Couverture minimale d'un graphe

Le sujet comporte 12 pages, numérotées de 1 à 12.

Début de l'épreuve.

Ce sujet est consacré au calcul d'un paramètre de graphe appelé ***couverture minimale***, défini comme étant le plus petit nombre de sommets d'un ***ensemble couvrant***. On étudiera plusieurs façon de le calculer, ce qui nous permettra de faire un tour d'horizon de quelques techniques algorithmiques modernes.

Ce sujet est constitué de 6 parties qui peuvent être traitées relativement indépendamment ; cependant, les concepts introduits dans une partie peuvent être réutilisés dans des parties ultérieures.

- Dans la première partie (page 3) on découvre progressivement certaines propriétés des couvertures minimales, leurs valeurs sur quelques familles de graphes simples et leurs liens avec d'autres paramètres classiques de la théorie des graphes.
- Dans la deuxième partie (page 4) on développe deux algorithmes efficaces qui calculent un ensemble couvrant minimum d'un arbre.
- Dans la troisième partie (page 5) on étudie deux heuristiques naturelles calculant un ensemble couvrant s'avérant loin d'être minimum, pour finalement découvrir un algorithme simple calculant une approximation de la couverture minimale.
- Dans la quatrième partie (page 7) on découvre la complexité paramétrée.
- Dans la cinquième partie (page 9) on continue notre exploration de la complexité paramétrée en introduisant la notion de noyau.
- Dans la sixième partie (page 10) on découvre la puissance de l'optimisation linéaire, permettant une autre approximation de la couverture minimale ainsi qu'un noyau plus petit que celui de la partie V.

Notations et définitions

— Un ***graphe*** (non-orienté) G est la donnée d'un ensemble fini $\mathrm{V}(G)$ de ***sommets*** et d'un ensemble $\mathrm{E}(G)$ dont les éléments, appelés ***arêtes***, sont des paires de sommets de V. On notera le fait que deux sommets u et v sont reliés par une arête par $\{u, v\} \in \mathrm{E}(G)$. On dit que u et v sont les ***extrémités*** de l'arête $\{u, v\}$ et que u et v sont ***adjacents*** dans G.

— Une conséquence de cette définition est que tous les graphes considérés sont ***simples***, c'est à dire sans boucle ($\{u, u\} = \{u\}$ n'est pas une arête), et avec au plus une arête entre deux sommets.

— Soit v un sommet d'un graphe G. Le ***voisinage*** $\mathrm{N}(v)$ de v est l'ensemble des sommets adjacents à v, c'est-à-dire $\{\, u \in \mathrm{V}(G) \mid \{u, v\} \in \mathrm{E}(G) \,\}$. Le ***degré*** $\mathrm{d}(v)$ est le cardinal de $\mathrm{N}(v)$.

— Un ***chemin*** $P = (v_1, \ldots, v_n)$ dans un graphe G est une liste de sommets deux à deux distincts tels que pour tout $1 \leqslant i < n$, v_i et v_{i+1} sont adjacents dans G.

— Un ***ensemble couvrant*** d'un graphe G est un ensemble de sommets X tel que toute arête de G a au moins une extrémité dans X. Le nombre de sommets d'un ***ensemble couvrant minimum*** (c'est-à-dire, un ensemble couvrant minimisant le nombre de sommets) de G est appelé ***couverture minimale*** de G et noté $\tau(G)$.

— Soit G un graphe et X un ensemble de sommets de G. On dénote par $G - X$ le graphe sur les sommets $\mathrm{V}(G) \setminus X$ avec comme arêtes $\{\, \{u, v\} \in \mathrm{E}(G) \mid u \notin X \text{ et } v \notin X \,\}$.

— Par ***complexité*** (en temps) d'un algorithme ALG, on entend le nombre d'opérations élémentaires nécessaires à l'exécution de ALG dans le pire des cas.

— Lorsque la complexité en temps dépend d'un ou plusieurs paramètres $x_1, \ldots, x_p$, on dit qu'elle est en $O(f(x_1, \ldots, x_p))$ s'il existe une constante $C > 0$ telle que, pour toutes les valeurs de $x_1, \ldots, x_p$, suffisamment grandes (c'est-à-dire, plus grandes qu'un certain seuil), la complexité est au plus $C \times f(x_1, \ldots, x_p)$. La complexité est dite ***linéaire*** si $f(x_1, \ldots, x_p) = x_1 + \cdots + x_p$.

— Dans tout l'énoncé (sauf dans la partie II où on travaille sur des arbres et dans la partie III où on ne s'intéresse pas à la complexité précise des algorithmes), on fera l'hypothèse que les graphes sont encodés par leur matrice d'adjacence.

Partie I

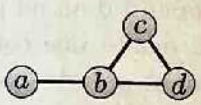
Un graphe est ***complet*** si tous les sommets sont deux à deux adjacents. On dénote par K_n le graphe complet sur n sommets $\{1, \ldots, n\}$.

Un ***ensemble indépendant*** d'un graphe est un ensemble de sommets du graphe dont les sommets sont deux à deux non-adjacents. On note $\alpha(G)$ le nombre maximum de sommets d'un ensemble indépendant de G. Un ***couplage*** est un ensemble d'arêtes deux à deux disjointes, c'est à dire ne partageant aucun sommet. Le nombre maximum d'arêtes dans un couplage de G est noté $\mu(G)$.

Un graphe est ***biparti*** si ses sommets peuvent être partitionnés en deux ensembles indépendants. Il est ***biparti complet*** s'il admet une partition en deux ensembles indépendants telle que chaque sommet d'une partie est adjacent à chaque sommet de l'autre partie. Étant donné deux entiers naturels a et b, on dénote par $K_{a,b}$ le graphe biparti complet dont les parties de la partition en deux ensembles indépendants ont respectivement a et b sommets (disons, les sommets étant numérotés de 1 à a dans la première partie et de $a+1$ à $a+b$ dans la deuxième).

On dénote par P_n le ***graphe chemin*** sur $n \geqslant 1$ sommets, c'est-à-dire le graphe dont les sommets sont $\{1, \ldots, n\}$ et les arêtes $\{\{1,2\}, \{2,3\}, \ldots, \{n-1, n\}\}$. Un ***graphe cycle*** est le graphe obtenu en ajoutant une arête entre les deux extrémités d'un graphe chemin. On dénote par C_n le graphe cycle sur $n \geqslant 2$ sommets.

Question I.1. Donner les valeurs de $\tau(G)$, $\alpha(G)$ et $\mu(G)$ pour le graphe G représenté ci-dessous. Justifier.



Question I.2. Donner la valeur de $\tau(K_n)$, $\tau(P_n)$, $\tau(C_n)$ et $\tau(K_{a,b})$ pour tous entiers naturels non nuls a, b, n. Justifier soigneusement les réponses.

Question I.3. Montrer que pour tout graphe G, $\tau(G) = |\mathrm{V}(G)| - \alpha(G)$.

Question I.4. Montrer que pour tout graphe G, $\tau(G) \geqslant \mu(G)$.

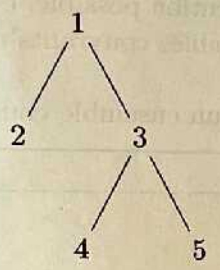
Aucun algorithme polynomial n'est connu pour calculer un ensemble couvrant minimum d'un graphe. Dans la question suivante, on demande un algorithme exponentiel.

Question I.5. Donner une description à haut niveau (sans détailler les lignes de pseudo-code) d'un algorithme calculant un ensemble couvrant minimum d'un graphe G en temps $O\left(2^{|\mathrm{V}(G)|} \cdot |\mathrm{V}(G)|^2\right)$.

Partie II

Un ***arbre*** est un graphe connexe sans cycle. Un sommet de degré $\leqslant 1$ d'un arbre est appelé une ***feuille***. Dans cette partie, on va développer deux algorithmes efficaces pour calculer un ensemble couvrant minimum d'un arbre.

Dans cette partie, contrairement au reste du sujet, les arbres ne sont pas représentés par leur matrice d'adjacence mais par un tableau T dont les éléments sont des ensembles de sommets : pour chaque sommet $1 \leqslant i \leqslant n$ (où n est le nombre de sommets de l'arbre), $T[i]$ est une représentation de l'ensemble des sommets adjacents à i. On supposera que la structure de données utilisée pour représenter l'ensemble $T[i]$ permet en temps $O(1)$ les opérations suivantes : renvoyer le nombre d'éléments de l'ensemble ; rechercher si un élément donné appartient à l'ensemble ; ajouter un nouvel élément à l'ensemble ; retirer un élément existant de l'ensemble. Par exemple, voici un arbre sur les sommets $\{1, \ldots, 5\}$ et sa représentation sous forme de tableau dont les éléments sont les ensembles de sommets adjacents :



1	{2, 3}
2	{1}
3	{1, 4, 5}
4	{3}
5	{3}

Question II.1. Soient T un arbre et $\{u, v\}$ une arête de T telle que v est une feuille de T. Montrer qu'il existe un ensemble couvrant minimum de T qui contient le sommet u, mais pas v.

Question II.2. En utilisant la question précédente, proposer un algorithme (en pseudo-code) qui reçoit un arbre T en entrée, et renvoie un ensemble couvrant minimum de T, en enlevant progressivement des arêtes à l'arbre. La complexité de cet algorithme doit être linéaire en le nombre n de sommets de l'arbre. Justifier de la complexité de l'algorithme.

Question II.3. Proposer un autre algorithme (en pseudo-code) qui renvoie le nombre de sommets d'un ensemble couvrant minimum d'un arbre, cette fois basé sur la programmation dynamique. On pourra commencer par ***enraciner*** l'arbre, c'est-à-dire choisir un sommet arbitraire qui jouera le rôle de racine de l'arbre. Pour chaque sommet $v \in \mathrm{V}(G)$, on note T_v le sous-arbre de T enraciné en v ; pour calculer le résultat final, l'algorithme devra d'abord calculer en programmation dynamique pour chaque $v \in \mathrm{V}(G)$ les trois valeurs suivantes :

- $D_{\in}(v)$ = nombre de sommets d'un ensemble couvrant minimum de T_v contenant v ;
- $D_{\notin}(v)$ = nombre de sommets d'un ensemble couvrant minimum de T_v ne contenant pas v ;
- $D(v)$ = nombre de sommets d'un ensemble couvrant minimum de T_v.

On ne demande pas la preuve de la correction, mais il faut donner et prouver sa complexité en fonction du nombre n de sommets de l'arbre.

Partie III

Dans cette partie, plutôt que d'essayer de calculer un ensemble couvrant minimum, on va chercher à en calculer un dont le cardinal est proche du minimum.

Formalisons cette idée. Soit ALG un algorithme prenant un graphe en entrée et renvoyant un ensemble couvrant (pas forcément minimum). On note $ALG(G)$ le nombre de sommets de l'ensemble couvrant renvoyé par ALG appliqué à G. On note $\mathrm{OPT}(G)$ le nombre de sommets d'un ensemble couvrant minimum de G. On dit que ALG est une ***c-approximation*** (pour un nombre réel $c \geqslant 1$) si pour tout graphe G, on a :

$$\frac{ALG(G)}{\mathrm{OPT}(G)} \leqslant c.$$

Noter que les algorithmes que nous allons considérer n'ont pas une façon unique de s'exécuter sur un graphe donné. Typiquement, à la ligne 3 de l'algorithme HEURISTIQUE ci-dessous, il peut y avoir beaucoup de façon différentes de choisir le sommet v. Dans ce cas, on considère que HEURSITIQUE(G) est le résultat de la pire exécution possible, c'est à dire celle renvoyant le plus grand ensemble couvrant parmi tous les ensembles couvrants qu'elle peut potentiellement renvoyer.

On considère un premier algorithme calculant un ensemble couvrant :

Algorithme 1 HEURISTIQUE(G)

1: $S \leftarrow \varnothing$
2: **while** G a des arêtes **do**
3: Choisir un sommet v de G tel que $\mathrm{d}(v) \geqslant 1$
4: $S \leftarrow S \cup \{v\}$
5: $G \leftarrow G - \{v\}$
6: **return** S

Question III.1. Montrer que l'ensemble S calculé par HEURISTIQUE est bien un ensemble couvrant.

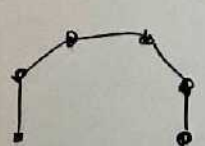
Question III.2. Montrer que, pour tout entier $n \geqslant 2$, il existe un graphe G à n sommets tel que

$$\frac{\text{HEURISTIQUE}(G)}{\mathrm{OPT}(G)} \geqslant n - 1.$$

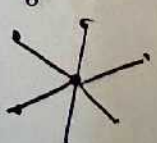
L'heuristique ci-dessus pouvant donner des résultats catastrophiques, on propose de l'améliorer en choisissant le sommet de degré maximum, donc celui couvrant le plus d'arêtes.

Algorithme 2 GLOUTON(G)

1: $S \leftarrow \varnothing$
2: **while** G a des arêtes **do**
3: Choisir un sommet v de degré maximum dans G
4: $S \leftarrow S \cup \{v\}$
5: $G \leftarrow G - \{v\}$
6: **return** S



Heuristique $(P_n) = n - 1$



GLOUTON renvoie un ensemble couvrant pour les mêmes raisons que HEURISTIQUE.

On va définir une série de graphes bipartis G_n avec bipartition A_n et B_n comme suit. A_n est un ensemble de n sommets. B_n est partitionné en n sous-ensembles $B_n^1, \ldots, B_n^n$ ayant les propriétés suivantes pour $k = 1, \ldots, n$:

(i) $|B_n^k| = \lfloor \frac{n}{k} \rfloor$,

(ii) chaque sommet de B_n^k a exactement k voisins dans A_n,

(iii) deux sommets de B_n^k n'ont pas de voisins communs dans A_n.

Question III.3. Dessiner (une réalisation possible de) G_4 et donner les valeurs de $\text{OPT}(G_4)$ et de $\text{GLOUTON}(G_4)$. On ne demande pas de justifier.

Question III.4. En utilisant la construction ci-dessus, montrer que pour toute constante c, GLOUTON n'est pas une c-approximation.

Ainsi, GLOUTON ne donne pas non plus une bonne approximation. On va maintenant considérer l'algorithme suivant.

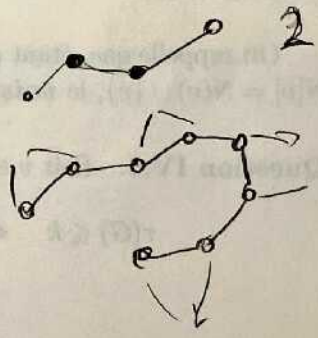
Algorithme 3 APPROX(G)

```
S ← ∅
while G a des arêtes do
    Choisir une arête {u, v} de G
    S ← (S ∪ {u}) ∪ {v}
    G ← (G − {u}) − {v}
return S
```

Question III.5. Montrer que APPROX renvoie un ensemble couvrant, en s'appuyant sur la question I.3.

Question III.6. Montrer que APPROX est une 2-approximation.

Question III.7. Donner une suite infinie de graphes justifiant que, pour tout $c < 2$, APPROX n'est pas une c-approximation.



Partie IV

Dans cette partie, on cherche à répondre à la question suivante : *Étant donné un graphe G et un entier k, est-ce que $\tau(G) \leqslant k$?* On rappelle que G est représenté par sa matrice d'adjacence.

Let but est de trouver un algorithme qui, étant donné un graphe G et un entier k, décide si $\tau(G) \leqslant k$ en temps $O(f(k) \times |\mathrm{V}(G)|^c)$ où f est une fonction (arbitraire, potentiellement exponentielle en son argument) et $c \geqslant 0$ une constante. Un algorithme avec une telle complexité est dit ***FPT*** pour ***fixed-parameter tractable*** ou ***tractable à paramètre fixé***.

On va commencer modestement. Noter que l'algorithme demandé à la question ci-dessous n'est pas FPT.

Question IV.1. Donner une description à haut niveau (sans détailler les lignes de pseudo-code) d'un algorithme qui, étant donné un graphe G et un entier k, décide si $\tau(G) \leqslant k$ en temps $O(|\mathrm{V}(G)|^{k+2})$.

Le reste de cette partie est dédié à la conception de notre premier algorithme FPT.

Question IV.2. Soit G un graphe et $\{u, v\} \in \mathrm{E}(G)$. Montrer l'équivalence suivante :

$$\tau(G) \leqslant k \iff \tau(G - \{u\}) \leqslant k - 1 \text{ ou } \tau(G - \{v\}) \leqslant k - 1.$$

Question IV.3. En s'aidant de la question précédente, donner un algorithme récursif (en pseudo-code) qui, étant donné un graphe G et un entier k, décide si $\tau(G) \leqslant k$ en temps $O(2^k \times |\mathrm{V}(G)|^2)$. Justifier la correction et la complexité de l'algorithme. On rappelle qu'avec la représentation en matrice d'adjacence, il est possible de supprimer l'ensemble des arêtes dont un sommet u est une extrémité en $O(|V(G)|)$.

On va maintenant légèrement modifier l'algorithme de la question précédente pour améliorer sa complexité.

Question IV.4. Donner une description précise des graphes de degré maximum 2. En déduire une description à haut niveau (sans détailler les lignes de pseudo-code) d'un algorithme en $O(|\mathrm{V}(G)|)$ pour décider si un graphe de degré maximum 2 a un ensemble couvrant avec au plus k sommets.

On rappelle que, étant donné un sommet v, $\mathrm{N}(v)$ est l'ensemble des voisins de v et on définit $\mathrm{N}[v] = \mathrm{N}(v) \cup \{v\}$, le ***voisinage fermé*** de v.

Question IV.5. Soit $v \in V$. Montrer que si $\mathrm{d}(v) \geqslant 3$, alors

$$\tau(G) \leqslant k \iff \tau(G - \{v\}) \leqslant k - 1 \text{ ou } \tau(G - \mathrm{N}[v]) \leqslant k - \mathrm{d}(v).$$

Question IV.6. En s'aidant de la question précédente, donner un algorithme récursif (en pseudo-code) qui décide si $\tau(G) \leqslant k$ en temps $O(1.4656^k \times |\mathrm{V}(G)|^2)$. Justifier la correction et la complexité de l'algorithme. On utilisera le fait que la fonction $f(k)$ définie récursivement comme suit :

$$f(i) = \begin{cases} f(i-1) + f(i-3) & \text{si } i \geqslant 3 \\ 1 & \text{sinon.} \end{cases}$$

satisfait $f(k) \leqslant 1.4656^k$.

Partie V

Comme dans la partie précédente, on cherche à répondre à la question suivante : *Étant donné un graphe G et un entier k, est-ce que $\tau(G) \leqslant k$?*

On va s'intéresser à la notion de noyau. On dit que le problème de l'ensemble couvrant a un ***noyau de taille*** $f(k)$ (pour une certaine fonction croissante f) s'il existe un algorithme qui étant donné une instance (G, k) renvoie en temps polynomial (en $|\mathrm{V}(G)|$) une instance (G', k') tel que :

(i) $\tau(G) \leqslant k$ si et seulement si $\tau(G') \leqslant k'$,
(ii) si $\tau(G) \leqslant k$, alors $|\mathrm{V}(G')| \leqslant f(k)$ et
(iii) $k' \leqslant k$.

Question V.1. Montrer que si le problème de l'ensemble couvrant a un noyau de taille $f(k)$, alors il existe un polynôme P tel qu'on peut décider si $\tau(G) \leqslant k$ en temps $O\left(P(|\mathrm{V}(G)|) + f(k)^{k+2}\right)$.

Le reste de la partie est dédié à la conception d'un algorithme montrant que le problème de l'ensemble couvrant a un noyau de taille $k^2 + k$.

Question V.2. Montrer que si un sommet v de G a degré 0, alors $\tau(G) \leqslant k$ si et seulement si $\tau(G - \{v\}) \leqslant k$.

Question V.3. Montrer que si un sommet v de G a degré au moins $k + 1$, alors $\tau(G) \leqslant k$ si et seulement si $\tau(G - \{v\}) \leqslant k - 1$.

On considère maintenant l'algorithme qui, étant donné (G, k), applique les deux règles suivantes (dans n'importe quel ordre) tant que c'est possible. On note (G', k') le couple obtenu.

Règle 1 : si un sommet v a degré 0, supprimer v.
Règle 2 : si un sommet v a degré au moins $k + 1$, supprimer v, et diminuer k de 1.

On remarque à l'aide des deux questions précédentes que $\tau(G) \leqslant k$ si et seulement si $\tau(G') \leqslant k'$.

Question V.4. Donner la complexité de cet algorithme en fonction de $|\mathrm{V}(G)|$ (on rappelle que G est représenté par sa matrice d'adjacence).

Question V.5. Montrer que pour tout sommet v de G', on a dans G' : $1 \leqslant \mathrm{d}(v) \leqslant k'$. En déduire que si $\tau(G') \leqslant k'$, alors $\mathrm{V}(G') \leqslant k' + k'^2$ et $\mathrm{E}(G') \leqslant k'^2$.

Question V.6. Conclure que le problème de l'ensemble couvrant a un noyau de taille $k^2 + k$.

Partie VI

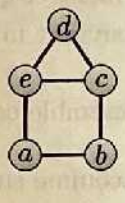
Soit $G = (V, E)$ un graphe avec $V = \{v_1, \ldots, v_n\}$. Un ensemble de sommets S peut être représenté par un vecteur $x = (x_{v_1}, x_{v_2}, \ldots, x_{v_n}) \in \{0,1\}^n$ tel que, pour $i = 1, \ldots, n$, $x_{v_i} = 1$ si $v_i \in S$ et $x_{v_i} = 0$ si $v_i \notin S$.

On introduit une variable x_v pour chaque sommet de G. Observer que les ensembles couvrant de G correspondent précisément aux solutions du système d'équations suivant :

$$\begin{cases} x_u + x_v \geqslant 1 \text{ pour tout } \{u, v\} \in \mathrm{E}(G); \\ x_v \in \{0,1\} \text{ pour tout } v \in \mathrm{V}(G). \end{cases} \tag{1}$$

Ainsi, une solution minimisant $\sum_{v \in \mathrm{V}(G)} x_v$ correspond à un ensemble couvrant minimum. Pour un graphe G donné, on nomme OLNEEC(G) (pour ***optimisation linéaire en nombre entier pour ensemble couvrant***) le système d'équations (1) associé à G.

Question VI.1. Écrire le système d'équations OLNEEC(G) pour le graphe ci-dessous.



En relaxant la contrainte $x_v \in \{0,1\}$ par $0 \leqslant x_v \leqslant 1$, on obtient le système d'équations suivant nommé OLEC(G) (pour ***optimisation linéaire pour ensemble couvrant***) :

$$\begin{cases} x_u + x_v \geqslant 1 \text{ pour tout } \{u, v\} \in \mathrm{E}(G); \\ 0 \leqslant x_v \leqslant 1 \text{ pour tout } v \in \mathrm{V}(G). \end{cases} \tag{2}$$

On peut considérer que $x_u = \frac{1}{3}$ (par exemple) correspond à mettre $\frac{1}{3}$ du sommet u dans la solution. Une solution à ce système d'équations s'appelle un ***ensemble couvrant fractionnaire*** de G et sa ***taille*** vaut $\sum_{v \in \mathrm{V}(G)} x_v$. Une solution minimisant $\sum_{v \in \mathrm{V}(G)} x_v$ est un ***ensemble couvrant fractionnaire minimum*** de G, et sa taille est noté $\tau_{\mathrm{f}}(G)$. Il est clair que pour tout graphe G,

$$\tau_{\mathrm{f}}(G) \leqslant \tau(G)$$

Question VI.2. Parmi les vecteurs $(x_a, x_b, x_c, x_d, x_e)$ suivants, dites lesquels sont des ensembles couvrants fractionnaires du graphe de la question VI.1 et lesquels ne le sont pas. Donner leur taille quand ils le sont, et expliquer précisément la raison quand ils ne le sont pas.

$$\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right), \quad \left(\frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{3}\right), \quad \left(1, 1, 0, 1, 0\right), \quad \left(1, 0, 1, 1, 0\right).$$

Question VI.3. Montrer que pour tout graphe G, on a $\tau_f(G) \leqslant \frac{|V(G)|}{2}$. En déduire qu'il n'existe pas de constante A telle que $\tau(G) - \tau_f(G) < A$ pour tout graphe G.

Soit G un graphe. Soit $(x_v)_{v \in V(G)} \in [0,1]^n$ un ensemble couvrant fractionnaire minimum de G. On partitionne les sommets de G de la façon suivante :

$$V_0 = \left\{v : x_v < \frac{1}{2}\right\} \qquad V_{\frac{1}{2}} = \left\{v : x_v = \frac{1}{2}\right\} \qquad V_1 = \left\{v : x_v > \frac{1}{2}\right\}$$

Question VI.4. Montrer que V_0 est un ensemble indépendant (c'est à dire qu'aucune arête n'a ses deux extrémités dans V_0), et qu'il n'y a pas d'arête ayant une extrémité dans V_0 et l'autre dans $V_{\frac{1}{2}}$.

Question VI.5. Montrer que $V_{\frac{1}{2}} \cup V_1$ est un ensemble couvrant de G et que $|V_{\frac{1}{2}} \cup V_1| \leqslant 2\tau(G)$. On admettra qu'il existe un algorithme en temps polynomial permettant de trouver un ensemble couvrant fractionnaire minimum. En déduire un algorithme polynomial qui donne une 2-approximation pour le problème de l'ensemble couvrant minimum.

Le but des 3 prochaines questions est de montrer qu'il existe un ensemble couvrant minimum inclus dans $V_{\frac{1}{2}} \cup V_1$. Soit S^* un ensemble couvrant minimum de G, et soit $S = \left(V_{\frac{1}{2}} \cap S^*\right) \cup V_1$.

Question VI.6. Montrer que S est un ensemble couvrant de G.

On va perturber la solution $(x_v)_{v \in V(G)}$ comme suit. On pose

$$\varepsilon = 0 \text{ si } V_0 \cup V_1 = \varnothing \qquad \text{et} \qquad \varepsilon = \min\left\{\left|x_v - \frac{1}{2}\right| : v \in V_0 \cup V_1\right\} \text{ sinon}$$

et pour tout $v \in V(G)$, on définit :

$$y_v = \begin{cases} x_v - \varepsilon & \text{si } v \in V_1 \setminus S^*; \\ x_v + \varepsilon & \text{si } v \in V_0 \cap S^*; \\ x_v & \text{dans tous les autres cas.} \end{cases}$$

Question VI.7. Montrer que $(y_v)_{v \in V(G)}$ est un ensemble couvrant fractionnaire de G.

Question VI.8. Montrer que S est un ensemble couvrant minimum de G.

On va maintenant étudier l'algorithme suivant pour déterminer, étant donné un graphe G et un entier k, si $\tau(G) \leqslant k$.

Algorithme 4 OLNoyau(G, k)

1: Calculer un ensemble couvrant fractionnaire minimum $(x_v)_{v \in V(G)}$ à l'aide d'un algorithme en temps polynomial.
2: Définir les ensembles V_0, $V_{\frac{1}{2}}$ et V_1 comme précédemment.
3: **if** $\sum_{v \in V(G)} x_v > k$ **then return** NON
4: **if** $\left|V_{\frac{1}{2}}\right| > 2k$ **then return** NON
5: Résoudre l'instance $(G - V_0 - V_1, k - |V_1|)$ en utilisant un algorithme par force brute.

Question VI.9. Montrer que si l'algorithme renvoie NON aux lignes 3 ou 4, alors $\tau(G) > k$.

Question VI.10. Montrer que $\tau(G) \leqslant k$ si et seulement si $\tau(G - V_0 - V_1) \leqslant k - |V_1|$. Conclure que le problème de l'ensemble couvrant a un noyau de taille $2k$.

Fin du sujet.