TD 5

Algorithmique

Rappel de probabilité:

Deux événements $\mathcal{E}_1$ et $\mathcal{E}_2$ sont dits *indépendants* si la probabilité que les deux arrivent en même temps est

$$\Pr[\mathcal{E}_1 \cap \mathcal{E}_2] = \Pr[\mathcal{E}_1] \times \Pr[\mathcal{E}_2].$$

Dans le cas le plus général où $\mathcal{E}_1$ et $\mathcal{E}_2$ ne sont pas nécessairement indépendants, on a:

$$\Pr[\mathcal{E}_1 \cap \mathcal{E}_2] = \Pr[\mathcal{E}_1|\mathcal{E}_2] \times \Pr[\mathcal{E}_2] = \Pr[\mathcal{E}_2|\mathcal{E}_1] \times \Pr[\mathcal{E}_1],$$

où $\Pr[\mathcal{E}_1|\mathcal{E}_2]$ représente la *probabilité conditionnelle* de $\mathcal{E}_1$ étant donné $\mathcal{E}_2$. Quand on a un ensemble d'événements non nécessairement indépendants, on a:

$$\Pr[\cap_{i=1}^{k}\mathcal{E}_i] = \Pr[\mathcal{E}_1] \times \Pr[\mathcal{E}_2|\mathcal{E}_1] \times \Pr[\mathcal{E}_3|\mathcal{E}_1 \cap \mathcal{E}_2] \ldots \Pr[\mathcal{E}_k| \cap_{i=1}^{k-1} \mathcal{E}_i].$$

Exercice 1: Coupe minimale dans un graphe

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 2: Clôture transitive dans un graphe

1. Rappeler l'algorithme de Floyd-Warshall pour calculer la distance minimale entre tout couple de sommets et donner sa preuve et sa complexité.
2. Montrer qu'il permet de calculer la clôture transitive en $\Theta(n^3)$ en temps.
3. Montrer que s'il existe un algorithme qui calcule la clôture transitive d'une matrice $n \times n$ en $A(n)$ additions, multiplications d'éléments d'un semi-anneau et si $A(3n) \leq c \cdot A(n)$ pour un réel $c > 0$ et tout n entier positifs, alors il existe un algorithme de multiplication avec $M(n) = O(A(n))$ additions et multiplications.
4. Montrer que si le produit de deux matrices $n \times n$ peut être calculé en $M(n)$ additions et multiplications et si $4 \cdot M(n/2) \leq M(n)$ et $M(2n) \leq c \cdot M(n)$ pour un c et tout n, alors la clôture transitive se calcule en $A(n) = O(M(n))$ additions et multiplications.
5. Montrer que le semi-anneau booléen $B = (\{0, 1\}, \vee, \wedge, 0, 1)$ n'est pas un anneau.
6. Soit A et B deux matrices $n \times n$ booléeennes. Le produit peut se calculer en $O(n^{\log 7} \cdot (\log n)^2) = O(n^{2.82})$ opérations binaires, ainsi que la clôture transitive.

Exercice 3: Distance minimale entre tout couple de sommets

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 ont montré comment résoudre ce problème en $\Theta(n^3)$. On cherche à ne pas dépasser cette borne sachant que la multiplication matricielle peut s'implanter plus rapidement qu'en $O(n^3)$.

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'.

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 $s = d \bmod 3$.

Exercice 4: Problème 2-SAT

Soit $x_1, \ldots, x_n$ un ensemble de n variables booléennes indexées par les entiers $\leq n$. Un littéral sur cet ensemble de variables est une variable x_i $i \leq n$, ou sa négation $\neg x_i$. Une 2-clause sur cet ensemble de variable est une disjonction de deux littéraux $\ell \vee \ell'$. Une assignation τ est une fonction de l'ensemble $x_1, \ldots, x_n$ dans $\{0, 1\}$. La valeur d'un littéral ℓ relativement à une assignation τ, soit $V(\ell, \tau)$ est égale à $\tau(x_i)$ si $\ell = x_i$ et à $1 - \tau(x_i)$ si $\ell = \neg x_i$. La valeur d'une clause c relativement à une assignation τ, soit $V(c, \tau)$, est le maximum des valeurs de ses deux littéraux. Le problème 2-SAT s'énonce comme suit :

Étant donné un entier n et une liste de 2-clauses sur l'ensemble de variables $x_1, \ldots, x_n$, déterminer s'il existe une assignation τ qui donne à toutes les clauses de la liste la valeur 1 (problème de décision). Calculer une telle assignation si elle existe (calcul d'une solution).

1. On associe à une liste de 2-clauses sur l'ensemble de variables $x_1, \ldots, x_n$ un graphe orienté G dont les sommets sont les littéraux sur l'ensemble de variables $x_1, \ldots, x_n$. Pour chaque clause $\ell \vee \ell'$, on ajoute à G une arête allant de $\neg\ell$ à ℓ' et une arête allant de $\neg\ell'$ à ℓ (où on pose $\neg\neg x_i = x_i$).
   (a) Montrer que l'on peut calculer G en temps $O(n + m)$ où n est le nombre de variables et m le nombre de clauses. On explicitera un choix de structures de données et on proposera une description de l'algorithme en terme de pseudo-code.
   (b) Montrer que, s'il existe dans G un chemin allant de u à v, il existe aussi un chemin allant de $\neg v$ à $\neg u$.
2. Soit τ une telle assignation. On étend τ à l'ensemble des sommets de G en posant $\tau(\neg x_i) = 1 - \tau(x_i)$. Montrer que $V(c, \tau)$ vaut 1 pour toutes les clauses de la liste donnée si et seulement si τ est croissante sur tout chemin γ du graphe G (ce qui signifie : si $\gamma = u_0, \ldots, u_k$, alors $\tau(u_i) \leq \tau(u_{i+1})$ pour $i = 0, \ldots, k - 1$).
3. En utilisant l'algorithme qui détermine les composantes fortement connexes d'un graphe orienté ou un autre des algorithmes sur les graphes, donner un algorithme qui résout le problème de décision 2-SAT. Montrer sa correction et évaluer sa complexité.
4. Comment calculer une assignation lorsque la réponse au problème de décision est positive ? On évaluera la complexité de l'algorithme proposé.