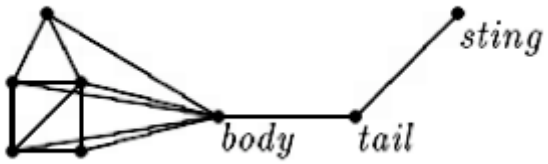
École normale supérieure **2010-2011**
Département d'informatique

Algorithmique et Programmation
TD n° 5 : Graphes

Exercice 1. Un *scorpion* est un graphe non-orienté G de la forme suivante : il y a 3 sommets spéciaux, appelé le *dard* (sting), la *queue* (tail), et le *corps* (body), de degrée 1, 2 et $n-2$, respectivement. Le dard est connecté seulement à la queue ; la queue est connecté seulement au dard et au corps ; et le corps est connecté à tous les sommets sauf le dard. Les autres sommets de G peuvent être connecté aux autres de façon arbitraire.



Donner un algorithme qui fait seulement $O(n)$ tests à la matrice d'adjacence de G et détermine si G est un scorpion.

Exercice 2. Imaginons d'avoir un graphe acyclique $G = (V, E)$, une k-coloration de G est une fonction $c : V \to \{1, k\}$ associant à chaque sommet u de G une *couleur* $c(u)$, telle que si $(u, v) \in E$ alors $c(u) \neq c(v)$. La fonction c est dit fonction de coloration. Un graphe pour lequel il existe une coloration qui utilise k couleurs est dit k-coloriable.

1. Montrer que chaque graphe connexe et acyclique est en fait 2-coloriable.
   Un graphe $G = (V, E)$ est connexe si et seulement si quels que soient les sommets u et v de E, il existe un chemin de u à v.
2. Montrer qu'un graphe qui admet de cycle de longueur 3 n'est pas 2-coloriable.
3. Donner un algorithme qui détermine si un graphe est 2-coloriable et, si c'est le cas, donner une 2-coloration. Quel est le nombre d'opérations, en ordre de grandeur, effectué par l'algorithme. (on représentera un graphe par liste d'adjacences.)
4. On suppose que, pour tout u, la paire $\{u, u\}$ n'est pas une arête et qu'il y a au plus Δ sommets v tel que $\{u, v\}$ est une arête. Montrer en construisant un algorithme, que le graphe est alors $(\Delta+1)$-coloriable. Quel est le nombre d'opérations, en ordre de grandeur, effectues par l'algorithme ?
5. Montrer que tout graphe possédant n sommets et 3-coloriable peut être colorié avec $O(\sqrt{n})$ couleurs par un algorithme effectuant un nombre non exponentiel d'opérations. On utilisera les algorithmes des questions précédentes en remarquant que dans un graphe 3-coloriable, pour tout sommet u donné, le sous-graphe correspondant aux sommets v tel que $\{u, v\}$ est une arête est lui 2-coloriable.
6. Généraliser la technique de la question précédente pour colorier avec $O(kn^{f(k)})$ couleurs un graphe k-coloriable. On fera en sorte que la fonction $f(k)$ (qu'on donnera) soit la plus petite possible.

Exercice 3. Soit G un graphe connecté, non-orienté avec éventuellement plusieurs arêtes entre les n sommets. Une *coupe* C dans G est un ensemble d'arêtes qui si on les enlève, G devient non-connexe. Une *coupe minimale* est une coupe de cardinalité minimale. Le problème de trouver une coupe maximale est NP-complet.

L'idée de l'algorithme est de choisir uniformément une arête et de fusionner les deux sommets en un seul sommet en mettant sur ce sommet les arêtes qui arrivaient aux deux sommets initiaux et en enlevant les boucles. On appelle cette opération une *contraction*. On voit que même si le graphe initial n'avait qu'une

seule arête entre chaque sommet, le graphe ayant subi une contraction peut en contenir au plus deux. Ce processus diminue d'une unité le nombre de sommets. L'algorithme effectue des contractions jusqu'à ce que le nombre de sommets soit égal à 2 et retourne comme valeur le nombre d'arêtes entre ces deux points.

1. Montrer qu'une contraction d'arête ne diminue pas la valeur d'une coupe minimale si on n'enlève pas d'arête d'une coupe minimale.
2. Soit k la valeur d'une coupe minimale. Montrer que G a au moins $kn/2$ arêtes.
3. Soit $\mathcal{E}_i$ l'événement de ne pas choisir une arête de C à la i-ième étape, pour $1 \leq i \leq n-2$.
   (a) Montrer que $\Pr[\mathcal{E}_1] \geq 1 - 2/n$.
   (b) Montrer que $\Pr[\mathcal{E}_2|\mathcal{E}_1] \geq 1-2/(n-1)$ et plus généralement que $\Pr[\mathcal{E}_i|\cap_{j=1}^{i-1}\mathcal{E}_j] \geq 1-2/(n-i+1)$.
   (c) Montrer que la probabilité trouver une coupe minimale par ce procédé est au moins $\frac{2}{n(n-1)}$.
4. Montrer comment obtenir un algorithme dont la probabilité d'échec soit $< 1/e$ et donner sa complexité où e est la base du logarithme népérien. Si on veut que la probabilité d'échec soit aussi petite que l'on veut, par exemple $1/n$, quelle est la complexité ?

Exercice 4. Soit un graphe non-orienté, connecté, avec un ensemble de sommets $V = \{1, \ldots, n\}$ et $|E| = m$ arêtes de poids unitaire. On note A la matrice d'adjacence booléenne $n \times n$ et $A_{ij} = A_{ji} = 1$ si l'arête (i, j) est dans E et 0 sinon. Étant donné A, la matrice des distances D est une matrice d'entiers positifs telle que D_{ij} vaut la longueur du plus court chemin entre le sommet i et le sommet j. Les diagonales de A et D valent 0. Le diamètre d'un graphe est le maximum des plus court chemins entre toute paire de sommets.

Le but de cet exercice est de montrer que pour toute paire de sommets le calcul de la plus courte distance (APD) et le calcul d'un plus court chemin (APSP) entre deux sommets, peut se faire en un peu plus du coût $\mathrm{MM}(n)$ d'une multiplication de 2 matrices $n \times n$. Floyd-Warshall s'exécute en $\Theta(n^3)$.

1. Soit $G'(V, E')$ le graphe obtenu en plaçant une arête entre chaque paire de sommets $i \neq j \in V$ qui sont à distance 1 ou 2 dans G. On notera A' la matrice d'adjacence de G' et D' la matrice des plus courtes distances dans G'. ([M] : montrer comment calculer le graphe G' s'il est donné par sa matrice d'adjacence ou sous forme de liste chaînée pour chaque sommet.)
   (a) Montrer que si $Z = A^2$, alors il existe un chemin de longueur 2 dans G entre chaque paire de sommets i et j si et seulement si $Z_{ij} > 0$. De plus, la valeur de Z_{ij} est le nombre de chemins distincts de longueur 2 entre i et j.
   (b) Supposons que le diamètre de G soit au plus 2. Montrer que G' est un graphe complet et exprimer dans ce cas D en fonction de A et A'.
2. En général, G peut avoir un diamètre arbitrairement grand $\leq n$.
   (a) Montrer alors que pout toute paire $i, j \in V$,
   – Si D_{ij} est paire, alors $D_{ij} = 2D'_{ij}$.
   – Si D_{ij} est impaire, alors $D_{ij} = 2D'_{ij} - 1$.
   On en déduit que la matrice D peut être calculée avec D' si on connaît la parité de tous les plus courts chemins.
   (b) Montrer que pour chaque paire de sommets distincts i et j dans G,
   – Pour chaque voisin k de i, $D_{ij} - 1 \leq D_{kj} \leq D_{ij} + 1$.
   – Il existe un voisin k de i tel que $D_{kj} = D_{ij} - 1$.
   (c) Montrer que pour chaque paire de sommets distincts i et j de G,
   – Si D_{ij} est paire, alors $D'_{kj} \geq D'_{ij}$ pour chaque voisin k de i dans G.
   – Si D_{ij} est impaire, alors $D'_{kj} \leq D'_{ij}$ pour chaque voisin k de i dans G. De plus, il existe un voisin k de i dans G tel que $D'_{kj} < D'_{ij}$.
   Soit $\Gamma(i)$ l'ensemble des voisins de i dans G et $d(i)$ le degré de i. En déduire la caractérisation suivante : Pour toute paire de sommets distincts i et j de G :

- D_{ij} est paire si et seulement si $\sum_{k\in\Gamma(i)} D'_{kj} \geq D'_{ij}d(i)$.
- D_{ij} est impaire si et seulement si $\sum_{k\in\Gamma(i)} D'_{kj} < D'_{ij}d(i)$

(d) Montrer comment calculer $\sum_{k\in\Gamma(i)} D'_{kj}$ en utilisant une multiplication matricielle. Remarquez que $Z_{ii} = d(i)$ pour tout i et proposer alors un algorithme récursif pour calculer D. Analyser sa complexité en terme de multiplication matricielle utilisant la fonction $T(n, \delta)$ où δ est le diamètre du graphe.

3. Pour calculer la matrice des plus courts chemins, on va utiliser une sous-routine BPWM qui calcule un témoin du produit de deux matrices booléennes.

   Soit A et B deux matrices booléeennes. Alors le produit $P = AB$ est tel que $P_{ij} = 1$ s'il existe un *témoin* k tel que $A_{ik} = B_{kj} = 1$. On se propose de chercher un algorithme qui calcule une matrice W tel que W_{ij} est un témoin du produit.

   (a) Supposons qu'il n'existe qu'un seul témoin. Montrer comment calculer W en effectuant une seule multiplication matricielle.

   Il existe un algorithme randomisé en $O(\mathrm{MM}(n)\log^2 n)$ pour calculer la matrice des témoins. On ne cherchera pas à construire cet algorithme.

   Pour calculer les plus courts chemins entre chaque paire de sommets, on ne va pas décrire explicitement tous les chemins car sinon on peut utiliser un temps $\Omega(n^3)$.

   (b) Construire un graphe à n sommets avec $\Omega(n^2)$ paires de sommets à distance $\Omega(n)$.

   On va utiliser la matrice des successeurs qui contient en (i, j), le successeur de i dans un plus court chemin de i à j.

   (c) Soit B^d la matrice booléenne telle que $B^d_{kj} = 1$ si et seulement si $D_{kj} = d - 1$. En appliquant l'algorithme BPWM entre A et B^d, montrer comment calculer la matrice des successeurs pour toute paire de sommets à distance d en $O(\mathrm{MM}(n)\log^2 n)$. Quelle est la complexité d'un algorithme basé sur cette idée si on veut la matrice des successeurs pour *toute* paire de sommets ?

   (d) Montrer qu'en utilisant la question 2b), on peut uniquement calculer B^s pour $s = d \bmod 3$.