Grands Réseaux d'Interaction
TP n° 5 : Graphes non-orientés aléatoires

Remarques importantes :
- Les règles générales en TP restent valides, voir feuille du TP 1. En particulier soumettez un **.tar** non compressé qui contient un répertoire. Et n'écrivez que ce qui est demandé sur la sortie standard, ce qui tient sur quelques lignes.
- Ce TP est à rendre sur Moodle pour le **2 novembre**

Le but de ce TP est de générer des graphes non-orientés aléatoires et d'afficher leurs caractéristiques fondamentales, et éventuellement de les enregistrer au format `dot`. On demande donc de rendre **un seul programme** qui prend des paramètres en ligne de commande. Si lancé sans aucun paramètre il doit afficher les options possibles. Le découpage en exercices n'est qu'indicatif

I) Erdős–Rényi

La génération de graphe aléatoire d'Erdős–Rényi consiste à créer un graphe aléatoire $G(n,p)$ dépendant de deux paramètres : le nombre n de sommets et la probabilité p d'existence de chaque arête. On a $0 \leq p \leq 1$ et on tire au sort indépendamment l'existence ou non de chaque arête, avec probabilité p.

Exercice 1 :
Faire une fonction qui étant donné n et p alloue et renvoie un graphe d'Erdős–Rényi $G(n,p)$

II) Power Law

Un modèle aléatoire qui correspond mieux à beaucoup de graphes «de la vraie vie» est le modèle *power law* (loi de puissance).

a) Distribution des degrés

La loi de puissance en question concerne la distribution des degrés : le nombre $Nb(x)$ de sommets de degré x doit être

$$Nb(x) = n_1 x^{-\gamma}$$

où n_1 et γ sont des constantes données. Plus exactement γ est la donnée du problème : c'est un nombre réel typiquement compris entre 2 et 3.

On note un problème dans la formule pour $x = 0$; heureusement les sommets isolés ne nous intéressent pas, on n'en générera donc pas. On fixe $Nb(0) = 0$ et on ne considère donc que $x \geq 1$. Si l'on prend $x = 1$ on a $Nb(1) = n_1 1^{-\gamma} = n_1$, ce qui veut dire que n_1 est le nombre de sommets de degré 1. Le nombre total de sommets n est en théorie donné par

$$n = \sum_{x=1}^{\infty} Nb(x) = \sum_{x=1}^{\infty} n_1 x^{-\gamma}$$

Pour $\gamma > 1$ cette série converge, et même assez vite. Par exemple si $\gamma = 2$ on a l'identité

$$n = \sum_{x=1}^{\infty} n_1 x^{-2} = n_1 \frac{\pi^2}{6} \simeq 1,645\ n_1 \qquad \text{et si } \gamma = 3 \ :\ n = \sum_{x=1}^{\infty} n_1 x^{-3} = n_1 \zeta(3) \simeq 1,202\ n_1$$

ce qui veut dire que si $\gamma = 2$ environ 60% des sommets ont degré 1, et plus de 80% des sommets pour $\gamma = 3$. Ce nombre est «en théorie» car en réalité $Nb(x)$ doit être un entier alors que cette formule produit des nombres réels. En particulier quand $x > (2n_1)^{1/\gamma}$ on a $Nb(x) < 0,5$. Si l'on arrondit à l'entier le plus proche, cela veut dire que $(2n_1)^{1/\gamma}$ est le degré maximal car au-delà il ne reste plus aucun sommet avec un degré supérieur.

Exercice 2 :

Faire une fonction qui, étant donnés les deux paramètres n_1 (entier) et γ (réel > 1) calcule n et génère les degrés de tous les sommets du graphe. Pour cela on peut par exemple procéder ainsi :

1. on commence par calculer la partie entière de $Nb(x)$ pour tout x (en arrondissant au plus près).
2. L'addition de tout cela donne n.
3. On alloue donc un tableau (rappel Java : préférer les structures simples comme les tableaux aux Vector, HashMap, ArrayList etc. quand c'est possible) de n entiers qui seront les degrés des sommets
4. On écrit $Nb(x)$ fois x dans le tableau. C'est-à-dire si $Nb(1) = 7$, $Nb(2) = 3$ et $Nb(3) = 1$ le tableau sera

1	1	1	1	1	1	1	2	2	2	3

5. Le sommet de plus fort degré est toujours le dernier, ce n'est pas aléatoire ! On permute donc aléatoirement les cases du tableau. Ce qui revient à tirer aléatoirement les «vrais» numéros de chaque sommet. Par exemple on obtient

1	1	1	2	3	1	2	1	1	2	1

b) Calcul d'un graphe aléatoire à distribution de degrés donnée

Maintenant que l'on a les degrés de chaque sommet, il s'agit d'avoir le graphe lui-même. L'algorithme le plus simple de génération d'un graphe aléatoire est l'algorithme de **Molloy-Reed**. Il consiste en :

1. Créer une liste L où le sommet u est écrit $deg(u)$ fois. Pour continuer l'exemple précédent, la liste sera donc $L = (0, 1, 2, 3, 3, 4, 4, 4, 5, 6, 6, 7, 8, 9, 9, 10)$
2. Permuter aléatoirement cette liste. Dans exemple on aura $L = (1, 4, 9, 4, 3, 3, 0, 6, 6, 5, 9, 8, 10, 7, 2, 4)$
3. Tenter de créer une arête par couple de la liste. Dans l'exemple on commence par créer les arêtes 1–4 puis 9–4. Il y a trois cas où l'on refuse de créer l'arête :
   - création d'une boucle. Dans l'exemple on aurait à créer l'arête 3–3 ce qui est interdit
   - création d'un arête déjà crée (ainsi si $L = (1, 2, 1, 2)$ on crée une seule arête 1–2)
   - enfin si la liste a un nombre impair d'éléments on ignore le dernier

Noter que cet algorithme se permet donc de modifier la séquence de degré finale : le sommet 3 aurait dû avoir degré 2, or, il aura degré 0 puisqu'on n'a pas créé la boucle 3–3. Mais pour ce TP cela peut être ignoré.

Exercice 3 :

Faire une fonction qui, étant donné une séquence de degré, génère un graphe aléatoire par l'algorithme de Molloy-Reed

Exercice 4 :

En mettant bout-à-bout les deux précédentes fonctions, faire une fonction qui génère un graphe aléatoire ou *Power Law* (en loi de puissance), étant donnés n_1 et γ

III) Génération et étude de graphes aléatoires

Exercice 5 :

faire un programme qui tire un graphe aléatoire d'Erdős–Rényi ou *Power Law*, étant donné les deux paramètres en ligne de commande, puis fait le plus de choses possibles parmi :

- Le sauvegarder en format `dot`
- L'affiche en appelant `dot`, `neato` ou autre programme du package `graphviz` (surtout pas d'affichage en mode texte !)
- Donne le nombre de sommets et d'arêtes
- Donne le nombre de composantes connexes et la taille de la plus grosse
- Donne la valeur de k produisant le k-cœur le plus dense
- Donne son diamètre (cela est mieux noté que le reste car est une question nouvelle)