Examen Algorithmique I (2024-25)

Tous documents écrits autorisés — spécialement les notes de cours.

On insistera sur la correction et la clarté des réponses.

Toute affirmation devra être justifiée.

Dans la mesure du possible, les algorithmes du cours utilisés ne devront pas être recopiés. On y fera référence par leur *nom*. On fera référence aux questions précédentes par leur *numéro*.

Un code, même bien écrit, est indéchiffrable seul. Donc, pas de pseudo-code (je ne suis pas une machine). Sauf indication contraire, on décrira les algorithmes à haut niveau (« on utilise la procédure de la question XXX pour calculer YYY », « on itère sur les ZZZ, et pour chacun on calcule BBB par l'algorithme CCC », etc.), le plus clairement possible.

N'inventez pas vos propres notations, et réutilisez les miennes, même si elles ne vous plaisent pas.

Je veux une copie au propre, pas un brouillon : pas de rature, pas de faute d'orthographe, notamment.

Je corrigerai toutes les copies en corrigeant d'abord toutes les questions 1 de tout le monde, puis toutes les questions 2, etc. : je ne me souviendrai donc pas de ce que vous avez écrit dans une question précédente.

Je me réserve le droit de ne pas chercher à comprendre une réponse illisible, insuffisamment claire, ou plus longue ou compliquée que nécessaire.

Les parties ne sont pas indépendantes, sauf la partie 5. Les parties 1 et 2 sont très faciles. J'estime que vous devriez les avoir terminées en moins de 20 minutes. Ne traînez pas, vous aurez besoin de davantage de temps pour les parties suivantes.

J'ai attribué à chaque question un niveau de difficulté, de ▯▯▯▯▯ (trivial) à ▮▮▮▮▮ (très difficile ou très long). Mon appréciation de la difficulté est subjective. Tout résultat qui est conséquence facile d'un résultat du cours, quelle que soit la difficulté de ce résultat, est considéré comme étant facile.

⋆ ⋆ ⋆

On rappelle qu'une représentation par listes de successeurs d'un graphe orienté $G \stackrel{\text{def}}{=} (S, A)$ consiste en un tableau `succ` indexé par les sommets $u \in S$, tel que `succ` (u) contient une liste *sans doublon* des successeurs de u dans G. (Une liste, pas un tableau.)

Tous nos graphes orientés sont représentés par listes de successeurs.

On supposera dans la suite que les sommets d'un tel graphe sont des entiers allant de 1 à n.

1 Des listes d'arcs aux listes de successeurs

Question 1 ▯▯▯▯▯ Donner un algorithme qui, étant donné un tableau T de p couples d'entiers $(u, v) \in \{1, \cdots, n\}^2$ (possiblement avec doublons), le trie dans l'ordre lexicographique ($(u, v) \leq_{lex} (u', v')$ si et seulement si $u < u'$, ou $u = u'$ et $v \leq v'$), en temps $O(p + n)$. C'est une question de cours : donner les noms de l'algorithme ou des algorithmes utilisés, et s'il y en a plusieurs, comment on les combine.

Je ne veux donc pas de réponse de plus de quelques lignes.

Question 2 ▮▮▯▯▯ En déduire un algorithme en temps $O(p+n)$ qui prend en entrée deux entiers naturels p et n et une liste L de p couples d'entiers entre 1 et n, possiblement avec doublons, et qui produit une représentation par listes de successeurs du graphe $G \stackrel{\text{def}}{=} (S, A)$ où $S \stackrel{\text{def}}{=} \{1, \cdots, n\}$ et A est la collection des arcs (u, v) apparaissant dans L. On notera `succ` le tableau tel que `succ` $[u]$ contienne la liste des successeurs de u (*sans doublon*) dans G à la fin de l'algorithme.

La **Question 2** sera utile dans la suite.

2 Calcul de $G^{\geq 2}$

Pour tout graphe orienté $G \stackrel{\text{def}}{=} (S, A)$, on note $G^{\geq 2}$ le graphe dont l'ensemble des sommets est S et tel qu'il existe un arc de u à w dans $G^{\geq 2}$ si et seulement s'il existe un chemin de longueur ≥ 2 de u à w dans G.

Question 3 ▮▯▯▯▯ Montrer que, si G est sans circuit, alors il existe un arc (u, w) dans $G^{\geq 2}$ si et seulement s'il existe un sommet $v \neq u$ dans G, un chemin $u \to^* v$ dans G, et un arc $v \to w$ dans G.

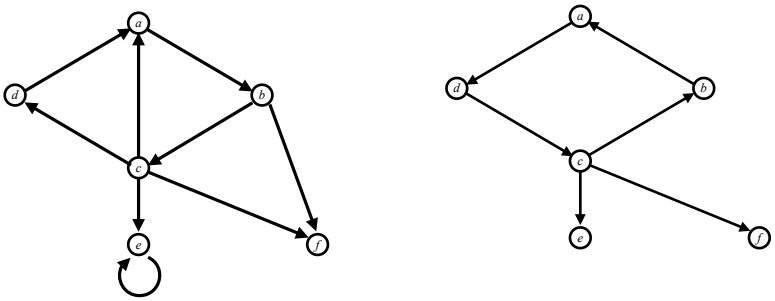
Question 4 ▮▮▯▯▯ Décrire un algorithme qui, étant donné un graphe orienté sans circuit G en entrée, calcule une liste L des arcs de $G^{\geq 2}$, en temps $O(n(n+m))$. L peut contenir des doublons, mais sera de longueur au plus n^2. [Erratum : au plus nm.] On pourra procéder par modification d'un algorithme (lequel ?) du cours.

Question 5 ▮▯▯▯▯ En déduire un algorithme en temps $O(n(n+m))$ calculant $G^{\geq 2}$ (sous forme de listes de successeurs) en fonction de G, lorsque G est sans circuit.

3 Réduction transitive I : le cas sans circuit

Un *équivalent* d'un graphe orienté $G \stackrel{\text{def}}{=} (S, \to)$ est un graphe $G' \stackrel{\text{def}}{=} (S, \to')$ de mêmes sommets que G et dont la relation d'accessibilité $\to'^*$ est identique à celle, $\to^*$, de G. Autrement dit, pour tout arc $u \to v$ dans G, il existe un chemin de u à v dans G', et pour tout arc de u à v dans G', il existe un chemin de u à v dans G.

On ne demande pas que G' soit un sous-graphe de G. Dans l'exemple ci-dessous, le graphe de droite est un équivalent du graphe de gauche :



Une *réduction transitive* de G est un équivalent G' de G dont aucun sous-graphe strict n'est un équivalent. (L'exemple ci-dessus est une réduction transitive.) Autrement dit, G' est un équivalent de G, mais enlever ne serait-ce qu'un arc de G produit un graphe qui n'est pas un équivalent de G. Tout graphe a (au moins) une réduction transitive.

On commence par regarder le cas des graphes sans circuit.

Question 6 ▮▮▮▯▯ Soit $G \stackrel{\text{def}}{=} (S, \to)$ un graphe orienté sans circuit, et soit G' le sous-graphe couvrant $(S, \to')$ où $u \to' v$ si et seulement $u \to v$ dans G

et il n'y a aucun chemin de longueur supérieure ou égale à 2 de u à v dans G. Montrer que G' est un équivalent de G.

Je ne veux pas d'algorithme ici. La méthode que vous emploierez pour la direction difficile est la suivante. Appelons un arc $u \to v$ de G *mauvais* s'il n'y a pas de chemin $u \to'^* v$ dans G'. S'il existe un mauvais arc $u \to v$ dans G, il y en a un qui minimise $(rto(u), rto(v))$ [Erratum : $(rto(u), n - rto(v))$, n étant le nombre de sommets de G] dans l'ordre lexicographique, où rto est un tri topologique inverse de G. Conclure.

Question 7 ▮▯▯▯▯ Montrer que le graphe G' de la **Question 6** est une réduction transitive de G.

Question 8 ▮▮▮▯▯ Décrire un algorithme qui prend un graphe orienté sans circuit G en entrée et retourne la réduction transitive G' de G décrite à la **Question 6**. Les graphes sont représentés, comme d'habitude, par listes de successeurs. Donner la complexité de l'algorithme ; pour espérer avoir tous les points, cette complexité devra être en $O(n(n+m))$.

4 Réduction transitive II : le cas général

On passe aux graphes généraux, pas nécessairement sans circuit.

Étant donné un graphe orienté G, on forme sa condensation, que l'on notera $\Gamma(G)$.

Question 9 ▮▮▯▯▯ Donner un algorithme qui prend un graphe orienté G en entrée, et calcule $\Gamma(G)$. On suppose que G est représenté par listes de successeurs, et que l'on a aussi en entrée des tableaux et `r`, `low` et `c` donnant respectivement pour chaque sommet u son rang $r(u)$, le rang $low(u)$ de son point d'attache pour un parcours en profondeur donné, et le point d'entrée $*_{[u]}$ de sa composante fortement connexe $[u]$. On demande que $\Gamma(G)$ soit représentée par listes de successeurs. Les sommets C de $\Gamma(G)$ seront représentés par leurs points d'entrée $*_C$; donc, en principe, $\Gamma(G)$ devrait être décrit par un tableau `succ` de listes `succ` $[u]$, un par sommet u de la forme $*_C$, mais on demandera à la place que `succ` soit indexé par les sommets u de G, les listes `succ` $[u]$ étant vides lorsque u n'est pas un point d'entrée. On rappelle que dans une représentation par liste de successeurs, `succ` $[u]$ ne contient pas de doublon. Donner la complexité de votre algorithme : $O(n^2+m)$ n'est pas mal, mais pour avoir tous les points, il faudra que votre algorithme (soit correct et) ait

une complexité en $O(n+m)$ au pire.

On rappelle que la condensation d'un graphe est toujours sans circuit, et on peut donc former sa réduction transitive $\Gamma(G)'$ comme à la **Question 6**. On utilise tout cela pour définir un graphe orienté $G_\circ$ comme suit.

- Les sommets de $G_\circ$ sont ceux de G.
- Pour chaque composante fortement connexe C de G contenant au moins deux sommets, on en choisit une énumération $\{x_1, \cdots, x_n\}$ ($n \geq 2$), et $G_\circ$ contient les arcs (x_1, x_2), (x_2, x_3), …, (x_{n-1}, x_n) et (x_n, x_1), formant ainsi un circuit $\gamma_C \stackrel{\text{def}}{=} (x_0 \to x_1 \to \cdots \to x_n)$ avec $x_0 \stackrel{\text{def}}{=} x_n$.
- Pour tout arc $C_1 \to C_2$ dans $\Gamma(G)'$, on choisit un arc — que nous noterons $\gamma_{C_1 \to C_2}$ dans la suite — d'un sommet de C_1 à un sommet de C_2 dans G, qu'on ajoute à $G_\circ$.
- $G_\circ$ ne contient aucun autre arc.

On admettra que $G_\circ$ est une réduction transitive de G.

Question 10 ▮▮▮▮▮ Décrire un algorithme le plus efficace possible calculant $G_\circ$ en fonction de G, et donner sa complexité. Les graphes sont toujours sous forme de listes de successeurs.

5 Arrondir les matrices

Etant donnée une matrice $A = (a_{ij})_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}}$ de nombres réels de dimension $p \times q$, un *arrondi* de A est une matrice $B = (b_{ij})_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}}$ telle que :

1. les valeurs b_{ij} sont des entiers (dans $\mathbb{Z}$) ;
2. pour tous i et j, b_{ij} est soit la partie entière inférieure $\lfloor a_{ij} \rfloor$ soit la partie entière supérieure $\lceil a_{ij} \rceil = -\lfloor -a_{ij} \rfloor$ de a_{ij} ;
3. la somme de chaque ligne est inchangée : pour tout $i \in \{1, \cdots, p\}$, $\sum_{j=1}^{q} b_{ij} = \sum_{j=1}^{q} a_{ij}$;
4. la somme de chaque colonne est inchangée : pour tout $j \in \{1, \cdots, q\}$, $\sum_{i=1}^{p} b_{ij} = \sum_{i=1}^{p} a_{ij}$.

Par exemple, avec :

$$A = \begin{pmatrix} 1,2 & 3,4 & 2,4 \\ 3,9 & 4,0 & 2,1 \\ 7,9 & 1,6 & 0,5 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{pmatrix}$$

B est un arrondi de A.

Le problème du calcul d'un arrondi de A se réduit au sous-problème **Arrondi** suivant :

Entrée : une matrice $M = (m_{ij})_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}}$ de nombres dans $[0, 1]$;

Sortie : un arrondi de M, ou bien $\perp$ s'il n'en existe pas.

En effet, étant donné A, on forme M en posant $m_{ij} \stackrel{\text{def}}{=} a_{ij} - \lfloor a_{ij} \rfloor$, on résout **Arrondi**, et si M a un arrondi $P = (p_{ij})_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}}$, alors en posant $b_{ij} \stackrel{\text{def}}{=} p_{ij} + \lfloor a_{ij} \rfloor$ pour tous i et j, on obtient un arrondi de A ; et A n'en a pas si M n'a pas d'arrondi.

On notera qu'un arrondi de M est une matrice dont les entrées valent toutes soit 0 soit 1.

À partir de M comme entrée de **Arrondi**, on forme un réseau G_M comme suit. En plus d'une source s et d'un puits p, on crée un sommet L_i par ligne ($1 \leq i \leq p$) et un sommet C_j par colonne ($1 \leq j \leq q$), et d'arcs $s \to L_i \to C_j \to p$. La capacité des arcs $L_i \to C_j$ vaut 1, celle des arcs $s \to L_i$ vaut $\sum_{j=1}^{q} m_{ij}$, et celle des arcs $C_j \to p$ vaut $\sum_{i=1}^{p} m_{ij}$.

Question 11 ▮▮▯▯▯ Exhiber un flot, construit à l'aide des valeurs m_{ij}, et de valeur $\sum_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}} m_{ij}$. En exhibant une coupe bien choisie, en déduire que la valeur maximale d'un flot sur G_M est exactement $\sum_{\substack{1 \leq i \leq p \\ 1 \leq j \leq q}} m_{ij}$.

Question 12 ▮▮▮▮▯ Il est clair qu'une entrée M de **Arrondi** ne peut avoir d'arrondi que si les sommes $\sum_{i=1}^{p} m_{ij}$ ($1 \leq j \leq q$) et $\sum_{j=1}^{q} m_{ij}$ ($1 \leq i \leq p$) sont toutes entières. Réciproquement, si toutes ces sommes sont entières, montrer que M a toujours un arrondi, qui est calculable grâce à un algorithme vu en cours, que l'on citera. En déduire un algorithme qui résout **Arrondi**. Quelle est sa complexité, lorsque $p = q$ tend vers $+\infty$?