- TP 1 : Graphe du Web, HITS -

Pour chaque exercice (excepté les exercices 1 et 2), un code en C++ pré-rempli est disponible à l'adresse suivante :
`http://www.lirmm.fr/~bessy/MathWeb/accueil.html`

- Exercice 1 - Internet Wayback Machine -
Aller sur le site de l'Internet Wayback Machine. Essayer avec la page `http://www.lirmm.fr`, puis la page de votre choix...

- Exercice 2 - Degré entrant d'une page -
1. Avec Google, utiliser l'expression *link :URL* pour connaître une partie des pages ayant un lien vers la page *URL* (les '*backlink*').
2. Par exemple, donner une borne inférieur du 'degré entrant' ainsi que le degré sortant' dans le graphe du web de la page *http ://www.barbapapa.fr*. Pour le degré sortant, on pourra observer qu'un lien sortant est fixé dans le code source html par la syntaxe *<a href="http ://www.loin.fr/"> Par la < /a >*.
3. Yahoo offre la même fonctionnalité. Après pour obtenir plus de précision, il faut utiliser des suites logicielles pour webmasters.

- Exercice 3 - Modélisation du graphe du web -
Le but de l'exercice est d'écrire une procédure permettant de générer un graphe orienté dont les degrés suivent une loi de Zipf :

$$\forall x \in D \; P(d^+(x) = k) = c.\frac{1}{(k+a)^\lambda}$$

On prendra les valeurs obtenues empiriquement par l'étude du graphe du web : pour les degrés entrants : $a_- = 5,75$, et $\lambda_- = 2,7$, pour les degrés sortants : $a_+ = 1,42$ et $\lambda_+ = 2,1$.

1. Montrer que
$$\forall p \in \mathbb{N}^\star \quad \sum_{k \geq p} \frac{1}{(k+a)^\lambda} \leq \frac{1}{(\lambda-1)(p+a-1)^{\lambda-1}}$$
2. En déduire un calcul permettant de normaliser les lois de probabilité, c'est-à-dire, trouver la valeur de c pour les degrés entrants et sortants (à une précision fixée). Dans le code à votre disposition, il s'agit de remplir la fonction `double normal(double a, double lambda)`.
3. Calculer pour chaque sommet un degré sortant et un degré entrant selon les lois précédentes. La fonction correspondante est
`int loi_de_degres(double c, double a, double lambda)`.
4. Calculer l'espérance des degrés entrants et sortants. Comparez ces résultats théoriques avec les valeurs que vous obtenez lors d'une génération aléatoire. La fonction correspondante est `double moyenne(double c, double a, double lambda)`.
5. Pour générer un graphe aléatoire, il faut que la somme des degrés entrants soit égale à la somme des degrés sortants. Si ce n'est pas le cas, équilibrer les degrés déficitaires aléatoirement. La fonction correspondante est
`void equilibre(int n, int degre_entrant[], int degre_sortant[])`.

6. Générer une liste L contenant les sommets du graphes dans laquelle chaque sommet x apparaît exactement $d^-(x)$ fois.
7. Pour chaque sommet x, tirer aléatoirement (sans remise) d^+ sommets dans la liste L, ces sommets forment alors la liste des voisins sortants de x (on ne tiendra pas compte des eventuelles boucles ou arêtes multiples).

- Exercice 4 - Scores de hub et d'authority -

Ecrire une fonction qui permet de calculer les scores d'authority et de hub des sommets d'un graphe orienté.
Testez votre code sur les exemples du cours et tes tds, puis essayez-le sur un graphe généré à l'exercice 3.

- Exercice 5 - Calcul de composantes fortement connexes -

Implémenter l'algorithme de calcul des composantes fortement connexes vu en TD.
Pensez à tester votre algorithme sur des exemples (une primitive donnée sert à obtenir des petits graphes aléatoires).

- Exercice 6 - Simulation d'un crawl -

Etudier la structure d'un crawl d'un graphe généré à l'exercice 3. Plus précisément :

1. Sur un graphe aléatoire généré par la procédure de l'exercice 3, effectuer un crawl (par un parcours en profondeur) sur la moité des sommets. Stocker le graphe G formé des sommets atteints et des arêtes induites sur ces sommets.
2. Identifier C la plus grande des composantes connexes de G. Calculer la proportion des sommets de G qui appartiennent à C, puis la proportion des sommets n'appartenant pas C mais atteignable par un chemin depuis C, ainsi que la proportion des sommets de G n'appartenant pas à C mais qui peuvent atteindre C par un chemin.
3. Calculer la distance moyenne entre deux sommets, ne considérer que les couples (x, y) pour lesquels il existe un chemin de x à y.
4. Calculer le cœfficient de clustering.
5. Comparer les valeurs obtenues à celle du graphe du web.