Graphes et algorithmes
Exercices

Denis Lapoire

25 janvier 2012

Table des matières

Chapitre 1

Graphes notions générales

Exercice 1 (Degré) Dessiner si c'est possible un graphe qui a 3 sommets de degré 3 et 4 sommets de degré 2. Le cas échéant, expliquer pourquoi ce n'est pas possible.

Exercice 2 (Degré) Soit G un graphe simple sans boucle non orienté ayant n sommets. Montrer que le degré de tout sommet s de G est inférieur ou égal à $n-1$, et que G ne peut contenir à la fois un sommet de degré 0 et un sommet de degré $n-1$; en déduire que G admet au moins deux sommets de même degré.

Exercice 3 (Petersen) Soit $X = \{0, 1, 2, 3, 4\}$. Le *graphe de Petersen* est le graphe non orienté défini de la façon suivante : ses sommets sont les paires d'éléments de X, et deux sommets sont joints par une arête si et seulement si ce sont deux paires disjointes d'éléments de X. Déterminer le nombre de sommets du graphe, le degré de chaque sommet, le nombre d'arêtes du graphe ; dessiner le graphe.

Exercice 4 Deux joueurs disposent de 2 ou plusieurs tas d'allumettes. À tour de rôle, chaque joueur peut enlever un certain nombre d'allumettes de l'un des tas (selon la règle choisie). Le joueur qui retire la dernière allumette perd la partie.

1. Modéliser ce jeu à l'aide d'un graphe dans le cas où l'on dispose au départ de deux tas contenant chacun trois allumettes, et où un joueur peut enlever une ou deux allumettes à chaque fois.
2. Que doit jouer le premier joueur pour gagner la partie à coup sûr ?

Exercice 5 Tante Yvonne possède trois récipients de capacités respectives 8, 5 et 3 litres. Initialement le récipient de 8 libres est plein. Elle souhaite isoler 4 litres dans deux des récipients. Est-ce possible ? Si oui, comment ? Quelle est la méthode la plus rapide ? la moins fatigante ?

Exercice 6 (Chemin) Un ami vous propose de définir dans un graphe orienté un chemin élémentaire comme un simple ensemble d'arcs vérifiant quelques propriétés supplémentaires. Qu'en pensez-vous ?

Exercice 7 (Énumération) Nous considérons ici les graphes ayant pour ensemble de sommets des intervalles d'entiers de la forme $[1, n]$ avec $n \geq 1$.

1. Quel est le nombre de graphes simples orientés ayant n sommets ?
2. Quel est le nombre de graphes simples non orientés ayant n sommets ?
3. Mêmes questions que celles précédentes pour des graphes sans boucles.

Exercice 8 (Isomorphisme) On considère ici des graphes simples orientés ou non.

1. Dessiner l'ensemble des graphes orientés ayant pour ensemble de sommets l'intervalle $[1, 4]$ et isomorphes au circuit $([1, 4], \{(1, 2), (2, 3), (3, 4), (4, 1)\})$.
2. Dessiner l'ensemble des graphes non orientés ayant pour ensemble de sommets l'intervalle $[1, 4]$ et isomorphes à la chaîne $([1, 4], \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}\})$.
3. Pour tout $n \geq 1$, compter le nombre de graphes orientés (resp. non orientés) ayant pour ensemble de sommets $[1, n]$ et isomorphe au circuit $([1, n], \{(1, 2), \ldots, (n-1, n), (n, 1)\})$ (resp. à la chaîne $([1, n], \{\{1, 2\}, \ldots, \{n-1, n\}, \{n, 1\}\})$).
4. Écrire un algorithme décidant si deux graphes fournis en entrée sont isomorphes ou non. Évaluer sa complexité en temps et en espace. Vous utiliserez un type `renommage` contenant notamment les primitives suivantes donct on supposera la complexité en temps 0(1) :
   - `fonction renommageSuivant(r:renommage):booléen` qui retourne le booléen vrai si il existe une bijection suivante selon l'ordre $<_n$ et retourne celle-ci par effet de bord sur `r`. existe une bijection suivante selon l'ordre $<_n$.
   - `procédure initialisation(n : entier)` qui retourne la première bijection selon $<_n$.

   où $<_n$ est un ordre total sur l'ensemble des bijections sur $[1, n] \to [1, n]$.

L'algorithme que vous écrirez sera indépendant de toute implémentation de graphes particulière et utilisera des fonctions auxiliaires pour lesquelles vous définierez précisément ce qu'elles font.

Exercice 9 Quels sont les dessins ci-dessous qui représentent le graphe de Petersen ?

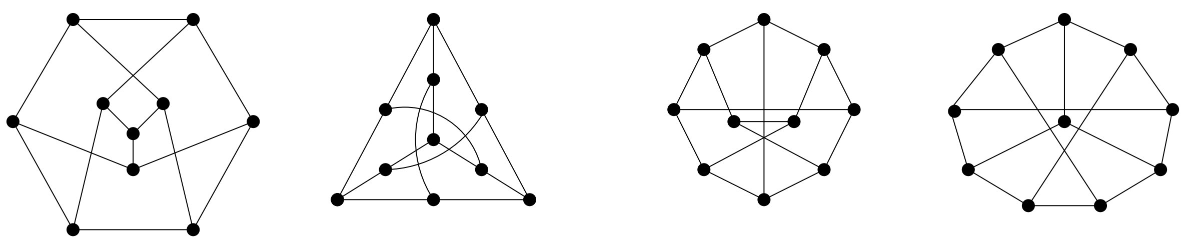

Fig. 1.1 – Quelques graphes

Chapitre 2

Implémentation de graphes

Exercice 10 (Implémentation) Nous considérons ici les graphes ayant pour ensemble de sommets des intervalles d'entiers de la forme $[1, n]$ avec $n \geq 0$. Décrire une implémentation de ces graphes comprenant notamment les primitives usuelles en utilisant :

1. une matrice de booléens de taille $n \times n$.
2. un tableau de n listes de sommets.

Exercice 11 (Implémentation) Même question que dans l'exercise précédent dans le cas de graphes où l'ensemble des sommets est un ensemble d'entiers qui n'est pas nécessairement un intervalle.

Exercice 12 (Graphe complémentaire) Le graphe *complémentaire* d'un graphe simple orienté (V, E) est le graphe $(V, V \times V - E)$. Selon que le graphe est représenté par sa matrice d'adjacence ou par des listes de successeurs, écrire un algorithme calculant le complémentaire de tout graphe. Évaluer sa complexité.

Exercice 13 (Graphe quotient) Écrire un algorithme calculant le quotient d'un graphe simple orienté selon une relation d'équivalence sur l'ensemble de ses sommets. Cette relation est fournie par un tableau qui associe à chaque sommet son représentant dans la classe. Selon que le graphe est représenté par sa matrice d'adjacence ou par des listes de successeurs, écrire un algorithme calculant le complémentaire de tout graphe. Évaluer sa complexité.

Exercice 14 (Puits dans un graphe) Un sommet s d'un graphe orienté simple (V, E) est un *puits* si pour tout sommet t distinct de s, l'arc (t, s) appartient à E mais pas l'arc (s, t).

1. Montrer que tout graphe a au plus un puits.
2. Selon que le graphe est représenté par sa matrice d'adjacence ou par des listes de successeurs, écrire une fonction qui décide l'existence d'un puits dans un graphe. Évaluer sa complexité.

Chapitre 3

Parcours en largeur et en profondeur

Nous notons $V(G)$ l'ensemble de sommets d'un graphe donné G. Nous supposons en outre qu'un tel ensemble est de la forme un intervalle d'entiers $[1, n]$ avec $n \geq 1$.

Exercice 15 Une *k-coloration* d'un graphe G est une fonction $V(G) \to [1, k]$ telle que toute paire de sommets adjacents (et distincts) sont coloriés différemment (c'est à dire associés par f à deux entiers distincts). Un graphe est *biparti* si il admet une 2-coloration.

1. Écrire un algorithme qui décide si un graphe non orienté connexe est biparti.
2. Écrire un algorithme qui décide si un graphe non orienté est biparti.

Exercice 16 Écrire un algorithme décidant si un graphe admet une 3-coloration. Évaluer sa complexité.

Exercice 17 Écrire l'algorithme de parcours en profondeur sans utiliser aucun appel récursif.

Exercice 18 Ecrire un algorithme décidant si un graphe orienté est acyclique.

Exercice 19 Un *tri topologique* d'un graphe orienté G est une numérotation des sommets $f : V(G) \to [1, n]$ telle que pour tout arc (s, t) de G on ait $f(s) < f(t)$.

1. Démontrer qu'un graphe orienté est acyclique si et seulement si il admet un tri topologique.
2. Écrire un algorithme qui calcule pour tout graphe orienté un tri topologique.

Exercice 20 Pour tout graphe orienté et acyclique et pour tout couple de sommets s et t, on souhaite calculer un ensemble L de chemins de s à t deux à deux arc-disjoints et maximal selon l'inclusion ; c'est à dire : L est tel que tout chemin $\notin L$ de s à t contient nécessairement un arc d'un des chemins de L. Pour ce faire, vous utiliserez un parcours en profondeur du graphe.

Exercice 21 (Un peu de mythologie) Thésée, après avoir tué le Minotaure, s'aperçoit avec horreur que le fil qu'Ariane lui avait confié était bien trop court : il est absolument perdu.

Fouillant ses poches, il n'y découvre que des morceaux de craie de différentes couleurs.

Le Labyrinthe est un inextricable fouillis de salles reliées entre elles par des couloirs tortueux. La plupart des salles sont quasiment indistinguables les unes des autres – la sortie mise à part.

- Décrire un algorithme permettant au héros de sortir de ce mauvais pas.
- Estimer la quantité de craie dont il a besoin.

Exercice 22 Le but de cet exercice est de démontrer la correction de l'algorithme vu en cours pour déterminer les composantes fortement connexes d'un graphe orienté simple : effectuer d'abord un parcours en profondeur du graphe, puis un parcours en profondeur du graphe transposé, avec la contrainte que, dans la boucle principale, le sommet à visiter doit être, parmi les sommets encore "blancs", celui qui a obtenu la plus grande date de fin de visite lors du premier parcours.

G désigne un graphe orienté simple, et ${}^tG$ son graphe transposé ; on notera $x \sim y$ lorsque x et y sont deux sommets appartenant à la même composante fortement connexe.

1. Montrer que G et ${}^tG$ ont les mêmes composantes fortement connexes.
2. Montrer que, pour tout parcours en profondeur de G, chaque composante fortement connexe de G est incluse dans une même arborescence du graphe de liaison.
3. Dans la suite, on suppose fixé un parcours en profondeur de G ; pour tout sommet x, $f(x)$ désigne la date de fin de visite de x. On définit pour chaque sommet x, un sommet $\Phi(x)$ de la manière suivante : $\Phi(x)$ *est, parmi les sommets de G accessibles à partir de x, celui qui maximise f.*
   (a) Que dire de $\Phi(\Phi(x))$?
   (b) Montrer que $\Phi(x)$ est un ascendant de x dans le graphe de liaison.
   (c) Montrer que $x \sim y$ si et seulement si $\Phi(x) = \Phi(y)$.
4. Montrer, par récurrence, que chaque arborescence du parcours de ${}^tG$ est une composante fortement connexe de G, et que sa racine est l'image par Φ de chacun de ses sommets.

Chapitre 4

Arbres

Exercice 23 Démontrer que pour tout graphe non orienté $G = (V, E, f)$, les assertions suivantes sont équivalentes :

1. G est un arbre (c.a.d connexe et acyclique).
2. G est connexe et est minimal à vérifier cette propriété (c.a.d la suppression d'une arête quelconque déconnecte G).
3. G est connexe et vérifie $|V| \geq |E| + 1$.
4. G est acyclique et est maximal à vérifier cette propriété (c.a.d l'ajout de toute nouvelle arête permet de créer un cycle simple).
5. G est acyclique et vérifie : $|V| \leq |E| + 1$.
6. pour tout couple de sommets (s, t), il existe un unique chemin simple allant de s à t.

Exercice 24 Écrire un algorithme de complexité optimale décidant si un graphe est un arbre.

Exercice 25 Sur un graphe de votre choix, éxécuter l'algorithme `ACM-Kruskal` en indiquant les états successifs des différentes variables.

Exercice 26 L'algorithme `ACM-Kruskal` requiert un type `partition`. Implémentez celui-ci puis calculez la complexité en temps de cet algorithme.

Exercice 27 Mêmes questions que précédemment relativement à l'algorithme `ACM-Prim`.

Exercice 28 On considère ici des graphes simples, non orientés à arêtes pondérées. Pour tout graphe G, toute arête e et tout réel $r > 0$, on note $G[e : r]$ le graphe obtenu à partir de G en ajoutant l'arête e si il n'existe déjà et en lui associant comme nouveau poids r. Le problème à considérer est le suivant :

`ACM-décalage`
`ENTRÉE : un graphe` G`, un ACM` D `de` G`, une arête` e `et un réel` $r > 0$ `;`
`SORTIE : un ACM` C `de` $G[e : r]$

Écrire un algorithme résolvant ce problème. Une solution évidente consiste à éxécuter n'importe quel algorithme solutionnant `ACM` sur l'entrée $G[r : e]$. Votre algorithme devra au contraire chercher à exploiter la donnée de D. Votre algorithme pourra distinguer deux cas selon que e appartienne à D ou non.

Chapitre 5

Plus court chemin

Exercice 29 Considérons l'algorithme `Bellman-Ford`.

1. Quelle est sa complexité en temps et en espace dans le pire et le meilleur des cas ?
2. Écrire une variante améliorant l'une de ces complexités. L'idée étant de poursuivre le calcul jusqu'à stabiliser `d`. Évaluer les complexités en temps dans le meilleur et le pire des cas.

Exercice 30 Dans l'algorithme de `Dijkstra`, implémenter l'instruction `extraire un sommet x de Y de valeur d minimale`. Évaluer la complexité en temps et en espace dans le pire et le meilleur des cas.

Exercice 31 Fournissez un graphe G et deux sommmets s et t tels qu'il existe un plus court chemin de s à t et pour lesquels `Dijkstra` retourne un résultat erroné.

Exercice 32 Nous nous intéressons ici au problème `PlusCourtCheminSourcesMultiples` ainsi défini :

`Input` G un graphe orienté sans circuit de longueur < 0.

`Output` la fonction distance $\delta_G(.,.)$.

Supposons que l'ensemble des sommets V de G forme un intervalle d'entiers de la forme $[1, n]$. Définissons pour tous entiers $i, j, k \in [1, n]$ la valeur $M^i[j, k]$ comme la longueur du plus court chemin allant de j à k et dont tous les sommets internes sont dans $[1, i]$.

1. Fournir une définition récursive de $M^i[j, k]$.
2. Déduire de la question précédente un algorithme résolvant le problème ci-dessus. Évaluer sa complexité.
3. Modifier l'algorithme afin de calculer les plus courts chemins.

Exercice 33 Nous nous intéressons ici au problème `PlusLongChemin` ainsi défini :

`Input` G un graphe acyclique, s et t deux sommets

`Output` w un chemin de s à t de plus grande longueur pondérée

1. Fournir une solution algorithmique qui utilise `Bellman-Ford`. Prouvez sa correction.
2. En vous inspirant de `Dijkstra`, fournir une autre solution de plus grande efficacité.

Exercice 34 Démontrer que le problème du calcul du temps minimal à l'éxécution d'un chantier par une entreprise présenté en introdution de cours se réduit au problème `PlusLongChemin`.

Chapitre 6

Flot maximal

Exercice 35 Soit le réseau $G := (V, E, 1, 6, c)$ où $V := [1, 6]$, où $E := V^2$ et où c associe aux arcs $(1, 2), (1, 4), (2, 3), (2, 4), (3, 4), (3, 6), (4, 5), (5, 6)$ respectivement les réels $8, 3, 7, 1, 2, 4, 5, 6$ et à tout autre arc la capacité 0.

- Calculer un flot maximal f de G en utilisant `FordFulkerson`.
- Trouver une coupe de G de capacité la valeur du flot f. Qu'en conclure ?

Exercice 36 Écrire un algorithme décidant si un flot d'un réseau est maximal dans ce réseau.

Exercice 37 Démontrer que tout réseau à capacités entières admet un flot maximal qui associe à tout arc une valeur entière.

Exercice 38 Soit le réseau $G := (V, E, 1, 4, c)$ où $V := [1, 4]$, où $E := V^2$ et où c associe aux arcs $(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)$ respectivement les réels $10^6, 10^6, 1, 10^6, 10^6$ et à tout autre arc la capacité 0. Fournir différentes éxécutions de `FordFulkerson` sur l'instance G.

Exercice 39 Démontrer que si les capacités du réseau a des valeurs entières, `FordFulkerson` termine et a une complexité en temps dans le pire des cas $\Theta(m \cdot v)$ où m désigne le nombre d'arcs de G de capacité non nulle et où v est la valeur d'un flot maximal.

Exercice 40 Chercher un exemple de réseau sur lequel `FordFulkerson` ne termine pas.

Exercice 41 L'association "Tout le monde à la plage" doit amener un maximum de citadins dans différentes villes de bord de mer et ce en utilisant des routes à capacité limitée. Réduire ce problème en un problème de flot maximal.

Exercice 42 L'entreprise "Électricité Sans Plutonium" souhaite approvisionner chacune des villes v d'une quantité $q(v)$ d'électricité. Le réseau électrique fort complexe dispose de lignes de capacités différentes. Réduire ce problème en un problème de flot maximal en distinguant selon que chaque centrale électrique a une capacité infinie ou non.

Exercice 43 (Couplage maximum) Une agence matrimoniale propose chaque mois à des hommes et femmes de se rencontrer. À la suite de ces rencontres, elle enregistre leur éventuelle affinité mutuelle et essaie de former ainsi un nombre maximum de couples.

1. Formaliser ce problème en un problème de graphes.

   On suppose maintenant que les rencontres sont uniquement hétérosexuelles.

2. Formaliser ce problème en un problème de graphes.
3. Réduire ce problème en un problème de flot maximal.
4. Résoudre le porblème initial.

Chapitre 7

Théorèmes de Menger

Rappel

Un ensemble S de sommets (resp. d'arcs, d'arêtes) *sépare* un sommet s d'un sommet t dans un graphe G si il ne contient ni s ni t et si tout chemin de G allant de s à t contient nécessairement un des éléments de S.

Exercice 44 L'objet de cet exercice est de démontrer le Théorème de Menger relatif à la k-arc-connexité dans le cas de graphes simples. Pour chaque graphe simple et orienté $G := (V, E)$ et chaque couple de sommets (s, t), nous notons $r(G, s, t)$ le réseau (V, V^2, c, s, t) où c associe la capacité 1 à tout arc de E et 0 sinon. Établir pour tout couple de sommets distincts (s, t) l'équivalence des 4 assertions suivantes :

1. dans G, tout ensemble d'arcs séparant s de t est cardinalité $\geq k$.
2. dans $r(G, s, t)$, toute coupe est de capacité $\geq k$.
3. dans $r(G, s, t)$, il existe un flot maximal de valeur $\geq k$.
4. dans G, il existe au moins k chemins de s à t deux à deux arcs-disjoints.

Exercice 45 Étendre le résultat précédent aux graphes à arcs mulitples.

Exercice 46 L'objet de cet exercice est de démontrer le Théorème de Menger relatif à la k-arête-connexité. Pour chaque graphe non orienté $G := (V, E, f)$, nous notons $o(G)$ un graphe orienté $(V, E \times \{0, 1\}, g)$ où chaque arête de G est transformée en deux arcs opposés. Définir précisément l'application $f(G)$, puis établir pour tout couple de sommets distincts (s, t) l'équivalence des 4 assertions suivantes :

1. dans G, tout ensemble d'arêtes séparant s de t est cardinalité $\geq k$.
2. dans $o(G)$, tout ensemble d'arcs séparant s de t est cardinalité $\geq k$.
3. dans $o(G)$, il existe au moins k chemins de s à t deux à deux arcs-disjoints.
4. dans G, il existe au moins k chemins de s à t deux à deux arêtes-disjoints.

Exercice 47 L'objet, ici, est de démontrer le Théorème de Menger relatif à la k-connexité. Pour tout graphe simple orienté $G := (V, E)$, nous notons $f(G)$ le graphe orienté $(V \times \{0, 1\}, F)$ où tout sommet s de G est remplacé par deux nouveaux sommets $(s, 0)$ et $(s, 1)$, le premier extrémité terminale des arcs entrants de s, le second extrémité initiale des arcs sortant de s. Définir précisément l'application $f(G)$, puis établir pour tout couple de sommets s et t non adjacents l'équivalence des 4 assertions suivantes :

1. dans G, tout ensemble de sommets séparant s de t est cardinalité $\geq k$.
2. dans $f(G)$, tout ensemble d'arcs séparant $(s, 1)$ de $(t, 0)$ est cardinalité $\geq k$.
3. dans $f(G)$, il existe au moins k chemins de $(s, 1)$ à $(t, 0)$ deux à deux arcs-disjoints.
4. dans G, il existe au moins k chemins de s à t deux à deux sommets-disjoints.

Chapitre 8

Couplages, Recouvrements, cliques

Exercice 48 Les graphes considérés ici sont non orientés et sans sommets isolés. Un *recouvrement* d'un graphe G est un ensemble d'arêtes $D \subseteq E(G)$ tel que tout sommet de $V(G)$ est incident à au moins une arête de D. Ce recouvrement est *minimum* si il est de cardinalité minimale parmi tous les recouvrements. Le problème à résoudre est le suivant :

```
Recouvrement minimum
  ENTRÉE : un graphe G sans sommets isolés.
  SORTIE : un recouvrement minimum de G
```

1. Trouver un recouvrement du graphe dessiné sur la Figure 8.1 de cardinalité ≤ 9.

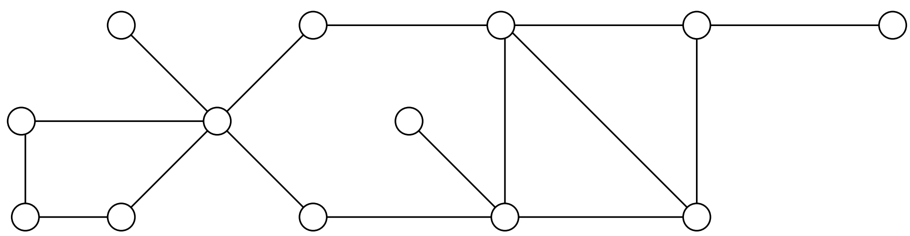

Fig. 8.1 – Un graphe

2. Trouver un couplage maximum du graphe dessiné sur la Figure 8.1. Prouver qu'il est maximum.
3. Trouver quelques propriétés significatives du graphe partiel engendré $G|D$ par un graphe G et un de ses recouvrements minimums D. Exprimer notamment le nombre de composantes connexes de $G|D$ en fonction de $|V(G)|$ et de $|D|$.

4. Indiquer comment construire un couplage à partir d'un recouvrement minimum. Prouver que si C est un couplage maximum de G et D est un recouvrement minimum de G, alors $|C| + |D| \leq |V(G)|$.
5. Indiquer comment construire un recouvrement à partir d'un couplage maximum. Prouver que si C est un couplage maximum de G et D est un recouvrement minimum de G, alors $|C| + |D| \geq |V(G)|$.
6. En utilisant un algorithme vu en cours, résoudre (en quelques lignes) le problème `Recouvrement minimum`.
7. Prouver sa correction.
8. Évaluer sa complexité en temps dans le pire des cas.
9. Trouver un recouvrement mimimum du graphe dessiné sur la Figure 8.1. Prouver qu'il est minimum.

Exercice 49 Nous considérons ici des graphes simples sans boucles non orientés. Une *clique* d'un graphe G est un ensemble de sommets deux à deux adjacents (toute paire de sommets distincts sont extrémités d'une arête) ; elle est *maximum* si elle est de cardinalité maximale parmi toutes les cliques. Un *recouvrement-sommets* d'un graphe H est un ensemble de sommets telle que toute arête de H est incidente à au moins l'un des sommets de cet ensemble. Il est *minimum* si il est de cardinalité minimale pami tous les recouvrement-sommets de H. Le graphe complémentaire de G est le graphe noté $\mathbf{co}(G)$ contenant les mêmes sommets que G et contenant comme arête toute paire de sommets distincts qui ne soit pas une arête de G. Soit I le graphe dessiné sur la Figure 8.2.

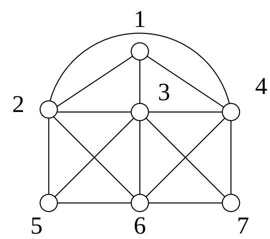

Fig. 8.2 – Un graphe I

1. Enumérer l'ensemble des cliques de I puis l'ensemble de ses cliques maximums.
2. Dessiner $\mathbf{co}(I)$ le graphe complémentaire de I.
3. Enumérer l'ensemble des recouvrement-sommets minimums de $\mathbf{co}(I)$.
4. Indiquer comment à partir d'une clique d'un graphe G obtenir un recouvrement-sommet de $\mathbf{co}(G)$. Prouver le.

5. Indiquer comment à partir d'un recouvrement-sommet d'un graphe H obtenir une clique de $\mathbf{co}(H)$. Prouver le.
6. Mêmes questions que précédemment en considérant des cliques maximums et des recouvrement-sommets minimums.
7. En supposant qu'il n'existe aucun algorithme de complexité en temps polynomiale pour calculer une clique maximum d'un graphe, que peut-on conclure et pourquoi ?
8. Connaissez-vous un problème proche (syntaxiquement) du recouvrement-sommet minimum qui admette une solution de complexité en temps polynomiale ?