solution, mais elle n'est pas satisfaisante « moralement ».

- Un troisième problème : **les poches web**.
Prenons le résultat d'une quatrième recherche :





On peut opter pour les PageRanks suivants :

$a = 0$, $b = 0$, $c = 0$, $d = 0{,}5$, $f = 0{,}5$

ou bien :

$a = 0$, $b = 0{,}5$, $c = 0{,}5$, $d = 0$, $f = 0$

ou bien :

$a = 0$, $b = 0{,}25$, $c = 0{,}25$, $d = 0{,}25$, $f = 0{,}25$

ou bien …

Bref, les trois derniers exemples mettent à mal le système proposé !

L'idée de Larry Page et de Sergey Brin est alors celle du surfeur aléatoire : un surfeur démarre d'une page web quelconque parmi celles des résultats de la recherche.

Il faut cependant régler le problème des pages qui n'en référencent aucune et qui arrêtent le surfeur aléatoire.
Lorsqu'il tombera dans un trou noir, c'est-à-dire vers une page qui n'en référence aucune autre, le surfeur aléatoire reprendra son surf sur une autre page au hasard. Autrement dit, on remplace une ligne de 0 par un renvoi équiprobable vers une page issue de la recherche (sans pour autant exclure la page elle-même).

La question est : avec quelle probabilité le surfeur aléatoire sera sur chacune des pages, après un très grand nombre de clics ?

Le PageRank d'une page web sera alors la probabilité, au bout d'un temps très long, d'être sur cette page.

Focus sur les chaînes de Markov :

1. Les chaînes de Markov (à espace des états discret)

Le principe d'une chaîne de Markov :
« Le futur d'un processus markovien ne dépend que de l'état présent et pas des états passés. »

Ce principe se modélise par une suite de variables aléatoires réelles $(X_n)_{n\in\mathbb{N}}$ prenant leurs valeurs dans un espace E fini (appelé espace des états) possédant la propriété suivante :
Pour tout entier n ≥ 0, pour tout (n+1)-uplets de E $(e_0, e_1, \ldots, e_{n-1}, e_n, e_{n+1}) \in E^{n+1}$, on a :
$P_{X_n=e_n,\, X_{n-1}=e_{n-1},\ldots,X_1=e_1,\, X_0=e_0}(X_{n+1}=e_{n+1}) = P_{X_n=e_n}(X_{n+1}=e_{n+1})$
Un telle suite de variables aléatoires réelles $(X_n)_{n\in\mathbb{N}}$ est dite chaîne de Markov à valeurs dans E.

Remarque : dans cette définition, la probabilité de passer d'un état à un autre dépend à priori de n.

2. Les chaînes de Markov homogènes

Soit une chaîne de Markov (X_n) d'espace des états $E = \{ e_i ; 1 \le i \le N \}$.
On dit que (X_n) est homogène si, pour tout $n \in \mathbb{N}$ et tous $1 \le i, j \le N$,
$P_{X_n = e_i}(X_{n+1} = e_j)$ ne dépend pas de *n*.

Dans la suite du document (ainsi qu'en Maths expertes), toutes les chaînes de Markov seront supposées homogènes.

3. Matrice de transition
Le nombre $P_{X_n = e_i}(X_{n+1} = e_j)$ ne dépendant pas de *n*, on le note Q*(i, j)*.
La matrice $Q = [Q(i,j)]_{1 \le i,j \le N}$ d'ordre *N* est appelée matrice de transition de la chaîne de Markov.
La probabilité que la chaîne de Markov passe de l'état e_i à l'état e_j est ainsi égale au coefficient *(i, j)* de la matrice de transition.

4. Matrice stochastique
On appelle matrice stochastique d'ordre *n* une matrice carrée d'ordre *n* dont la somme des coefficients de chaque ligne est égale à 1.
La matrice de transition d'une chaîne de Markov est une matrice stochastique.

(Preuve intéressante à faire avec la formule des probabilités totales).

5. Distribution initiale
On appelle distribution initiale de la chaîne de Markov la loi de la variable aléatoire X_0 , notée π_0 . Elle est représentée par une matrice ligne :
$\pi_0 = (P(X_0) = e_1 \quad P(X_0) = e_2 \quad \ldots \quad P(X_0) = e_N)$

6. Distribution en *m* étapes et matrice de transition
Soit *m* un entier naturel strictement non nul.
Pour tout $n \in \mathbb{N}$, et tous états e_i et e_j , la probabilité $P_{X_n = e_i}(X_{n+m} = e_j)$ est le coefficient *(i, j)* de la matrice Q^m .

On appelle distributions de la chaîne de Markov (X_n) les matrices lignes donnant la loi des variables aléatoires X_n pour $n \in \mathbb{N}$.
On note : $\pi_n = (P(X_n) = e_1 \quad P(X_n) = e_2 \quad \ldots \quad P(X_n) = e_N)$

Calcul de distribution :
Les distributions de la chaîne de Markov (X_n) vérifient la relation de récurrence $\pi_{n+1} = \pi_n * Q$ pour tout $n \in \mathbb{N}$.
On a alors : $\pi_n = \pi_0 * Q^n$ pour tout $n \in \mathbb{N}$.

7. Représentation d'une chaîne de Markov
Une chaîne de Markov peut être représentée par :
- une matrice (on vient de le détailler) ;
- un graphe orienté tel que :
  . les sommets sont les états possibles de la chaîne ;
  . deux sommets e_i et e_j sont reliés par un arc, étiqueté avec le coefficient *(i, j)*, allant de e_i vers e_j quand la chaîne peut passer de l'état e_i à l'état e_j en une seule étape.

8. Convergence d'une chaîne de Markov

Loi stationnaire :
Une loi de probabilité π vérifiant $\pi = \pi * Q$ est appelée loi stationnaire de Q.

Théorème de Perron-Froebenius :
- Une chaîne de Markov est irréductible si pour tout couple (e_i , e_j) avec $i \neq j$ de sommets du graphe, il existe un chemin de *i* vers *j* et un chemin de *j* vers *i*.
  Autrement dit : - si la matrice de transition Q ne possède aucun coefficient nul, excepté sur sa diagonale principale ;
  - si tout état est accessible à partir de n'importe quel autre état.
- Une chaîne de Markov irréductible à état fini possède exactement une loi de probabilité stationnaire.
- La suite de matrices $(Q^n)_{n\geq 1}$ converge vers une matrice limite dont toutes les lignes sont égales à la loi stationnaire.

Revenons à notre surfeur aléatoire :

La question était : avec quelle probabilité le surfeur aléatoire sera sur chacune des pages, après un très grand nombre de clics ?

Le PageRank d'une page sera alors la probabilité, au bout d'un temps très long, d'être sur cette page.

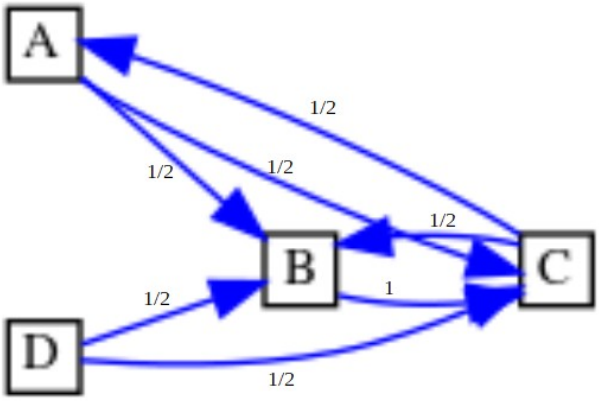
On obtient donc une chaîne de Markov (X_n) avec une distribution initiale π_0 qui sera du type (1,0,0,0) ou (0,1,0,0) etc. , une matrice de transition P et pour tout entier naturel *n* :
$\pi_{n+1} = \pi_n * P$ et $\pi_n = \pi_0 * P^n$

L'ensemble des PageRanks cherchés sera alors une distribution invariante (ou un état stable ou loi stationnaire) de la chaîne de Markov.

A ce stade, on n'est pas assuré de :
- l'existence d'un état stable,
- l'unicité de l'état stable (cf l'exemple sur les poches web),
- ni de sa pertinence pour effectuer un tri des pages web retournées lors de la recherche (cf le puits).

Nous allons reprendre nos 5 exemples et construire les matrices d'adjacences des graphes probabilistes, ou encore les matrices de transition associées à la chaîne de Markov.

Le premier exemple :



Matrice de transition :

$$P = \begin{pmatrix} 0 & 0,5 & 0,5 & 0 \\ 0 & 0 & 1 & 0 \\ 0,5 & 0,5 & 0 & 0 \\ 0 & 0,5 & 0,5 & 0 \end{pmatrix}$$

```
matrix(4,4,[[0,0.5,0.5,0],[0,0,1,0],[0.5,0.5,0,0],[0,0.5,0.5,0]])
```

$$\begin{bmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix}$$

```
matrix(4,4,[[0,0.5,0.5,0],[0,0,1,0],[0.5,0.5,0,0],[0,0.5,0.5,0]])**100
```

$$\begin{bmatrix} 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \\ 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \\ 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \\ 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \end{bmatrix}$$

Deux remarques :
- L'état stable correspond bien aux PageRanks trouvés au début.
- La distribution après un grand nombre d'étapes est la même quelle que soit la distribution π_0 initiale :

```
M:=matrix(1,4,[1,0,0,0])
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

```
N:=matrix(4,4,[[0,0.5,0.5,0],[0,0,1,0],[0.5,0.5,0,0],[0,0.5,0.5,0]])**100
```

$$\begin{bmatrix} 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \\ 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \\ 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \\ 0.222222222222 & 0.333333333333 & 0.444444444444 & 0.0 \end{bmatrix}$$

```
Q:= matrix(1,4,[0,1,0,0])
```

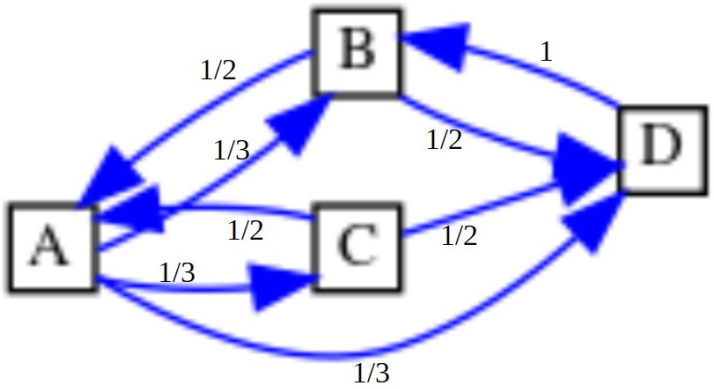
$$\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$$

```
M*N==Q*N
```

vrai

Même le logiciel de calcul formel nous le dit !?!

Le deuxième exemple :



Matrice de transition :

$$P = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0,5 & 0 & 0 & 0,5 \\ 0,5 & 0 & 0 & 0,5 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

```
matrix(4,4,[[0,1/3,1/3,1/3],[0.5,0,0,0.5],[0.5,0,0,0.5],[0,1,0,0]])
```

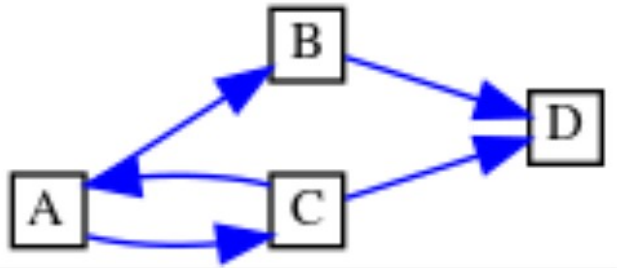
$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

```
matrix(4,4,[[0,1/3,1/3,1/3],[0.5,0,0,0.5],[0.5,0,0,0.5],[0,1,0,0]])**100
```

$$\begin{bmatrix} 0.230769230775 & 0.384615384609 & 0.0769230769206 & 0.307692307696 \\ 0.23076923075 & 0.384615384637 & 0.076923076931 & 0.307692307681 \\ 0.23076923075 & 0.384615384637 & 0.076923076931 & 0.307692307681 \\ 0.230769230793 & 0.384615384588 & 0.076923076913 & 0.307692307706 \end{bmatrix}$$

Là encore, RAS.

Le troisième exemple (celui avec **le trou noir**) :



Matrice de transition :

$$P = \begin{pmatrix} 0 & 0{,}5 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \\ 0{,}5 & 0 & 0 & 0{,}5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

remplacée par :

$$P = \begin{pmatrix} 0 & 0{,}5 & 0{,}5 & 0 \\ 0 & 0 & 0 & 1 \\ 0{,}5 & 0 & 0 & 0{,}5 \\ 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \end{pmatrix}$$

```
matrix(4,4,[[0,0.5,0.5,0],[0,0,0,1],[0.5,0,0,0.5],[0.25,0.25,0.25,0.25]])
```

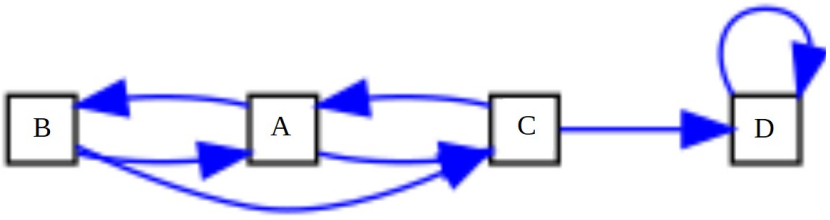
$$\begin{pmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \\ 0.5 & 0 & 0 & 0.5 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

```
matrix(4,4,[[0,0.5,0.5,0],[0,0,0,1],[0.5,0,0,0.5],[0.25,0.25,0.25,0.25]])**100
```

$$\begin{pmatrix} 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.4 \end{pmatrix}$$

Voilà un premier problème qui semble réglé… nous y reviendrons.

Le quatrième exemple (celui avec **le puits**) :



Matrice de transition :

$$P = \begin{pmatrix} 0 & 0{,}5 & 0{,}5 & 0 \\ 0{,}5 & 0 & 0{,}5 & 0 \\ 0{,}5 & 0 & 0 & 0{,}5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
matrix(4,4,[[0,0.5,0.5,0],[0.5,0,0.5,0],[0.5,0,0,0.5],[0,0,0,1]])
```

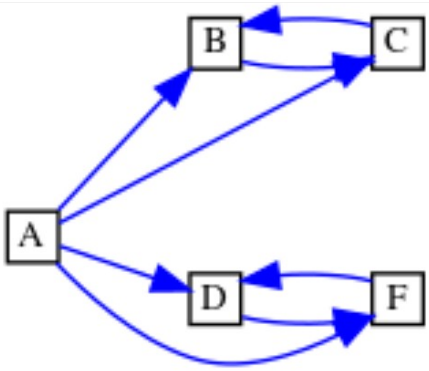
$$\begin{pmatrix} 0 & 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
matrix(4,4,[[0,0.5,0.5,0],[0.5,0,0.5,0],[0.5,0,0,0.5],[0,0,0,1]])**100
```

$$\begin{pmatrix} 2.79434134386\text{e-}10 & 1.72699792668\text{e-}10 & 2.79434134386\text{e-}10 & 0.999999999268 \\ 2.79434134386\text{e-}10 & 1.72699792668\text{e-}10 & 2.79434134386\text{e-}10 & 0.999999999268 \\ 1.72699792668\text{e-}10 & 1.06734341719\text{e-}10 & 1.72699792668\text{e-}10 & 0.999999999548 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

Le résultat est sans surprise mais toujours assez inintéressant… Bref, on n'a pas encore de PageRank exploitable dans ce cas-là.

Le cinquième problème (celui avec **les poches du web**) :



Matrice de transition :

$$P = \begin{pmatrix} 0 & 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

```
matrix(5,5,[[0,0.25,0.25,0.25,0.25],[0,0,1,0,0],[0,1,0,0,0],[0,0,0,0,1],[0,0,0,1,0]])
```

0	0.25	0.25	0.25	0.25
0	0	1	0	0
0	1	0	0	0
0	0	0	0	1
0	0	0	1	0

```
matrix(5,5,[[0,0.25,0.25,0.25,0.25],[0,0,1,0,0],[0,1,0,0,0],[0,0,0,0,1],[0,0,0,1,0]])**2
```

0.0	0.25	0.25	0.25	0.25
0.0	1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	1.0

```
matrix(5,5,[[0,0.25,0.25,0.25,0.25],[0,0,1,0,0],[0,1,0,0,0],[0,0,0,0,1],[0,0,0,1,0]])**3
```

0.0	0.25	0.25	0.25	0.25
0.0	0.0	1.0	0.0	0.0
0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0
0.0	0.0	0.0	1.0	0.0

Là, il y a toujours un problème. En effet, il apparaît que $P^3 = P$ et donc, pour tout entier naturel non nul, on aura :

- $P^{2n} = P^2$
- $P^{2n+1} = P$

Ici, la suite des puissances de P n'est pas convergente…
Bref, le problème du puits et des poches du web n'est pas résolu.

Mais l'idée maîtresse de Larry Page et Sergey Brin est d'entrer absolument dans le cadre d'application du **théorème de Perron-Froebenius**, qui dit, en simplifiant, que :
- Si la matrice de transition P ne contient aucun 0 (sauf sur la diagonale), alors la chaîne de Markov admet une unique distribution invariante (un unique état stable).
- La suite des matrices $(P^n)_{n\geq 1}$, converge vers une matrice limite dont les coefficients de chaque ligne sont la distribution invariante.

Ainsi, à chaque clic sur un hyperlien par le surfeur aléatoire (i.e. à chaque étape) :
- on poursuit le surf aléatoire avec une probabilité de 0,85 ;
- on fait un saut aléatoire (vers n'importe quelle page issue de la recherche) avec une probabilité de 0,15. Si *n* est le nombre de pages de la réponse à la recherche, la probabilité de tomber sur une certaine page est bien sûr $\frac{1}{n}$.

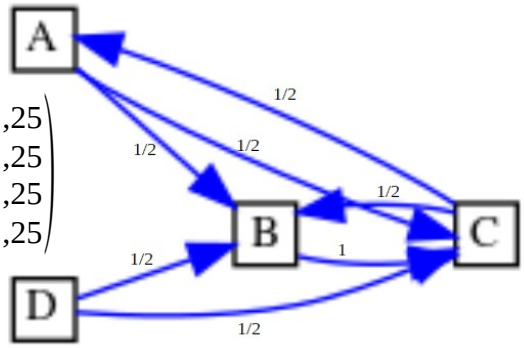
La matrice de transition P sera donc remplacée désormais par :

$$Q = 0{,}85*P + 0{,}15*[\ \frac{1}{n}\ ]_{1 \leq i,j \leq n}$$

Reprenons une dernière fois nos cinq cas d'école :

Cas n°1 :

$$Q = 0{,}85 * \begin{pmatrix} 0 & 0{,}5 & 0{,}5 & 0 \\ 0 & 0 & 1 & 0 \\ 0{,}5 & 0{,}5 & 0 & 0 \\ 0 & 0{,}5 & 0{,}5 & 0 \end{pmatrix} + 0{,}15 * \begin{pmatrix} 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \\ 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \\ 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \\ 0{,}25 & 0{,}25 & 0{,}25 & 0{,}25 \end{pmatrix}$$




```
R := 0.85*matrix(4,4,[[0,0.5,0.5,0],[0,0,1,0],[0.5,0.5,0,0],[0,0.5,0.5,0]])
```

0.0	0.425	0.425	0.0
0.0	0.0	0.85	0.0
0.425	0.425	0.0	0.0
0.0	0.425	0.425	0.0

```
S := 0.15*matrix(4,4,[[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25]])
```

0.0375	0.0375	0.0375	0.0375
0.0375	0.0375	0.0375	0.0375
0.0375	0.0375	0.0375	0.0375
0.0375	0.0375	0.0375	0.0375

```
Q:=R+S
```

0.0375	0.4625	0.4625	0.0375
0.0375	0.0375	0.8875	0.0375
0.4625	0.4625	0.0375	0.0375
0.0375	0.4625	0.4625	0.0375

```
Q**100
```

0.216578177901	0.324561403509	0.42136041859	0.0375
0.216578177901	0.324561403509	0.42136041859	0.0375
0.216578177901	0.324561403509	0.42136041859	0.0375
0.216578177901	0.324561403509	0.42136041859	0.0375

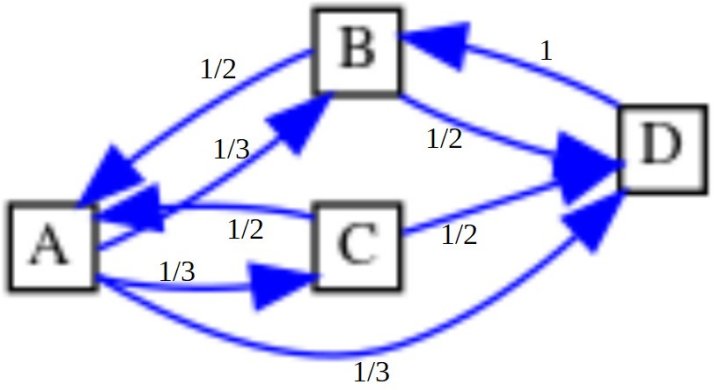
```
matrix(4,4,[[0,0.5,0.5,0],[0,0,1,0],[0.5,0.5,0,0],[0,0.5,0.5,0]])**100
```

0.222222222222	0.333333333333	0.444444444444	0.0
0.222222222222	0.333333333333	0.444444444444	0.0
0.222222222222	0.333333333333	0.444444444444	0.0
0.222222222222	0.333333333333	0.444444444444	0.0

Le classement des pages n’a pas évolué, mais les PageRanks de chacune des pages a évolué un peu avec, notamment, plus de PageRank nul.

Cas n°2 :

$$P = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0,5 & 0 & 0 & 0,5 \\ 0,5 & 0 & 0 & 0,5 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$




```
R:=0.85*matrix(4,4,[[0,1/3,1/3,1/3],[0.5,0,0,0.5],[0.5,0,0,0.5],[0,1,0,0]])
```

0.0,	0.283333333333,	0.283333333333,	0.283333333333
0.425,	0.0,	0.0,	0.425
0.425,	0.0,	0.0,	0.425
0.0,	0.85,	0.0,	0.0

```
S := 0.15*matrix(4,4,[[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25]])
```

0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375

```
Q:=R+S
```

0.0375,	0.320833333333,	0.320833333333,	0.320833333333
0.4625,	0.0375,	0.0375,	0.4625
0.4625,	0.0375,	0.0375,	0.4625
0.0375,	0.8875,	0.0375,	0.0375

```
Q**100
```

0.234721928526,	0.360047050116,	0.104004546416,	0.301226474942
0.234721928526,	0.360047050116,	0.104004546416,	0.301226474942
0.234721928526,	0.360047050116,	0.104004546416,	0.301226474942
0.234721928526,	0.360047050116,	0.104004546416,	0.301226474942

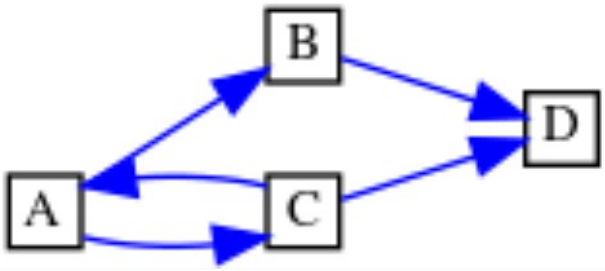
```
matrix(4,4,[[0,1/3,1/3,1/3],[0.5,0,0,0.5],[0.5,0,0,0.5],[0,1,0,0]])**100
```

0.230769230775,	0.384615384609,	0.0769230769206,	0.307692307696
0.23076923075,	0.384615384637,	0.076923076931,	0.307692307681
0.23076923075,	0.384615384637,	0.076923076931,	0.307692307681
0.230769230793,	0.384615384588,	0.076923076913,	0.307692307706

Pas d’évolution notable !

Cas n°3 (le trou noir) :

$$P = \begin{pmatrix} 0 & 0,5 & 0,5 & 0 \\ 0 & 0 & 0 & 1 \\ 0,5 & 0 & 0 & 0,5 \\ 0,25 & 0,25 & 0,25 & 0,25 \end{pmatrix}$$




```
R:=0.85*matrix(4,4,[[0,0.5,0.5,0],[0,0,0,1],[0.5,0,0,0.5],[0.25,0.25,0.25,0.25]])
```

0.0,	0.425,	0.425,	0.0
0.0,	0.0,	0.0,	0.85
0.425,	0.0,	0.0,	0.425
0.2125,	0.2125,	0.2125,	0.2125

```
S := 0.15*matrix(4,4,[[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25]])
```

0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375

```
Q:=R+S
```

0.0375,	0.4625,	0.4625,	0.0375
0.0375,	0.0375,	0.0375,	0.8875
0.4625,	0.0375,	0.0375,	0.4625
0.25,	0.25,	0.25,	0.25

```
Q**100
```

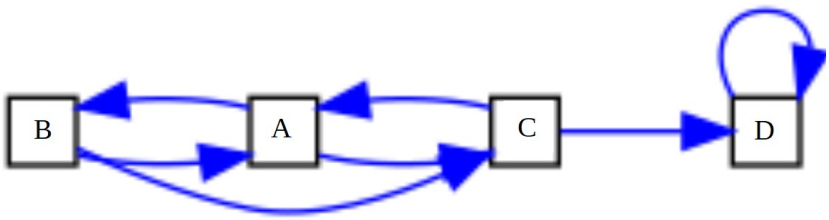
0.20618556701,	0.20618556701,	0.20618556701,	0.381443298969
0.20618556701,	0.20618556701,	0.20618556701,	0.381443298969
0.20618556701,	0.20618556701,	0.20618556701,	0.381443298969
0.20618556701,	0.20618556701,	0.20618556701,	0.381443298969

```
matrix(4,4,[[0,0.5,0.5,0],[0,0,0,1],[0.5,0,0,0.5],[0.25,0.25,0.25,0.25]])**100
```

0.2,	0.2,	0.2,	0.4
0.2,	0.2,	0.2,	0.4
0.2,	0.2,	0.2,	0.4
0.2,	0.2,	0.2,	0.4

Les PageRanks se lissent un peu...

Cas n°4 (le puits) :



$$P = \begin{pmatrix} 0 & 0,5 & 0,5 & 0 \\ 0,5 & 0 & 0,5 & 0 \\ 0,5 & 0 & 0 & 0,5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
R:=0.85*matrix(4,4,[[0,0.5,0.5,0],[0.5,0,0.5,0],[0.5,0,0,0.5],[0,0,0,1]])
```

0.0,	0.425,	0.425,	0.0
0.425,	0.0,	0.425,	0.0
0.425,	0.0,	0.0,	0.425
0.0,	0.0,	0.0,	0.85

```
S := 0.15*matrix(4,4,[[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25],[0.25,0.25,0.25,0.25]])
```

0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375
0.0375,	0.0375,	0.0375,	0.0375

```
Q:=R+S
```

0.0375,	0.4625,	0.4625,	0.0375
0.4625,	0.0375,	0.4625,	0.0375
0.4625,	0.0375,	0.0375,	0.4625
0.0375,	0.0375,	0.0375,	0.8875

```
Q**100
```

0.135499207607,	0.0950871632329,	0.135499207607,	0.633914421553
0.135499207607,	0.0950871632329,	0.135499207607,	0.633914421553
0.135499207607,	0.0950871632329,	0.135499207607,	0.633914421553
0.135499207607,	0.0950871632329,	0.135499207607,	0.633914421553

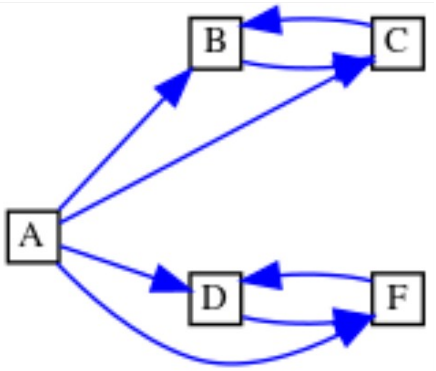
```
matrix(4,4,[[0,0.5,0.5,0],[0.5,0,0.5,0],[0.5,0,0,0.5],[0,0,0,1]])**100
```

2.79434134386e-10,	1.72699792668e-10,	2.79434134386e-10,	0.999999999268
2.79434134386e-10,	1.72699792668e-10,	2.79434134386e-10,	0.999999999268
1.72699792668e-10,	1.06734341719e-10,	1.72699792668e-10,	0.999999999548
0.0,	0.0,	0.0,	1.0

Le puits a disparu ! Et on a des PageRanks exploitables.

Cas n°5 (les poches du web) :

$$P = \begin{pmatrix} 0 & 0,25 & 0,25 & 0,25 & 0,25 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



```
R:=0.85*matrix(5,5,[[0,0.25,0.25,0.25,0.25],[0,0,1,0,0],[0,1,0,0,0],[0,0,0,0,1],[0,0,0,1,0]])
```

0.0	0.2125	0.2125	0.2125	0.2125
0.0	0.0	0.85	0.0	0.0
0.0	0.85	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.85
0.0	0.0	0.0	0.85	0.0

```
S := 0.15*matrix(5,5,[[0.2,0.2,0.2,0.2,0.2],[0.2,0.2,0.2,0.2,0.2],[0.2,0.2,0.2,0.2,0.2],[0.2,0.2,0.2,0.2,0.2],[0.2,0.2,0.2,0.2,0.2]])
```

0.03	0.03	0.03	0.03	0.03
0.03	0.03	0.03	0.03	0.03
0.03	0.03	0.03	0.03	0.03
0.03	0.03	0.03	0.03	0.03
0.03	0.03	0.03	0.03	0.03

```
Q:=R+S
```

0.03	0.2425	0.2425	0.2425	0.2425
0.03	0.03	0.88	0.03	0.03
0.03	0.88	0.03	0.03	0.03
0.03	0.03	0.03	0.03	0.88
0.03	0.03	0.03	0.88	0.03

```
Q**100
```

0.03	0.2425	0.2425	0.2425	0.2425
0.03	0.242500065607	0.242499978131	0.242499978131	0.242499978131
0.03	0.242499978131	0.242500065607	0.242499978131	0.242499978131
0.03	0.242499978131	0.242499978131	0.242500065607	0.242499978131
0.03	0.242499978131	0.242499978131	0.242499978131	0.242500065607

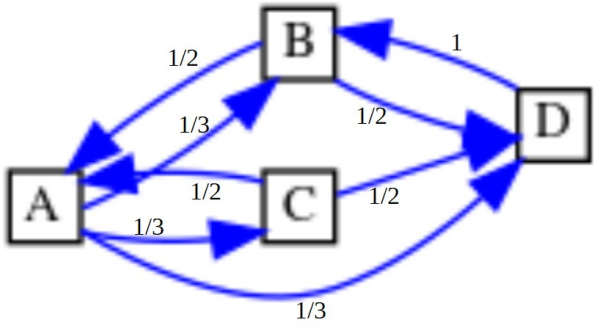
Grâce au saut aléatoire, on est assuré de la convergence de la suite des matrices $(Q^n)_{n \geq 1}$ par le théorème de Perron-Froebenius et on est plus coincé par les poches du web.
On a donc des PageRanks exploitables :

$a \simeq 0,03$, $b \simeq 0,2425$, $c \simeq 0,2425$, $d \simeq 0,2425$, $f \simeq 0,2425$

Le PageRank en SNT en seconde

Larry Page et Sergey Brin utilisent un « surfeur aléatoire » :
- Une fois qu'on a établi la liste, sans aucun classement, des pages web répondant à une requête de recherche, on dépose le surfeur aléatoire sur une des pages au hasard.
- Le surfeur regarde ensuite les liens hypertexte de la page sur laquelle il se trouve, pointant vers les autres pages web de la liste. Il en choisit une au hasard, chaque lien possible étant équiprobable.
- Il répète un très grand nombre de fois cette opération tout en prenant soin de compter le nombre de fois qu'il a visité chaque page.
- Après un nombre « suffisamment grand » de visites par le surfeur aléatoire, le moteur de recherche peut afficher les pages web répondant à la requête en les classant dans l'ordre décroissant de leur nombre de visites.

Si on a un peu de temps, en SNT en seconde, on manipule :
En utilisant une console Python ou un dé, on peut estimer « manuellement » le PageRank des 4 pages web issues d'une requête de recherche représentées par le graphe cité en exemple.




```
>>> from random import randint
>>> randint(1,4)
```

On peut, par exemple, effectuer 50 surfs aléatoires et calculer le PageRank de chaque page.

Ou alors, on écrit un programme Python :
- on peut écrire une fonction surf_aleat(*premier*, *n*) qui simule un surf de *n* clics à partir du sommet *premier* et qui renvoie le PageRank de chaque page web sous la forme d'un dictionnaire (ou d'une liste si on préfère) :

```
def surf_aleat(premier,n):
    """fonction simulant le surfeur aléatoire sur un graphe.
    premier est le sommet de départ, n est le nombre d'étapes à réaliser
    Renvoie un dictionnaire dans lequel : les clés sont les sommets et les valeurs sont les PageRanks
    """
    sommet = premier
    resultat = {'A':0, 'B':0, 'C':0, 'D':0}
    for etape in range(n):
        alea = random()
        if sommet == 'A':
            resultat['A']+=1
            if alea <1/3:
                sommet =  'B'
            elif alea > 2/3:
                sommet = 'C'
            else :
                sommet = 'D'
        elif sommet == 'B':
            resultat['B']+=1
            if alea <1/2:
                sommet =  'A'
            else :
                sommet = 'D'
        elif sommet == 'C':
            resultat['C']+=1
            if alea <1/2:
                sommet =  'A'
            else :
                sommet = 'D'
        elif sommet == 'D':
            resultat['D']+=1
            sommet = 'B'
    for sommet in resultat.keys():
        resultat[sommet] = resultat[sommet]/n
    return resultat
```

Quelques appels on console :

```
*** Console de processus distant Réinitialisée ***
{'C': 0.07695, 'D': 0.30774, 'A': 0.23081, 'B': 0.3845}
>>> surf_aleat('A',100000)
{'A': 0.23141, 'B': 0.38359, 'C': 0.07793, 'D': 0.30707}
>>> surf_aleat('B',100000)
{'A': 0.22957, 'B': 0.38539, 'C': 0.07651, 'D': 0.30853}
>>> surf_aleat('C',100000)
{'A': 0.23096, 'B': 0.38562, 'C': 0.07549, 'D': 0.30793}
>>> surf_aleat('D',100000)
{'A': 0.22942, 'B': 0.38635, 'C': 0.07514, 'D': 0.30909}
>>>
```

Le résultat est sans appel et on retrouve nos résultats.

Autre application des chaînes de Markov : les urnes d'Ehrenfest (1907)

On considère deux urnes A et B et un entier N > 0. Initialement, toutes les boules, numérotées de 1 à N sont dans l'urne A.
On répète n fois les opérations suivantes :
- choisir un nombre entier entre 1 et N ;
- transférer la boule ayant ce numéro dans l'urne dans laquelle elle n'est pas.

On s'intéresse à la variable aléatoire (X_n) donnant le nombre de boules dans l'urne A à l'étape *n*.

Exemple de représentation pour N = 4 :