Master 2 Informatique et Interactions Advanced graphs

Examen, Octobre 2022

1 heure 30.

Aucun document n'est autorisé. Pas d'ordinateurs, ni de téléphones portables.

Instructions et commentaires : les points attribués pour votre réponse seront basés sur la justesse de votre réponse ainsi que sur la clarté des principales étapes de votre raisonnement. Toutes les solutions proposées doivent être prouvées. **Tous les exercices sont indépendants.** *Il est également conseillé de lire tout l'énoncé avant de commencer, mais il semble que les élèves ne lisent jamais cette phrase... Les points et durées approximatifs sont indiqués pour que pour que vous puissiez adapter votre effort.*

Exercice 1 (Un peu de graphes planaires. 6 points, 20 minutes)
Pour tout graphe connexe simple et **planaire** $G = (V, E)$, on note $n = |V|$ son nombre de sommets, $m = |E|$ son nombre d'arêtes et f son nombre de faces.

Soit $k \in \mathbb{N}$. Une **k-coloration** de G est une fonction $c : V \to \{1, \cdots, k\}$ qui affecte une couleur (un entier) à chaque sommet. Une k-coloration est **propre** si deux sommets adjacents reçoivent des couleurs différentes, i.e., pour tout $\{u, v\} \in E$, $c(u) \neq c(v)$. Le **nombre chromatique** $\chi(G)$ est le plus petit entier k tel que G admette une k-coloration propre.

La **maille** $g(G)$ (g pour "girth" en anglais) est la longueur d'un plus petit cycle de G. Si G est acyclique, on convient que $g(G) = \infty$.

1. Soit $v \in V$ un sommet. Montrer que $g(G \setminus v) \geq g(G)$ où $G \setminus v$ est le graphe obtenu de G en supprimant v.
2. Montrer que toute face de G a longueur au moins $g(G)$.
3. Rappeler la formule d'Euler en fonction de n, m et f.
4. Donner une borne supérieure de m en fonction de n si $g(G) \geq 4$.
5. En déduire que tout graphe planaire de maille $g(G) \geq 4$ a un sommet de degré au plus 3.
6. En déduire que, pour tout graphe planaire sans triangles, $\chi(G) \leq 4$.

Exercice 2 (Un peu de programmation linéaire. 6 points, 20 minutes)
Soit $G = (V, E)$ un graphe connexe de n sommets et m arêtes et une fonction $w : E \to \mathbb{R}^+$ de poids sur les arêtes. Soit $s \in V$. On rappelle que, pour tout sommet $v \in V$, $N(v)$ est l'ensemble des voisins de v.

Une arête entre deux sommets i et j est notée $\{i, j\} \in E$ (l'arête $\{i, j\}$ est un ensemble de deux sommets, l'ordre des éléments n'est pas important). Lorsqu'il y a des parenthèses (au lieu des accolades), on parle de couples de sommets et l'ordre est important. Par exemple, pour tous $i, j \in V$, $\{i, j\} = \{j, i\}$, alors que $(i, j) \neq (j, i)$ lorsque $i \neq j$.

On considère le programme linéaire suivant :

Minimiser
$$\sum_{e \in E} w(\{i, j\}) * y_{\{i,j\}}$$

Sous les contraintes :
$$f_{(i,j)} + f_{(j,i)} \leq n * y_{\{i,j\}} \qquad \forall \{i, j\} \in E$$
$$\sum_{u \in N(s)} f_{(s,u)} = n - 1$$
$$\sum_{u \in N(v)} f_{(u,v)} - \sum_{u \in N(v)} f_{(v,u)} = 1 \quad \forall v \in V \setminus \{s\}$$

Domaine de définition :
$$y_{\{i,j\}} \in \{0, 1\} \qquad \forall \{i, j\} \in E$$
$$f_{(i,j)} \geq 0 \qquad \forall \{i, j\} \in E$$
$$f_{(j,i)} \geq 0 \qquad \forall \{i, j\} \in E$$

1. Donner le nombre de variables et de contraintes en fonction de m. Quelle est la différence majeure entre les variables y et les variables f ? Pourquoi cette différence est-elle importante ?

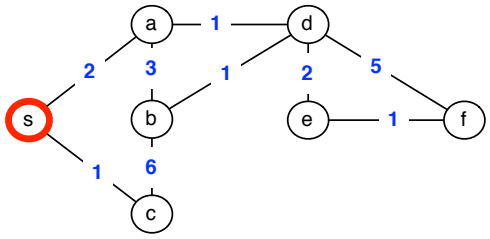

Figure 1: Un graphe $G = (V, E)$ avec $V = \{s, a, b, c, d, e, f\}$ et les poids des arêtes indiqués en bleu.

2. Donner une solution valide à ce programme dans le cas de l'exemple de la Figure 1. C'est-à-dire, donner une affectation à chaque variable qui satisfait les contraintes (sans être forcément une solution optimale). *On pourra se restreindre à des valeurs entières pour les variables f.*

   Donner votre solution ici :

$$
\begin{array}{lll}
y_{\{s,a\}} = & f_{(s,a)} = & f_{(a,s)} = \\
y_{\{s,c\}} = & f_{(s,c)} = & f_{(c,s)} = \\
y_{\{a,b\}} = & f_{(a,b)} = & f_{(b,a)} = \\
y_{\{a,d\}} = & f_{(a,d)} = & f_{(d,a)} = \\
y_{\{b,d\}} = & f_{(b,d)} = & f_{(b,d)} = \\
y_{\{b,c\}} = & f_{(b,c)} = & f_{(c,b)} = \\
y_{\{d,e\}} = & f_{(d,e)} = & f_{(e,d)} = \\
y_{\{d,f\}} = & f_{(d,f)} = & f_{(f,d)} = \\
y_{\{e,f\}} = & f_{(e,f)} = & f_{(f,e)} =
\end{array}
$$

   Que vaut la valeur de la fonction objectif pour votre solution ?

3. Que calcule le programme linéaire ? Expliquer le rôle de chaque variable et de chaque contrainte.

   Astuce : rappelons que pour un flot dans un graphe non orienté, la contrainte de capacité est que la somme des flots dans les deux sens sur une arête doit être inférieure à la capacité de cette arête.

Exercice 3 (Un peu d'algorithme pour calculer les distances. 7 points, 25 minutes)

Soit $G = (V, E)$ un graphe avec une fonction de longueur $\ell : E \to \mathbb{R}^+$ sur les arêtes. Un **chemin** entre $u \in V$ et $v \in V$ dans G est une séquence $P = (u = u_0, u_1, \cdots, u_q = v)$ de sommets distincts telle que $\{u_{i-1}, u_i\} \in E$ pour tout $1 \leq i \leq q$. La **longueur** d'un chemin $(u = u_0, u_1, \cdots, u_q = v)$ entre deux sommets u et v est $\sum_{1 \leq i \leq q} \ell(\{u_{i-1}, u_i\})$. La **distance** $dist(u, v)$ entre deux sommets u et v est la longueur minimum d'un chemin entre u et v.

1. Donner un algorithme qui prend en entrée un graphe $G = (V, E)$ et $\ell : E \to \mathbb{R}^+$, et retourne, pour tous $u, v \in V$, la distance entre u et v. Donner la complexité de votre algorithme en fonction de $|V|$ et $|E|$.

   Astuce : On pourra utiliser l'algorithme de Dijkstra comme sous-procédure.

Étant donné un graphe $G = (V, E)$, **identifier** deux sommets $u, v \in V$ dans G signifie informellement "fusionner" ces deux sommets en un unique sommet w_{uv}. Formellement, le graphe $G' = (V', E')$ obtenu de G après identification de u et v est défini par $V' = (V \setminus \{u, v\}) \cup \{w_{uv}\}$ et $E' = (E \setminus (\{\{u, x\} \in E \mid x \in N(u)\} \cup \{\{v, x\} \in E \mid x \in N(v)\})) \cup \{\{w_{uv}, x\} \mid x \in N(u) \cup N(v)\}$. Voir des exemples sur la Figure 2. Si ce n'est pas clair, demandez moi.

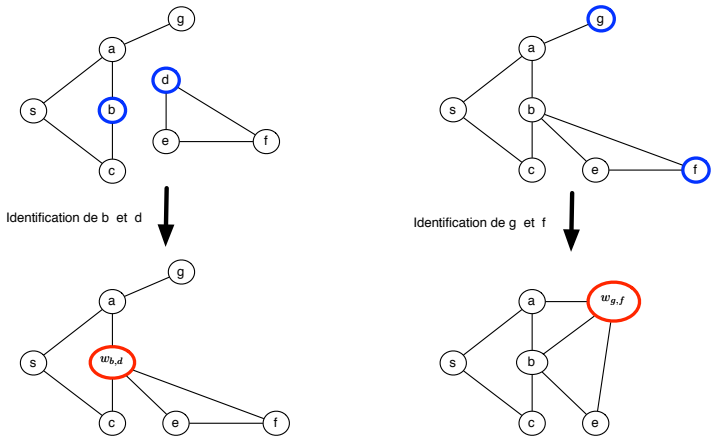

Figure 2: Exemples d'identification de deux sommets dans des graphes.

La classe des graphes **série-parallèles** est obtenue de manière récursive:

Initialisation: Le graphe (G, s, p) avec uniquement 2 sommets s et p et une arête $\{s, p\}$ est un graphe série-parallèle avec source s et puit p.

Composition série: Étant donnés deux graphes série-parallèles (G_1, s_1, p_1) et (G_2, s_2, p_2), le graphe (G, s_1, p_2) obtenu en identifiant p_1 et s_2 est un nouveau graphe série-parallèle avec source s_1 et puit p_2.

Composition parallèle: Étant donnés deux graphes série-parallèles (G_1, s_1, p_1) et (G_2, s_2, p_2), le graphe $(G, s_1 = s_2 = s, p_1 = p_2 = p)$ obtenu en identifiant d'une part s_1 et s_2, et d'autre part p_1 et p_2, est un nouveau graphe série-parallèle avec source s et puit p.

Sur l'exemple de la Figure 3, le graphe G (en haut) est obtenu des graphes G_1 à G_7 (qui sont des arêtes) en mettant G_1 et G_2 en série (soit G_{12} le graphe obtenu), puis G_3 en parallèle avec G_{12} (soit G_{123} le graphe obtenu) et G_4 en série avec G_{123} (soit G_{1234} le graphe obtenu avec source s_1 et puit p_4). Par ailleurs, on met G_5 et G_6 en série (obtenant le graphe G_{56}) et G_{56} en série avec G_7 (soit G_{567} le graphe obtenu avec source s_5 et puit p_7). Le graphe G est enfin obtenu en mettant G_{1234} et G_{567} en parallèle.

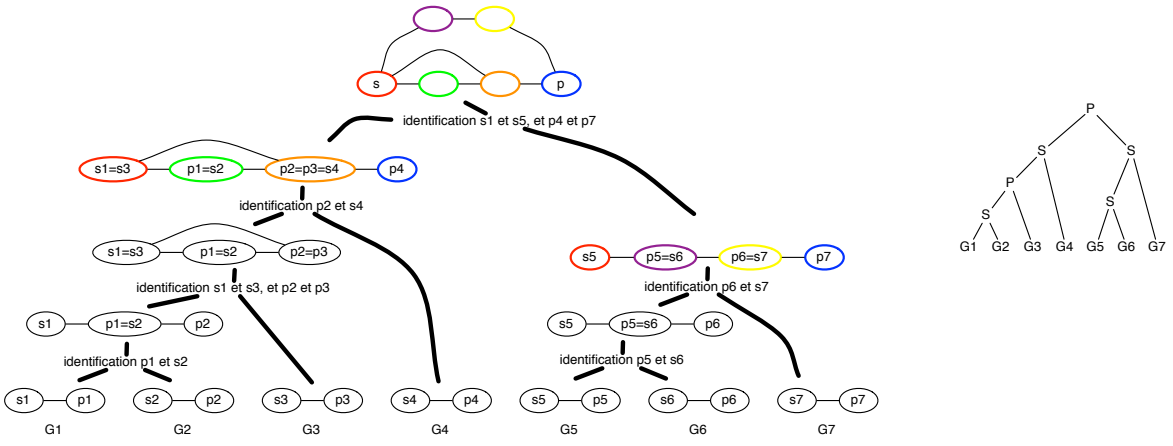

Figure 3: Exemple de construction récursive d'un graphe série-parallèle.

Sur la droite de la Figure 3 se trouve l'**arbre de construction** du graphe G où les noeuds étiquetés "S" correspondent à des compositions séries, et les noeuds étiquetés "P" correspondent à des compositions parallèles. On admet le théorème suivant.

Théorème 1 *Étant donné un graphe série-parallèle de n sommets, son arbre de construction peut être calculé en temps $O(n)$.*

Soit $G = (V, E)$ un graphe et $\ell : E(G) \to \mathbb{R}^+$ et $W \subseteq V$ et $u, v \in W$. On note $dist_W(u, v)$ la distance entre u et v dans W, c'est-à-dire, $dist_W(u, v)$ est la longueur minimum d'un chemin entre u et v dans le sous graphe de G induit par les sommets de W, c'est-à-dire, un chemin qui ne "passe" que par des sommets de W.

2. Prouver que tout graphe série-parallèle est planaire (on n'attend pas une preuve rigoureuse, mais la "moins informelle" possible... un dessin peut aider).

3. Soit $G(s_1, p_2)$ le graphe obtenu par composition série de 2 graphes série-parallèles (G_1, s_1, p_1) et (G_2, s_2, p_2). Soit $u \in V(G_1)$:

   - Pour $v \in V(G_1)$, prouver que $dist_{V(G)}(u, v) = dist_{V(G_1)}(u, v)$.
   - Pour $v \in V(G_2)$, prouver que $dist_{V(G)}(u, v) = dist_{V(G_1)}(u, p_1) + dist_{V(G_2)}(s_2, v)$.

4. Soit $G(s, p)$ le graphe obtenu par composition parallèle de 2 graphes série-parallèles (G_1, s_1, p_1) et (G_2, s_2, p_2). Soit $u \in V(G_1)$:

   - Pour $v \in V(G_1)$, exprimer $dist_{V(G)}(u, v)$ en fonction de $dist_{V(G_1)}(u, v)$, $dist_{V(G_1)}(u, s_1)$, $dist_{V(G_1)}(u, p_1)$, $dist_{V(G_1)}(v, s_1)$, $dist_{V(G_1)}(v, p_1)$ et $dist_{V(G_2)}(s_2, p_2)$.
   - Pour $v \in V(G_2)$, exprimer $dist_{V(G)}(u, v)$ en fonction de $dist_{V(G_1)}(u, s_1)$, $dist_{V(G_1)}(u, p_1)$, $dist_{V(G_2)}(v, s_2)$, et $dist_{V(G_2)}(v, p_2)$.

5. Déduire des questions précédentes un algorithme qui prend en entrée un graphe série-parallèle $G = (V, E)$, $\ell : E \to \mathbb{R}^+$, et calcule, pour tout $u, v \in V$, $dist_{V(G)}(v, u)$. Donner la complexité temporelle de votre algorithme. Comparer avec la complexité de l'algorithme que vous avez proposé à la question 1.

Exercice 4 (Un peu de diamètre. 7 points, 25 minutes)
Étant donné un graphe $G = (V, E)$, un **chemin** entre $u \in V$ et $v \in V$ dans G est une séquence $P = (u = u_0, u_1, \cdots, u_\ell = v)$ de sommets distincts telle que $\{u_{i-1}, u_i\} \in E$ pour tout $1 \leq i \leq \ell$. La longueur de P est son nombre d'arêtes ℓ. La **distance** $dist(u, v)$ entre u et v est la longueur minimum d'un chemin entre u et v dans G. L'**eccentricité** $ecc(v)$ d'un sommet $v \in V$ est la plus grande distance de v à un autre sommet, i.e., $ecc(v) = \max_{u \in V} dist(u, v)$. Le **diamètre** $diam(G)$ d'un graphe G est la plus grande distance entre deux sommets de G, i.e., $diam(G) = \max_{v \in V} ecc(v) = \max_{u,v \in V} dist(u, v)$.

1. Proposer un algorithme qui prend une entrée un graphe $G = (V, E)$ et $u \in V$ et calcule $ecc(u)$ en temps linéaire.

2. Soient $G = (V, E)$ un graphe et $u \in V$. Montrer que $ecc(u) \leq diam(G) \leq 2 * ecc(u)$.

3. Soient $G = (V, E)$ un graphe, $u \in V$ et, pour tout $1 \leq i \leq ecc(u)$, soit $L_i = \{v \in V \mid dist(u, v) = i\}$. Soit $1 \leq i \leq ecc(u)$ et supposons que, pour tout $v \in \bigcup_{i \leq j \leq ecc(u)} L_j$, $ecc(v) \leq 2(i-1)$.

   Montrer que $diam(G) \leq 2(i-1)$.

   Astuce : Prenez deux sommets $x, y \in V$, montrez que $dist(x, y) \leq 2(i-1)$. Il faut distinguer les cas selon que x, y, les deux ou aucun appartiennent à $\bigcup_{i \leq j \leq ecc(u)} L_j$.

4. Appliquer l'algorithme 1 sur le graphe de la Figure 4 depuis le sommet U. Quelle valeur retourne-t-il ? En particulier, indiquer l'évolution des variables $BorneInf$ et $BorneSup$ et les sommets considérés lorsque leurs valeurs sont modifiées. Préciser également les ensembles L_i et $ecc(U)$.

5. Quelle est la complexité temporelle de l'algorithme 1 en fonction de n ? (expliquez votre réponse)

6. Que calcule l'algorithme 1 ? Prouvez le en utilisant les questions précédentes.

   On pourra montrer qu'à la fin de l'itération i de la première boucle for, $BorneInf = \max_{v \in \{u\} \cup \bigcup_{i \leq j \leq ecc(u)} L_j} ecc(v)$.

Pour info : Dans la pratique, l'algorithme 1 est bien plus efficace que ce que sa complexité en pire cas suggère. En particulier, il a été appliqué avec succès sur plusieurs très (très) grands réseaux réels.

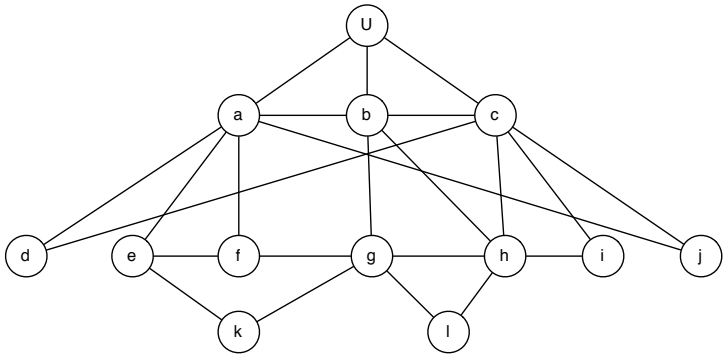

Figure 4: Appliquer l'algorithme 1 sur ce graphe en partant de U.

Algorithm 1

Require: Un graphe $G = (V, E)$ et un sommet $u \in V$.
Ensure: ???

Faire un parcours en largeur (BFS) depuis u et, pour tout $1 \leq i \leq ecc(u)$, soit $L_i = \{v \in V \mid dist(u, v) = i\}$;
BorneSup $\leftarrow 2 * ecc(u)$;
BorneInf $\leftarrow ecc(u)$;
for $i = ecc(u)$ jusqu'à 1 **do**
  for $x \in L_i$ **do**
    Faire un parcours en largeur (BFS) depuis x
    if $ecc(x) > BorneInf$ **then**
      $BorneInf \leftarrow ecc(x)$;
    if $BorneInf = BorneSup$ **then**
      return $BorneInf$
  if $BorneInf = BorneSup - 1$ **then**
    return $BorneInf$
  else
    $BorneSup \leftarrow BorneSup - 2$