TP 1 : Connexité des graphes aléatoires

lionel.rieg@ens-lyon.fr

Ce TD se focalise sur l'étude des graphes aléatoires : comment les générer et quelles sont leur propriétés, en particulier comment se repartissent leurs composantes connexes. On considère ici des graphes non orientés.

Le langage de programmation que j'utilise ici est le C/C++ mais vous êtes libre d'utiliser celui de votre choix.

1 Génération d'un graphe aléatoire

On s'intéresse ici à la génération d'un graphe aléatoire à n sommets et m arêtes par le modèle uniforme d'Erdös et Rényi. Après avoir demandé les valeurs respectives de n et de m, votre algorithme engendre un graphe comme suit :

- L'ensemble des sommets de G est codé par $\{0, \dots, n-1\}$.
- L'ensemble des m arêtes de G est stocké dans un tableau **edge** de taille $m \times 2$ et dont les entrées appartiennent à $\{0, \dots, n-1\}$.

Ainsi, si xy est l'arête de G d'indice k, on aura **edge**$[k][0] = x$ et **edge**$[k][1] = y$. Par exemple, le graphe sur l'ensemble de sommets $\{0, 1, 2, 3\}$ ayant pour arêtes 01, 02, 03, 12, 23 peut être codé par le tableau **edge**[5][2] = {{0,1},{0,2},{0,3},{1,2},{2,3}}.

Écrire une fonction **randomgraph** qui, à partir de deux entiers n et m, construit un graphe aléatoire à n sommets et m arêtes. On permettra la création d'arêtes multiples et de boucles.

En C/C++, on pourra utiliser les fonctions :

- srand (time(NULL)) qui initialise la graine (seed) de la fonction rand sur l'horloge.
- rand()%k qui retourne un entier entre 0 et $k-1$.

2 Calcul des composantes connexes

On va utiliser ici l'algorithme que vous avez vu en cours.

1. Déterminer quelles sont les entrées et les sorties de cet algorithme.
2. Écrire cet algorithme en pseudo-code.
3. À partir du pseudo-code, déterminer quelles sont les structures de données dont vous allez avoir besoin.
4. L'implémenter.

3 Étude d'un graphe aléatoire

On va combiner les deux exercices précédents pour observer comment se répartissent les composantes connexes dans un graphe aléatoire. Pour cela, écrire une fonction **main** sans argument qui :

1. lit deux entiers n et m sur l'entrée standard
2. construit un graphe aléatoire à n sommets et m arêtes
3. calcule ses composantes connexes
4. calcule le nombre de composantes de chaque taille
5. affiche le résultat sous la forme suivante :

   Il y a 464 points isoles.
   Il y a 41 composantes de taille 2.
   Il y a 12 composantes de taille 3.
   Il y a 5 composantes de taille 4.
   Il y a 1 composante de taille 4398.

4 Évaluation de la complexité

Dans l'algorithme de calcul des composantes connexes, lorsqu'on fusionne deux composantes connexes, on réécrit les représentants de tous les sommets de l'une des deux composantes. On peut optimiser cette réécriture en choisissant la composantes la plus petite.

1. Modifier votre algorithme pour inclure cette optimisation.
2. Essayer votre algorithme avec et sans cette optimisation.

5 Seuils de connexité

Utiliser la fonction de l'exercice précédent sur les valeurs de n et m suivantes :

$$
\begin{array}{ll}
n = 10000 & m = 4000 \\
n = 10000 & m = 6000 \\
n = 10000 & m = 10000 \\
n = 10000 & m = 40000 \\
n = 10000 & m = 50000 \\
 & \\
n = 50000 & m = 20000 \\
n = 50000 & m = 30000 \\
n = 50000 & m = 50000 \\
n = 50000 & m = 200000 \\
n = 50000 & m = 250000
\end{array}
$$

Que se passe-t-il aux alentours de $m = n/2$ et $m = n \log_{10} n$?

6 Explication de la complexité

À l'aide des résultats de l'exercice précédent, expliquer la différence de complexité observée avec et sans l'optimisation.