Grands Réseaux d'Interaction
TP 3 : génération aléatoire et routage glouton

Fabien de Montgolfier
fm@irif.fr

mars 2023

Les règles pour le rendu restent les mêmes que le TP1 (archive sans sous-répertoire, pas de package..). En particulier votre programme doit pouvoir être lancé par :

```
unzip *.zip
javac TP3.java
java TP3 -w WS1 100 4 0.1 25 75
```

La dernièere séance de TP, où l'on pourra discuter du rendu du TP2 et du code du TP3 sera présenté, aura lieu : pour le groupe 1, le 10 mars, et pour le groupe 2, le 17 mars. Le rendu du TP est à faire le soir de cette dernière séance.

Il n'y a plus de contrainte de temps et d'espace mémoire dans l'évaluation de ce TP : seuls comptent la correction et la beauté du code.

Le but de ce TP est d'étudier la navigabilité de deux modèles différents : l'anneau de Watts et Strogatz, et la grille de Kleinberg. Il est donc divisé en deux étapes : d'abord générer un graphe muni d'un système de coordonnées, ensuite calculer un routage glouton au sein de ce graphe.

On saura quel modèle utiliser grâce à l'option passée en premier paramètre de la ligne de commande. Le programme se lance de l'une des deux façons suivantes :
`java TP3 -w sortie n k p origine cible`
pour le modèle de Watts et Strogatz, et
`java TP3 -k sortie c origine_x origine_y cible_x cible_y`
pour le modèle de Kleinberg.

1 fichiers en sortie

Le deuxième paramètre en ligne de commande, `sortie`, permet de créer **deux** fichiers de sortie (les deux doivent toujours être créés, sauf en cas d'erreur dans les paramètres). Le premier sera appelé `sortie.txt`, donc en rajoutant l'extension `.txt`. Le fichier correspondant doit suivre le format de Stanford des deux premiers TPs.

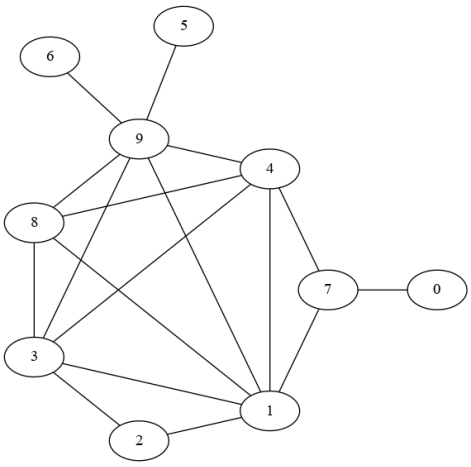
On générera aussi un autre fichier texte nommé `sortie.dot`. Il suit une version minimale du **format DOT** : la première ligne doit être
`graph sortie {`
Ensuite il y a une ligne par arête comme dans le format de Stanford. Mais les deux nombres

sont séparés par deux tirets, et la ligne se termine par un point virgule. Par exemple l'arête entre 5 et 12 donne

```
5 -- 12 ;
```

Le fichier se termine par une dernière ligne avec une accolade fermante :

```
}
```



Cela permet de visualiser vos sortie : le package Linux `graphviz` offre plusieurs programmes convertissant en image un fichier `*.dot`. Il y a `dot, neato, twopi, circo, fdp,` ou encore `sfdp` (voir `man dot`). Leur syntaxe est la même : par exemple `circo -Tpng -O exemple2.dot` va créer l'image `exemple.dot.png` ci-contre (les autres programmes tels que `dot` donnent un résultat moins joli sur `exemple2.dot`). Il existe aussi le programme (plus ou moins) interactif `dotty`.

2 Watts-Strogatz

Avec la commande `-w` on génère un anneau de Wattz et Strogatz. Elle est suivie, dans l'ordre, après le nom de fichier de sortie, des trois paramètres du modèle : n, k et p. n est le nombre de sommets. Les sommets sont numérotés de 0 à $n-1$. Tous les sommets ont même degré : $2k$. Initialement le sommet i est relié aux k sommets situés avant et après lui (ceux de numéro entre $i-k$ et $i+k$, modulo n).

Puis on fait les **rebranchements**. Le rebranchement de l'arête xy est le fait de garder l'extrémité x de l'arête, maus de changer l'autre extrémité (qui ne sera donc probablement plus y). On tire au uniformément au hasard un nouveau sommet z pour remplacer y. Il est pris de sorte que l'on obtienne bien un graphe, sans boucle ni arêtes parallèles. Donc z doit être un sommet qui n'est pas déjà voisin de x, à part y lui-même. Et donc on remplace l'arête xy par xz. Par exemple, si on veut rebrancher l'arête 0–1, avec $n = 10$ et 0 est déjà voisin de 1, 3, 7, et 8, alors les sommets éligibles pour devenir l'autre extrémité de l'arête sont 1, 2, 4, 5, 6 et 9. Donc on a une chance sur six que l'arête 0–1 ne change pas ; une chance sur six qu'elle soit remplacée par 0–2 ; et de même par 0–4 0–5 0–6 et 0–9.

Dans l'exemple donné en haut de la feuille de TP, on veut générer un anneau de 100 sommets, chacun relié initialement aux quatre précédents et suivants (degré 8), avec 10% de chance de rebranchement pour chaque arête.

Chaque arrête a probabilité p d'être rebranchée. Attention à ne pas la tester deux fois (ce qui doublerait p *de facto*) ni aucune fois. Une façon de faire est de regarder, pour x croissant, le rebranchement des arêtes xy pour y allant de $x+1$ à $x+y$.

3 Routage glouton

Le **routage** consiste à faire circuler un message depuis le sommet d'origine jusqu'au sommet cible (la destination finale) en passant de sommet en sommet, seulement en suivant des arêtes du graphe.

Au mieux, on suit un plus court chemin dans le graphe et la distance à la cible diminue de 1 à chaque fois : c'est le routage *optimal*, qui n'est pas traité par ce TP. Mais cela suppose que l'on dispose des distances dans le graphe, ce qui n'est pas toujours le cas.

Un modèle plus général consiste à supposer que l'on dispose de *coordonnées* pour les sommets. Un sommet x enverra alors le message à celui de ses voisins qui est le plus proche de la cible, c'est-à-dire dont les coordonnées se rapprochent le plus des coordonnées de la cible. Notez que le routage glouton peut échouer, si aucun voisin de x n'est strictement plus proche de la cible que x. Ici «proche» signifie au sens des coordonnées et non au sens de la distance dans le graphe.

Pour prendre un exemple, si on fait un routage glouton dans le réseau routier, et que la cible est le pôle Nord, alors à chaque carrefour on prendra toujours la route qui va le plus vers le Nord. Si on arrive dans une impasse (un mur se dresse au Nord devant nous) alors le routage glouton a échoué [1].

Le système de coordonnées pour Watts-Strogatz a une seule dimention (un anneau est en effet une figure géométrique à une dimension). Les coordonnées d'un sommet sont son numéro. Attention quand on va vers la cible au fait que les calculs sont modulo n : le sommet $n-3$ et le sommet 2 sont à distance 5.

Les deux derniers paramètre de la commande `-w` sont donc le numéro du sommet origine et le numéro du sommet destination.

4 La grille de Kleinberg

La grille de Kleinberg est, comme son nom l'indique, une grille : chaque sommet a une abscisse et une ordonnée, et un sommet est répertorié par ses coordonnées (x, y). Un sommet (x, y) est relié à ses quatre voisins $(x-1, y)$ au Nord, $(x, y-1)$ à l'Ouest etc. Un sommet sur les bords n'a que trois voisins, et les quatres coins deux voisins. La grille est un carré de côté c (le troisième paramètre de la ligne de commande, qui est le seul paramètre di modèle) ce qui fait que le nombre de sommets est $n = c^2$. À chaque couple (i, j), pour i et j entre 0 (inclus) et c (exclus), correspond un unique sommet.

Donc pour le routage glouton, on essaye de se rapprocher du voisin le plus proche, sachant que la proximité entre deux sommets $A = (x, y)$ et $B = (x', y')$ est donnée par la distance euclidienne $d(A, B) = \sqrt{(x-x')^2 + (y-y')^2}$. Les quatre derniers paramètres du programme sont les coordonnées (x, y) de l'origine puis de la cible.

Cette proximité $d(A, B)$ n'est pas la même chose que la distance dans le graphe à cause des **liens longs**. Chaque sommet $A = (x, y)$ reçoit un lien long. Ce lien long va le relier à un autre sommet choisi aléatoirement. Mais pas aléatoirement uniformément : la probabilité d'être relié à $B = (x', y')$ est inversement proportionnelle à l'**inverse du carré de la distance** $d(A, B)$.

1. comme à ma connaissance on ne peut pas aller en voiture au pôle Nord on va nécéssairement échouer, mais dans une impasse à Paris ou à Tromsø ?

À cause des liens longs, le degré moyen des sommets est 5, mais un sommet peut voir plusieurs liens longs lui arriver dessus (s'il a été tiré au sort par plusieurs sommets différents) ou aucun. Notez que, encore une fois, il faut créer un graphe non-orienté sans boucle ni multi-arête.

Toute la difficulté pour la grille de Kleinberg consiste à générer, pour tout sommet A, le lien long de A. Une façon de faire est de calculer pour tous les sommets (différents de A) l'inverse du carré de leur distance à A. La somme de tous ces inverses carrés de distances est S.

$$S = \sum_{B \neq A} \frac{1}{d(A,B)^2}$$

La probabilité qu'un sommet B soit reliée à A est alors l'inverse du carré de sa distance à A divisée par S (pour avoir une probabilité qui somme à 1) soit

$$p_B = \frac{1}{S.d(A,B)^2}$$

Pour pouvoir tirer facilement selon les p_B on calcule leur somme partielle P_B

$$P_B = \sum_{i \leq B} p_i$$

où les sommets sont numéroté de 0 à n-1. On a ainsi $P_0 = p_0$, $P_1 = p_0 + p_1$ jusqu'à $P_{n-1} = 1$ (la somme de toutes les probabilités fait 1).

On tire un nombre p au hasard entre 0 et 1 : si $P_{i-1} < p \leq P_i$ alors c'est le sommet numéro i qui est choisi. On ajoute donc une arête entre A et i. Notez qu'il existe d'autres façons de faire le calcul...

Les quatre derniers paramètres de la ligne de commande sont les coordonnées de l'origine (abcisse et ordonnée) puis de la cible.

5 Résultats demandés et exemples

Votre programme doit

- **générer** deux ficihiers : le graphe au format de Stanford et le même au format DOT sous les noms `sortie.txt` et `sortie.dot`, où `sortie` est le deuxième paramètre du programme
- **afficher** deux lignes : la première est les différents sommets parcourus par le routage glouton depuis l'origine jusqu'à la cible (ou jusqu'à l'échec), et la deuxième est soit la longueur du chemin, soit la notification d'un échec.

Pour la sortie DOT de la grille de Kleinberg le noeud doit avoir comme nom ses coordonnées (x, y). On crée dans le fichier DOT des lignes de la forme

```
num [label ="(x,y)"];
```

Voir les fichiers DOT fournis en exemple et obtenus par les commandes suivantes :

```
java TP3 -w WS1 100 4 0.1 25 75
25 21 17 13 10 6 2 98 94 90 86 82 78 75
longueur chemin : 13
```

```
java TP3 -w WS2 100 4 0.1 25 75
25 95 78 75
longueur chemin : 3

java TP3 -w WS3 100 4 0.1 25 75
25 21 13 57 61 74
Glouton coincé, échec !

java TP3 -k Kleiberg1 12 1 1 10 10
13 (1,1) ; 14 (1,2) ; 127 (10,7) ; 128 (10,8) ; 129 (10,9) ; 130 (10,10) ;
longueur chemin : 5

java TP3 -k Kleiberg2 100 1 1 99 99
101 (1,1) ; 1090 (10,90) ; 1190 (11,90) ; 1290 (12,90) ; 1393 (13,93) ; 1586 (15,86) ;
1693 (16,93) ; 1793 (17,93) ; 3789 (37,89) ; 3889 (38,89) ; 3989 (39,89) ; 4188 (41,88) ;
4490 (44,90) ; 4689 (46,89) ; 4790 (47,90) ; 4890 (48,90) ; 5188 (51,88) ; 5888 (58,88) ;
5988 (59,88) ; 6998 (69,98) ; 8397 (83,97) ; 8790 (87,90) ; 8890 (88,90) ; 8990 (89,90) ;
9090 (90,90) ; 9598 (95,98) ; 9698 (96,98) ; 9798 (97,98) ; 9998 (99,98) ; 9999 (99,99) ;
longueur chemin : 29
```

Pour le dernier exemple j'ai coupé la longue première ligne