

# Le Web -Activité « Pagerank »

Cette activité permet d'expérimenter la mise en œuvre de l'**algorithme PageRank** dans une version très simplifiée. Le moteur de recherche a sélectionné un certain nombre de pages représentées par des lettres.

On considère ici un internaute qui clique **au hasard** sur un hyperlien présent sur une page et ainsi de suite.

**Problématique** : quel site possède la plus grande popularité ?

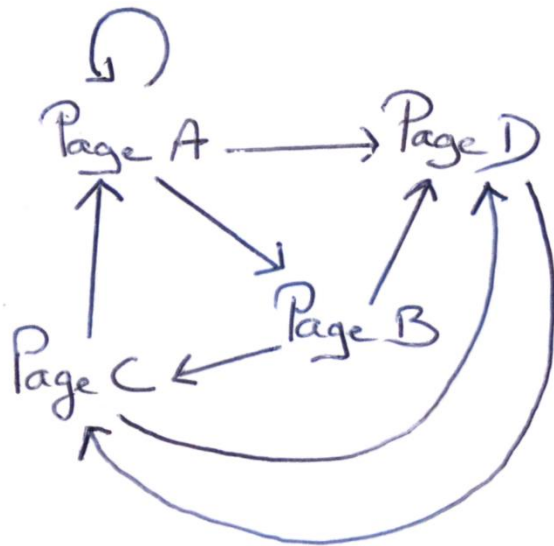
## I/ Simulation du PageRank à l'aide d'un dé

Voici un **graphe orienté** représentant les liens existants entre 4 pages Web.

Une flèche signifie qu'il existe un hyperlien allant de la page actuelle vers la page ciblée.

**Voici les hyperliens présents par page :**

- La page **A** permet de visiter la page B ou la page D ou de rester sur la page A.
- La page **B** permet de visiter la page C ou la page D.
- La page **C** permet de visiter la page C ou la page D.
- La page **D** permet de visiter la page C.



Le programme Python suivant permet de simuler le jet d'un dé à 6 faces non truqué (ou non pipé).

```
from random import randint  
  
print(randint(1,6))
```

A l'aide ce programme à recopier et exécuter sur **Jupyter**, **simuler 50 clics au hasard d'un internaute**. En **déduire le pourcentage** de visite de chaque site et celui qui semble le plus populaire.

**Page A :**

**Page C :**

**Page B :**

**Page D :**

**(-Oral-) Comparer** les résultats obtenus, pourquoi sont-ils différents ? Comment faire pour limiter la **dispersion** des résultats ?

**Appeler le professeur pour validation.**

## II/ Programmation du PageRank à l'aide de Python

Voici un programme écrit en Python permettant de simuler l'algorithme PageRank du graphe orienté précédent.

1/ **Ecrire** ce programme sur Python. **On n'écrira pas les commentaires (en grisé, italique, après le #).**

```
5 from random import choice
6 from random import randint
7
8 # Répertoire des sites enregistrés par le moteur de recherche
9 nom = ["A", "B", "C", "D"]
10
11 # Taille de la liste nom
12 szNom = len(nom)
13
14 # Répertoire des hyperliens sortants de chaque site (à faire en fonction du graphe)
15 # 0 correspond au site "A", 1 correspond au site B etc.
16 hyperliens = [[0,1,3],[2,3],[0,3],[2]]
17
18 # Demande du nombre de tests souhaités converti en entier
19 # Ne pas dépasser 1 millions de tests
20 nbEtapes = int(input("Nombre de tests souhaités : "))
21
22 # Déclaration et initialisation de variables
23 nbVisites = [] # Nombre de fois où chaque site est visité
24 for i in range(0,szNom) :
25     nbVisites.append(0) # Crée le tableau souhaité en fonction du nombre de sites
26
27 page = randint(0,szNom) # Site actuellement visité, valeur aléatoire au départ
28
29 for i in range(nbEtapes) :
30     # Permet de sélectionner aléatoirement le futur site visité PARMI
31     # les hyperliens disponibles du site en question
32     page = choice(hyperliens[page])
33
34     # On incrémente le nombre de fois où le site ainsi visité
35     nbVisites[page] = nbVisites[page] + 1
36
37
38 # Pourcentage de visite de chaque site et affichage des résultats
39 for i in range(0,szNom) :
40     nbVisites[i] = 100.0 * nbVisites[i] / nbEtapes # Conversion implicite en float
41     print(nom[i] + " = " + str(nbVisites[i]) + " %") # Affichage des résultats
```

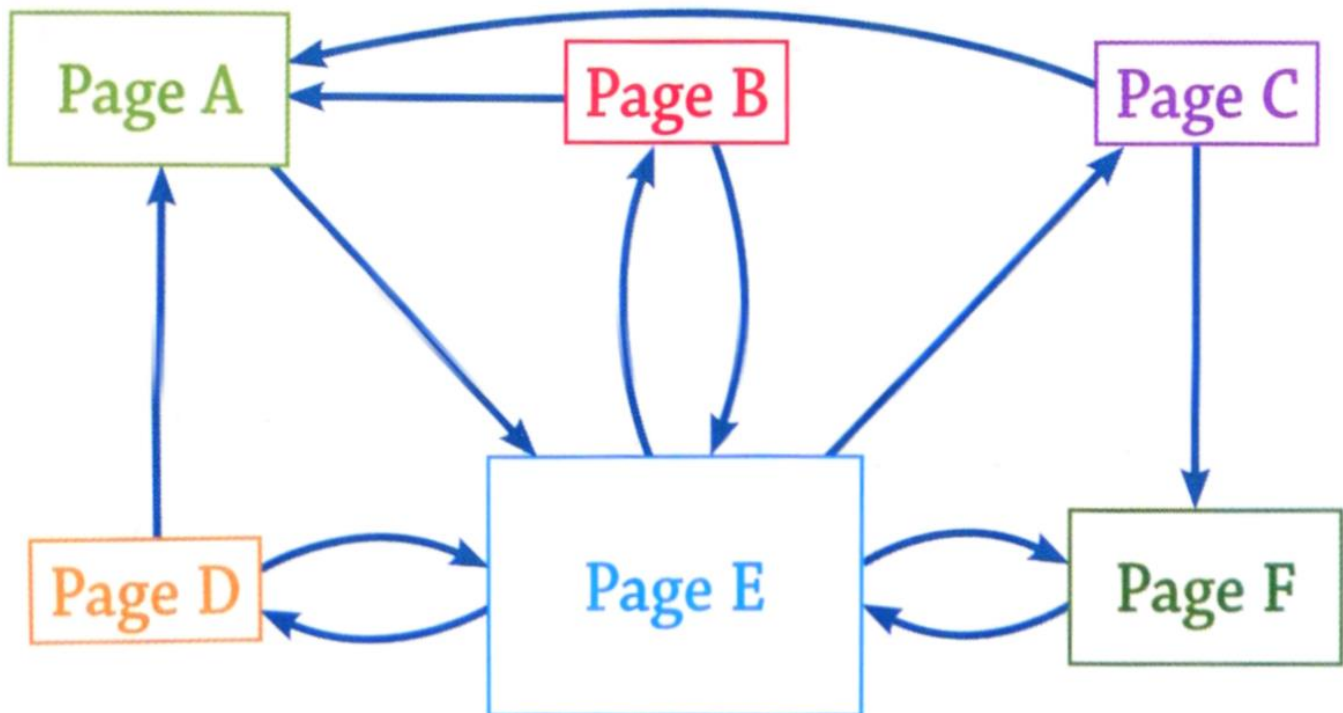
2/ **Tester** pour 100 ; 1 000 ; 10 000 ; 100 000 et 1 000 000 de tests. **Compléter** le tableau ci-dessous après chaque test.

Nbre.Tests	100	1 000	10 000	100 000	1 000 000
% Chaque Page	Page A :	Page A :	Page A :	Page A :	Page A :
	Page B :	Page B :	Page B :	Page B :	Page B :
	Page C :	Page C :	Page C :	Page C :	Page C :
	Page D :	Page D :	Page D :	Page D :	Page D :

3/ **En déduire** quelle page est la plus populaire en **justifiant** la réponse.

### III/ Améliorer la popularité d'un site

Voici un autre graphe orienté représentant une autre situation.



1/ **Copier-coller** dans Jupyter le programme à ce lien :

<https://github.com/Imayer65/SNT/blob/Private/Partie.I.Le%20Web/LeWeb.PageRank.II.py>

2/ **Modifier** ce programme pour qu'il corresponde au graphe orienté ci-dessus.

3/ **Tester** pour 100 ; 1 000 ; 10 000 ; 100 000 et 1 000 000 de tests. **Compléter** le tableau ci-dessous après chaque test.

Nbre.Tests	100	1 000	10 000	100 000	1 000 000
%Chaque.Page	Page A :	Page A :	Page A :	Page A :	Page A :
	Page B :	Page B :	Page B :	Page B :	Page B :
	Page C :	Page C :	Page C :	Page C :	Page C :
	Page D :	Page D :	Page D :	Page D :	Page D :
	Page E :	Page E :	Page E :	Page E :	Page E :
	Page F :	Page F :	Page F :	Page F :	Page F :

A noter : pour un million de tests, les résultats obtenus doivent être proches de ceux-là :

<https://github.com/Imayer65/SNT/blob/Private/Partie.I.Le%20Web/LeWeb.PageRank.Resultats.III.png>

4/ **Vérifier** que si l'on remplace le lien de la page A vers la page E par un lien de la page A vers la page C, les popularités de A et C augmentent.

Aide : on modifiera le programme pour répondre à la question.

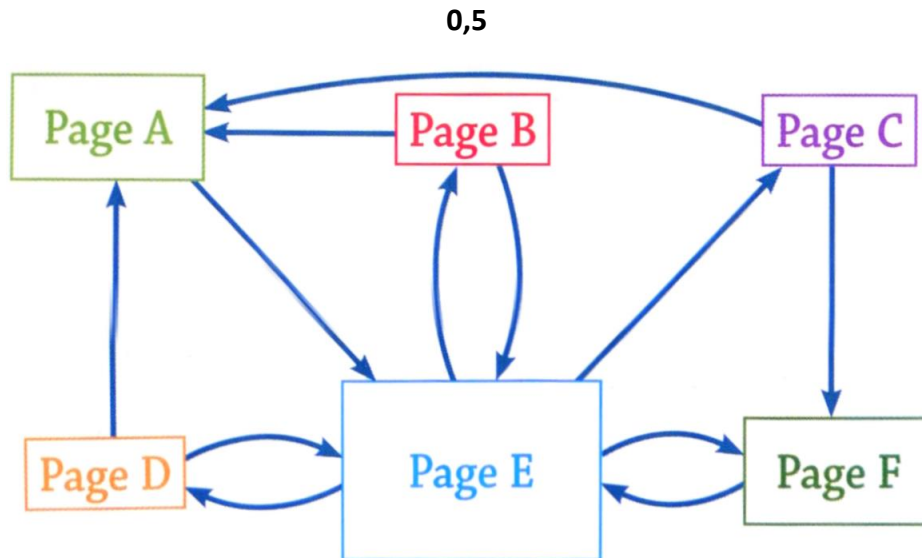
5/ **Expliquer** ce phénomène.

## IV/ Aller plus loin : les matrices de transition

1/ Tout comme pour les arbres pondérés, **compléter** les probabilités d'aller d'une page à l'autre.

**Exemple :**

De la page C, on peut aller à la page A ou F. Il y a donc une probabilité de 0,5 d'aller à la page A (ou F).



2/

**Compléter** le tableau suivant.

Remarque : on écrira 0 s'il n'y a aucun lien.

Ce tableau (sans les cellules en bleu) représente une **matrice de transition** d'un état au suivant. En multipliant successivement suffisamment de fois cette matrice par elle-même, on est capable d'estimer l'état final qui correspond ici à la popularité de chaque site grâce aux valeurs de chaque colonne.

Vers	A	B	C	D	E	F
A	0	0	0	0	1	0
B						
C	0.5					
D						
E						
F						

3/ **Aller** sur le site <https://matrixcalc.org/fr/>

**Compléter** la matrice A (il faut cliquer sur « + » pour augmenter le nombre de lignes/colonnes), **cocher** « Montrer les nombres décimaux » à **2 décimales**, **l'élever à la puissance 100** et **appuyer** sur entrée.

**Important** : écrire les nombres décimaux avec un point comme séparateur et non une virgule.

**Vérifier** alors que les résultats de chaque colonne correspondent à la popularité trouvée par le programme Python précédent ici :

<https://github.com/Imayer65/SNT/blob/Private/Partie.I.Le%20Web/LeWeb.PageRank.Resultats.III.png>.

**Appeler le professeur pour validation.**