Rapport TME 3 CPA
Handling a large graph

BAH Thierno
3408625

Exercice 2:
J'ai fait une fonction simple qui à chaque arêtes et chaque noeuds du fichier incrémente des compteurs. Je stock les noeuds déjà croisé pour éviter de les raconter plus tard.

Exercice 3:

Pour nettoyer les fichiers j'ai utiliser les commandes bash suivante:
#tail -n +5 com-amazon.ungraph.txt >> ~/S2/CPA/tme3/amazon.ungraph.txt
#sort -g amazon.ungraph.txt|uniq > amazon_ungraphClean.txt
#sort -g amazon_ungraphClean.txt > amazon_ungraphClean2.txt

Pour trier le fichier selon la première colonne et enlever tout les doublons de type A B, B A ou A A.
Il y a aussi la fonction *clean(fileName)* qui fait le même résultat mais garde en mémoire quelques éléments du graphe.

A partir de là j'ai commencé à travailler sur les fichiers clean mis sur le vrac car nous ne pouvions pas télécharger des fichiers aussi volumineux avec le quotas mis à disposition par la ppti.

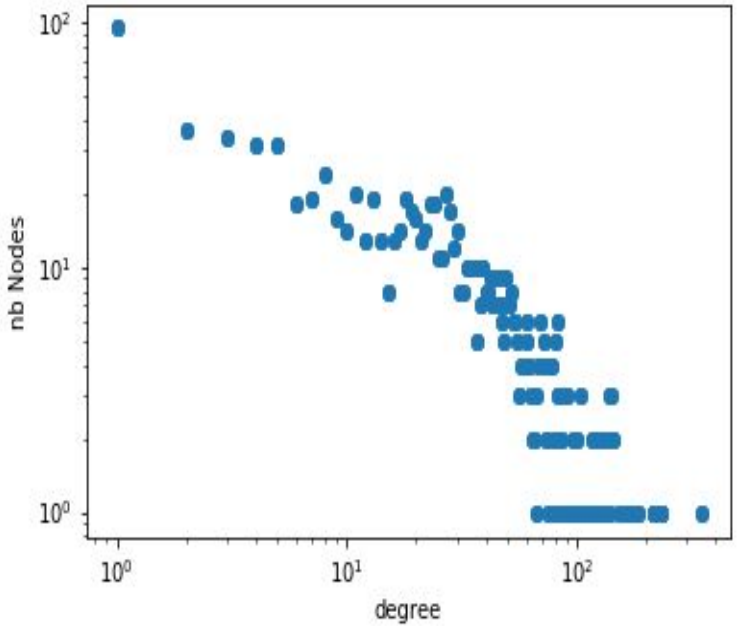
Exercice 4:
degree(fileName) parcours le fichier et stock dans un dictionnaire de clé l'identifiant du noeud et son degré en tant que valeur.

Exercice 5:
En utilisant la fonction *degree(fileName)* j'ai pus avoir accès au degré de chaque noeud dans un second parcours du fichier pour calculer la somme "Quantity" .

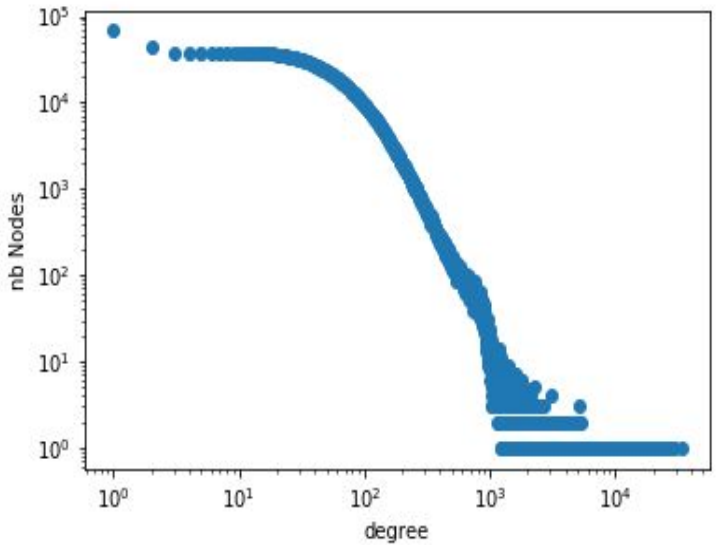
Mesures de temps pour "Quantity" sur les graphes :
email-Eu-core-clean.txt : 88109182 pour 0.024881839752197266 secondes
com-amazon.ungraph-clean.txt : 103415531 pour 1.5718021392822266 secondes
com-lj.ungraph-clean.txt : 789000450609 pour 58.566508054733276 secondes
com-orkut.ungraph-clean.txt : 22292678512329 pour 200.53568148612976 secondes
com-friendster.ungraph.txt : 379856554324947 pour 4390.462798095187 secondes

Exercice 6:

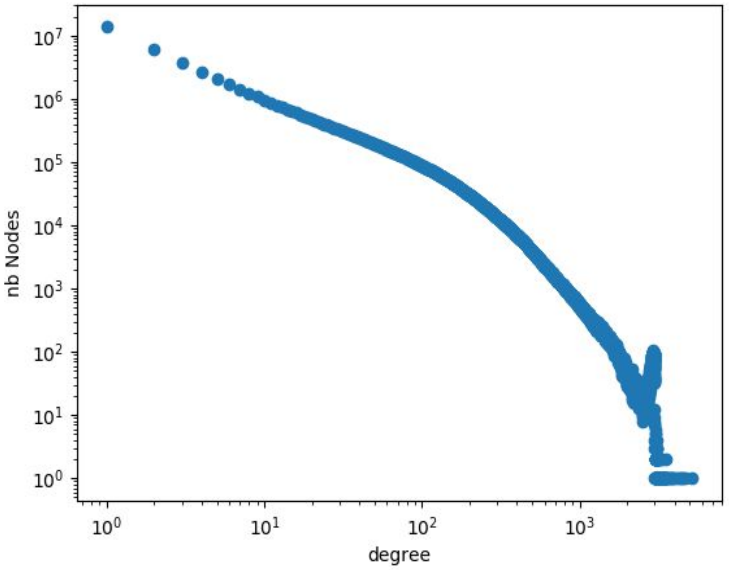
Pour chaque graphe nous obtenons les distributions suivante



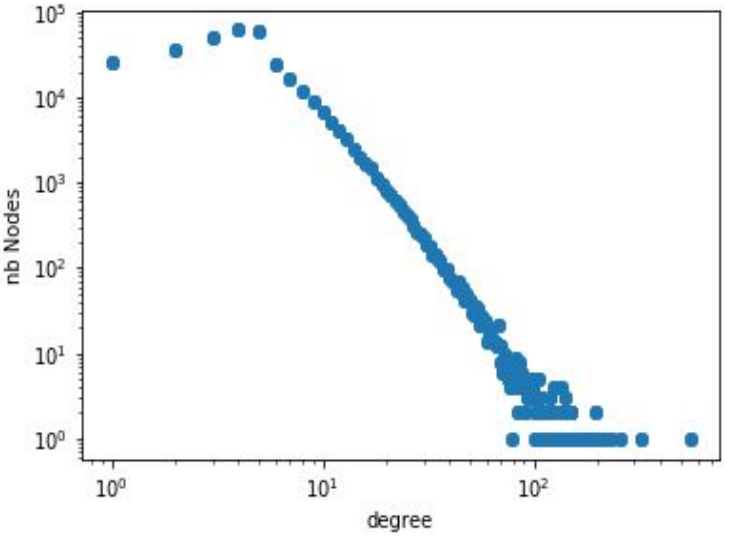
Distribution email-EU-core



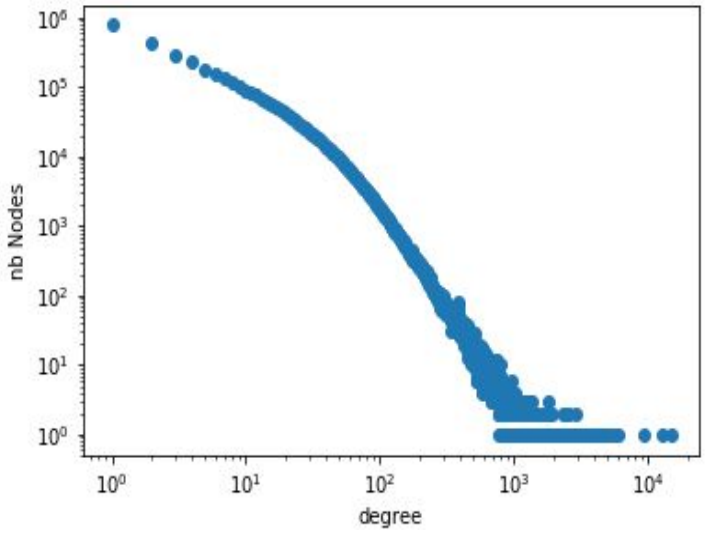
Distribution orkut



Distribution friendster



Distribution amazon



Distribution lj

les échelles des abscisses et des ordonnées sont logarithmiques pour avoir une bonne visibilité de l'ensemble qui est trop "étendue" .

Exercice 7 :

Charger un graphe en mémoire:

Pour charger ma liste d'arêtes je pars d'un fichier clean et pour chaque ligne (qui représente une arête entre deux noeuds) je la stock dans une liste.

Mesures pour la liste d'arête :

email-Eu-core-clean.txt : 0.01457236289978027 secondes
com-amazon.ungraph-clean.txt : 1.353329181671143 secondes
com-lj.ungraph-clean.txt : 1248.564356843643468 secondes
com-orkut.ungraph-clean.txt : ram pleine pc bloqué
com-friendster.ungraph.txt :

Pour charger la matrice je trouve le noeud d'indice maximum et je fait une matrice de taille (maximum*maximum) pour couvrir tous les indices possibles, ensuite je parcours mon fichier clean et pour chaque arête (i,j) je mets les cases d'indices [i][j] et [j][i] à 1. On considère qu'une arête est bidirectionnel.

Mesures pour la matrice d'adjacence :

email-Eu-core-clean.txt : 0.09201383590698242 secondes
com-amazon.ungraph-clean.txt : MemoryError
com-lj.ungraph-clean.txt :
com-orkut.ungraph-clean.txt :
com-friendster.ungraph.txt :

Pour charger la liste d'adjacence je parcours mon fichier et j'ai un dictionnaire qui a comme clé les noeuds et comme valeur la liste des noeuds avec qui il partage une arête (ses voisins).

Mesures pour la liste d'adjacence :

email-Eu-core-clean.txt : 0.04520750045776367 secondes
com-amazon.ungraph-clean.txt : 1.8801968097686768 secondes
com-lj.ungraph-clean.txt : 70.97990083694458 secondes
com-orkut.ungraph-clean.txt : MemoryError (j'ai tuer mon noyau ...)
com-friendster.ungraph.txt :

Exercice 8 :

J'ai implémenter l'algorithme BFS vu en cours, il me retourne tout les noeuds accessibles depuis un point de départ.
Pour trouver la plus grande composante connecté je lance BFS de façon à couvrir tous les noeuds de mon graphe et je retourne la plus grande composante.

Mesures pour la plus grande composante connecter :

email-Eu-core-clean.txt : 986
com-amazon.ungraph-clean.txt : 334863
com-lj.ungraph-clean.txt : 3997962
com-orkut.ungraph-clean.txt :ram pleine pc bloquer
com-friendster.ungraph.txt :

Pour la borne inférieure du diamètre j'ai un BFS2 qui donne la hauteur de chaque points de la composante connecter par rapport au point à partir duquel on lance le BFS2.
Grâce à ce BFS2 je peux trouver un des points les plus éloignés du point à partir du quel j'ai lancer BFS2 et ensuite je peux relancer BFS2 a partir de ce point et ainsi de suite jusqu'à avoir la hauteur maximum qui a été atteinte entre deux points et cela est le diamètre.

Mesures pour la borne inférieure du diamètre :

email-Eu-core-clean.txt : 7
com-amazon.ungraph-clean.txt : 47
com-lj.ungraph-clean.txt : ram pleine pc bloquer
com-orkut.ungraph-clean.txt :
com-friendster.ungraph.txt :

Exercice 9 :

J'ai implémenter l'algorithme du cours, ca marche pour les très petit graphe mais malheureusement je n'ai pas réussi à l'optimiser ou à stocker les données de manière assez intelligente pour me permettre de lancer sur les graphes réel.