Optimisation de réseau

Informatique

La partie I de ce problème étudie des généralités sur les arbres couvrants et une fonction de tri. La partie II présente l'algorithme de Kruskal inversé pour trouver un arbre couvrant de poids minimal dans un graphe connexe (Les questions de programmation seront en Ocaml). La partie III montre la complexité du problème d'optimisation de réseau en se ramenant au problème de satisfaisabilité. La partie IV traite le problème du goulot maximal dans un graphe (Les questions de programmation seront en C). Enfin, la partie V présente comment approcher une solution au problème d'optimisation en utilisant des arbres couvrants.

Définitions et notations

Pour un ensemble fini E, on note $|E|$ le cardinal de E. On note $\mathscr{P}_2(E)$ l'ensemble des paires d'éléments de E, c'est-à-dire les sous-ensembles de E de cardinal 2.

On définit **un graphe non orienté** comme un couple $G = (S,A)$ tel que S est un ensemble fini d'éléments appelés sommets et A un ensemble de paires d'éléments distincts de S appelées arêtes, c'est-à-dire $A \subset \mathscr{P}_2(S)$. S'il n'y a pas d'ambiguïté, une paire $\{s,t\}$ sera notée st. Si st est une arête du graphe, on dit que s et t sont **adjacents**.
Le graphe $G_1 = (\{s_0,s_1,s_2,s_3,s_4,s_5\}, \{s_0s_2,s_0s_3,s_0s_4,s_1s_2,s_1s_3,s_2s_3,s_2s_5,s_3s_4\})$ est représenté sur la figure 1.

On appelle **sous-graphe** de $G = (S,A)$ un graphe $H = (X,B)$ tel que $X \subset S$ et $B \subset A \cap \mathcal{P}_2(X)$. Si $X \subset S$, on appelle **sous-graphe induit par X** le graphe $G[X] = (X, A \cap \mathcal{P}_2(X))$, c'est-à-dire le graphe ayant X comme ensemble de sommets et contenant toutes les arêtes de G dont les deux extrémités sont dans X.

Pour $(s,t) \in V^2$, on appelle **chaîne** de s à t une suite de sommets distincts $(s_0,s_1,\dots,s_k)$ telle que $s_0 = s$, $s_k = t$ et pour $i \in [\![1,k]\!]$, $s_{i-1}s_i \in A$. On appelle **cycle** de G une suite de sommets $(s_0,s_1,\dots,s_k)$ telle que $k \geqslant 3$, $(s_0,\dots,s_{k-1})$ est une chaîne, $s_0 = s_k$ et $s_0s_{k-1} \in A$.

Un graphe est dit **connexe** s'il existe toujours une chaîne entre deux sommets distincts. Si $G = (S,A)$, on appelle **composante connexe** de G un sous-ensemble X de S tel que $G[X]$ est connexe et pour tout $s \in S \backslash X$, $G[X \cup \{s\}]$ n'est pas connexe.

Un graphe est dit un **arbre** (à différencier des arbres enracinés) s'il est connexe et ne contient pas de cycle. On dit qu'un sous-graphe $T = (X,B)$ de $G = (S,A)$ qu'il est un **arbre couvrant** si $S = X$ et T est un arbre.

On appelle **graphe pondéré** un triplet (S,A,f) tel que (S,A) est un graphe et $f : A \to \mathbb{R}_+$est une fonction qui à chaque arête associe un **poids** qui est un réel positif. La figure 3 contient une représentation graphique d'un graphe pondéré G_2.

Dans un graphe pondéré $G = (S,A,f)$, le **poids** d'un sous-graphe $H = (X,B,f)$ de G est la somme des poids des arêtes qui le composent, c'est-à-dire $f(H) = \sum_{a \in B} f(a)$.

I Généralités

Q. 1 Dessiner sans justifier un arbre couvrant de G_2 de poids minimal.

Q. 2 Soit $G = (S,A)$ un graphe non orienté. Montrer les deux propriétés suivantes :

- Si G est connexe, alors $|A| \geqslant |S| - 1$.
- Si G est sans cycle, alors $|A| \leqslant |S| - 1$.

Q. 3 En déduire l'équivalence entre les trois propriétés suivantes, pour $G = (S,A)$ un graphe non orienté :

- G est un arbre.
- G est connexe et $|A| = |S| - 1$.
- G est sans cycle et $|A| = |S| - 1$.

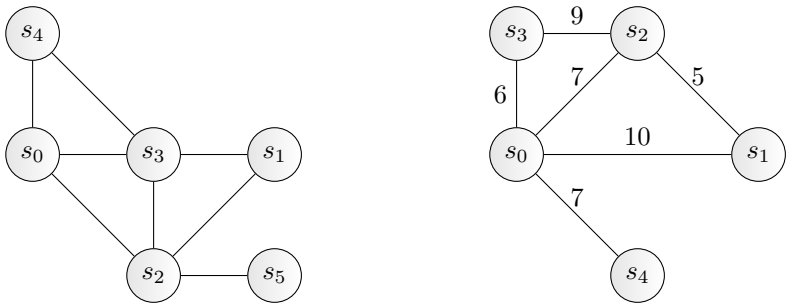

FIG. 1 – *Les graphes G_1 et G_2*

Pour les deux questions suivantes, on suppose que $G = (S,A,f)$ est un graphe pondéré tel que f est injective (toutes les arêtes ont un poids différent).

Q. 4 Justifier que si un graphe G est connexe alors il admet au moins un arbre couvrant et donc un arbre couvrant de poids minimal.

Q. 5 On va montrer que G possède un unique arbre couvrant de poids minimal. On va considérer par l'absurde qu'il en admet deux différents notés T_1 et T_2.

(a) Justifier qu'il existe une arête de T_1 qui n'appartient pas à T_2 et respectivement qu'il existe une arête de T_2 qui n'appartient pas à T_1. On fixera e_1 (resp. e_2) une telle arête de poids minimal.

(b) Supposons sans perte de généralité que $f(e_1) < f(e_2)$. Montrer qu'il existe e_3 une arête de T_2 qui n'appartient pas à T_1 telle que si on substitue e_1 à e_3 dans T_2 alors on a obtenu un arbre couvrant de poids strictement meilleur que celui de T_2.

(c) Conclure.

Q. 6 Soit $T^* = (S,B^*)$ l'unique arbre couvrant de poids minimal de G. On suppose que $a \in A$ est une arête dont le poids est maximal dans un cycle de G. Montrer que $a \notin B^*$.

Les trois questions qui suivent ont pour objectif d'implémenter un algorithme de tri efficace.

Q. 7 Écrire une fonction `separation` telle que si `lst` est une liste, alors `separation lst` renvoie un couple `(lst1, lst2)` tel que chaque élément de lst apparaît soit dans `lst1`, soit dans `lst2`, et les tailles de `lst1` et `lst2` diffèrent d'au plus 1.

```
separation : 'a list -> 'a list * 'a list
```

Q. 8 Écrire une fonction fusion telle que si `lst1` et `lst2` sont deux listes triées par ordre croissant, alors `fusion lst1 lst2` renvoie une liste triée par ordre croissant contenant tous les éléments de `lst1` et `lst2`.

```
fusion : 'a list -> 'a list -> 'a list
```

Q. 9 En déduire une fonction `tri_fusion` qui prend en argument une liste et renvoie une liste triée par ordre croissant contenant les mêmes éléments. Quelle est la complexité temporelle de cette fonction? (On ne demande pas de justifier.)

```
tri_fusion : 'a list -> 'a list
```

II Algorithme de Kruskal inversé

Q. 10 Rappeler le principe de l'algorithme de Kruskal et sa complexité temporelle en fonction de $|S|$ et $|A|$ lorsqu'il est appliqué à un graphe pondéré connexe $G = (S,A,f)$.

Entrées : Graphe pondéré $G = (S,A,f)$ non orienté connexe
$B \leftarrow$ copie de A
Trier B par ordre décroissant de poids
pour $a \in B$ *par ordre décroissant de poids* **faire**
  si $(S,B\setminus\{a\})$ *est connexe* **alors**
    $B \leftarrow B\setminus\{a\}$
  fin
fin
retourner (S,B)

Algorithme 1 : Algorithme de Kruskal

L'algorithme de Kruskal inversé est une variante de l'algorithme de Kruskal qui permet de trouver un arbre couvrant de poids minimal dans un graphe pondéré connexe. En voici une description en pseudo-code.

Q. 11 Appliquer l'algorithme de Kruskal inversé au graphe G_3 de la figure figure 2. On ne demande pas la description de l'exécution détaillée, mais simplement une représentation graphique de l'arbre obtenu.

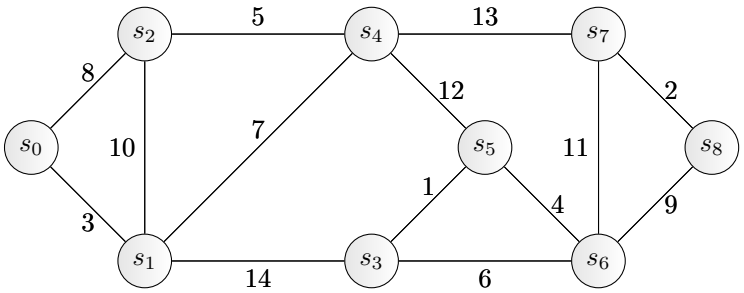

FIG. 2 – *le graphe pondéré* G_3

On implémente un graphe pondéré par tableau de listes d'adjacences en OCaml, selon le type suivant :

```
type graphe = (int * float) list array
```

Si $G = (S,A,f)$ est implémenté par un objet `g` de type graphe, alors :
- l'ensemble des sommets est $S = [\![0,n-1]\!]$, où $n = |S|$;
- `g` est un tableau de taille n tel que pour tout $s \in S$, g. (s) est une liste contenant des couples (t,p) tels que st $\in A$ et $f(st) = p$.

Les listes d'adjacence ne sont pas nécessairement supposées triées par ordre croissant des sommets ou des poids. Par exemple, le graphe G_2 de la figure 1 peut être créé par la commande :

```
let g2 = [|[(2,7.); (3,6.); (1,10.); (4,7.)]; [(2,5.); (0,10.)];
[(0,7.); (1,5.); (3,9.)]; [(0,6.); (2,9.)]; [(0,7.)] |]
```

On rappelle que les opérations de comparaisons (`max`, `min`, `<`, `>`, ...) peuvent s'effectuer sur des uplets selon l'ordre lexicographique.

Q. 12 Écrire une fonction `tri_aretes` telle que si `g` est la représentation d'un graphe pondéré $G = (S,A,f)$, alors `tri_aretes g` renvoie la liste des triplets $(f(st),s,t)$, triée par ordre décroissant des $f(st)$, tels que $st \in A$ et $s < t$. Cette liste ne devra pas contenir de doublons.

```
tri_aretes : graphe -> (float * int * int) list
```

Q. 13 Écrire une fonction `connectes` telle que si g est la représentation d'un graphe pondéré $G = (S,A,f)$, $(s,t) \in S^2$ et `poids_max` est un flottant, alors `connectes g s t poids_max` renvoie un booléen qui vaut `true` si et seulement s'il existe un chemin de s à t dans G ne passant que par des arêtes dont

le poids est strictement inférieur à `poids_max`. La fonction devra avoir une complexité linéaire en $|S| + |A|$ et on demande de justifier brièvement.

```
connectes : graphe -> int -> int -> float -> bool
```

Q. 14 En déduire une fonction `kruskal_inverse` qui prend en argument un graphe $G = (S,A,f)$ et renvoie le graphe obtenu selon le principe de l'algorithme de Kruskal inversé. On supposera que le graphe G donné en argument est connexe et que la fonction de pondération f est injective.

```
kruskal_inverse : graphe -> graphe
```

Q. 15 Montrer que si G est connexe, le graphe renvoyé par l'algorithme de Kruskal inversé appliqué à G est un arbre couvrant de G.

Q. 16 Montrer que l'arbre couvrant renvoyé par l'algorithme de Kruskal inversé appliqué à un graphe pondéré connexe G est de poids minimal. On pourra se contenter de traiter le cas où la fonction de pondération est injective. On pourra montrer l'invariant suivant : $B^* \subset B$ avec (S,B^*) l'unique arbre couvrant de poids minimal.

Q. 17 Déterminer la complexité temporelle de la fonction `kruskal_inverse`. Commenter.

III Difficulté de calcul de l'arbre optimal

Dans cette partie, on présente le problème du calcul de l'arbre optimal, appelé arbre de Steiner, et on veut montrer que c'est un problème difficile à résoudre.

Soit $G = (S,A)$ un graphe non orienté connexe. On appelle **arbre de Steiner de sommets terminaux X** un sous-graphe de G de la forme $T = (Y,B)$ tel que T est un arbre et $X \subset Y$. Les sommets de $Y \backslash X$ sont appelés les **sommets de Steiner**. Le problème de l'arbre de Steiner consiste, étant donné un graphe $G = (S,A)$, un ensemble $X \subset S$ et un entier k, à déterminer s'il existe un arbre de Steiner de sommets terminaux X ayant moins de k sommets de Steiner.

Q. 18 Représenter graphiquement un arbre de Steiner du graphe G_1 de la figure 1 , ayant $X = \{s_1,s_4,s_5\}$ comme sommets terminaux et ayant moins de 2 sommets de Steiner. Justifier qu'il n'en existe pas ayant moins de 1 sommet de Steiner.

Pour montrer la difficulté de ce problème, on se ramène au problème de satisfaisabilité d'une formule booléenne.

On rappelle qu'un **littéral** est une variable ou la négation d'une variable. Une **clause** est une disjonction ($\vee$) de littéraux. Une **forme normale conjonctive** (FNC) est une conjonction ($\wedge$) de clauses. Une formule φ est dite en `3-FNC` si elle est en forme normale conjonctive telle que chaque clause contient au plus trois littéraux.

On admet qu'étant donnée une formule propositionnelle φ en 3-FNC, il est difficile de savoir si φ possède un modèle, c'est-à-dire une valuation qui la satisfait. Ce problème est NP-complet.

III.A- De la satisfaisabilité au stable

Soit $G = (S,A)$ un graphe non orienté. On dit qu'une partie $X \subset S$ est un stable si c'est un ensemble de sommets deux à deux non adjacents, c'est-à-dire si pour $(s,t) \in X^2$, $st \notin A$.

Q. 19 Déterminer, en justifiant, un stable de cardinal maximal dans le graphe G_1 représenté figure 1.

Soit $\mathcal{V} = \{v_1,\dots,v_n\}$ un ensemble de variables et φ une formule booléenne en 3-FNC, de la forme $\varphi = \bigwedge_{j=1}^{m} C_j$, où $C_j = \bigvee_{k=1}^{m_j} \ell_{jk}$, les ℓ_{jk} étant des littéraux (de la forme v_i ou $\neg v_i$) et $m_j \in [\![1,3]\!]$.

On définit le graphe $G_\varphi = (S_\varphi,A_\varphi)$ par :

- S_φ contient un sommet par littéral apparaissant dans une clause, c'est-à-dire $S_\varphi = \bigcup_{j=1}^{m} S_j$ où :

$$S_j = \left\{ v_i^{(j)} \mid v_i \text{ apparait dans } C_j \right\} \cup \left\{ \overline{v}_i^{(j)} \mid \neg v_i \text{ apparait dans } C_j \right\}$$

- A_φ contient les arêtes entre chaque sommet associé à une même clause, et entre les sommets associés à une variable et sa négation :

$$A_\varphi = \left\{st \mid (s,t) \in S_j^2, s \neq t\right\} \cup \left\{v_i^{(j)}\overline{v}_i^{(j')} \mid i \in [\![1,n]\!], j \neq j'\right\}$$

La figure 3 représente le graphe G_{φ_0} pour $\varphi_0 = (v_1 \vee \neg v_2 \vee \neg v_3) \wedge (\neg v_1 \vee v_2) \wedge (v_1 \vee v_2 \vee v_3)$.

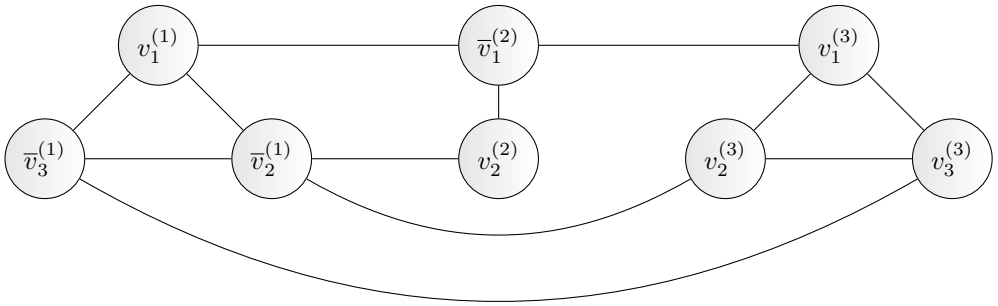

FIG. 3 – *Le graphe G_{φ_0} pour $\varphi_0 = (v_1 \vee \neg v_2 \vee \neg v_3) \wedge (\neg v_1 \vee v_2) \wedge (v_1 \vee v_2 \vee v_3)$.*

Q. 20 Tracer le graphe G_{φ_1} pour $\varphi_1 = (\neg v_0 \vee v_2) \wedge (v_0 \vee \neg v_1 \vee \neg v_2) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_0 \vee \neg v_1 \vee v_2)$.

Q. 21 Montrer que s'il existe un stable X de taille m dans G_φ, alors φ est satisfaisable. On expliquera comment construire un modèle de φ en fonction de X.

Q. 22 De même, montrer que si φ est satisfaisable, alors il existe un stable de taille m dans G_φ.

Q. 23 Montrer que le problème `STABLE` qui prend en entrée un graphe G et un entier m et qui renvoie `true` ssi G admet un stable de taille au moins m est dans la classe NP.

Q. 24 Montrer que le problème `STABLE` est NP-complet.

III.B- Du stable à l'arbre de Steiner

Si $G = (S,A)$ est un graphe non orienté, on pose $G' = (S_G,A_G)$ un graphe tel que :

- S_G contient S et un nouveau sommet par arête de G, c'est-à-dire :

$$S_G = S \cup \{s_a \mid a \in A\}$$

- A_G contient toutes les arêtes entre deux sommets de S, et des arêtes entre chaque sommet s_a vers chacune des extrémités de a, c'est-à-dire :

$$A_G = \left\{st \mid (s,t) \in S^2\right\} \cup \bigcup_{st \in A} \{ss_{st}, ts_{st}\}$$

Par exemple, la figure 6 est une représentation de G'_1 où G_1 est le graphe de la figure 1.

Q. 25 Montrer que s'il existe un stable de G de cardinal k, alors le graphe $G' = (S_G,A_G)$ admet un arbre de Steiner ayant $X = \{s_a \mid a \in A\}$ comme sommets terminaux et $|S| - k$ sommets de Steiner.

Q. 26 Réciproquement, montrer que si le graphe $G' = (S_G,A_G)$ admet un arbre de Steiner ayant $X = \{s_a \mid a \in A\}$ comme sommets terminaux et k sommets de Steiner, alors G admet un stable de cardinal $|S| - k$.

Q. 27 Donner la définition du problème de décision à seuil associé au problème d'optimisation suivant : trouver un arbre de Steiner ayant un nombre minimal de sommets de Steiner.

Q. 28 Justifier que le problème à seuil défini dans la question précédente est NP-complet.

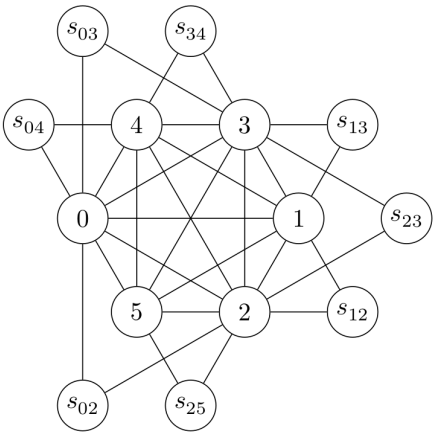

FIG. 4 – *Une représentation de G'_1. Pour simplifier, on a renommé chaque sommet s_i par i.*

IV Goulot maximal

Dans cette partie, on considère $G = (S,A,f)$ un graphe pondéré non orienté connexe.

Pour $\sigma = (s_0,s_1,\ldots,s_k)$ un chemin simple dans G, la **capacité** de σ, notée $c(\sigma)$ est le poids minimal de ses arêtes :

$$c(\sigma) = \min_{i=0}^{k-1} f(s_i s_{i+1})$$

Pour $s,t \in S^2$, le **goulot maximal** de s à t est un chemin de s à t de capacité maximale. S'il n'y a pas d'ambiguïté, on appellera également goulot maximal le poids de ce chemin.

Q. 29 Déterminer, sans justifier, un goulot maximal et sa capacité de s_0 à s_8 dans le graphe G_3 de la figure 2.

Q. 30 Expliquer en français comment calculer efficacement un arbre couvrant de poids **maximal** de G. Justifier succinctement sa correction.

Q. 31 Soit $(s,t) \in S^2$ et $T = (S,B)$ un arbre couvrant de poids **maximal** de G. Montrer qu'un chemin de s à t dans T est un goulot maximal de s à t dans G.

On représente un graphe $G = (S,A,f)$ en C par le type suivant :

```
struct Graphe {
    int n;
    int* degres;
    int** voisins;
    int** poids;
};

typedef struct Graphe graphe;
```

tel que si `G` est un objet de type `graphe` représentant G, alors :

- `g.n` $= n = |S|$ et $S = [\![0,n-1]\!]$;
- `G.degres` est un tableau de taille n tel que pour $s \in S$, `G.degres[s]` $= \deg(s)$;
- `G.voisins` est un tableau de taille n tel que pour $s \in S$, `G.voisins[s]` est un tableau de taille $\deg(s)$ contenant les voisins de s ;
- `G.poids` est un tableau de taille n tel que pour $s \in S$, `G.poids[s]` est un tableau de taille $\deg(s)$ contenant les poids des arêtes entre s et ses voisins. On garantit que pour $0 \leq i < \deg(s)$, si $t =$ `G.voisins[s][i]`, alors $f(st) =$ `G.poids[s][i]`.

Q. 32 Écrire une fonction `void liberer_graphe(graphe G)` qui libère l'espace mémoire occupé par un graphe.

Q. 33 Écrire une fonction `int* predecesseurs(graphe T, int s)` qui prend en argument un graphe $T = (S,B)$ qui est un arbre et un sommet $s \in T$ et renvoie un tableau d'entiers `pred` de taille $n = |S|$ tel que `pred[s] = -1` et pour $t \neq s$, `pred[t]` est le parent de t dans l'arbre T enraciné en s.
Cette fonction devra avoir une complexité linéaire en n et on demande de le justifier.

On suppose disposer d'une fonction `graphe arbre_max(graphe)` qui prend en argument un graphe et renvoie un arbre couvrant de poids maximal de ce graphe.

Q. 34 En déduire une fonction `int* goulot_max(graphe G, int s, int t, int* lc)` qui prend en argument un graphe et deux sommets s et t de ce graphe, ainsi qu'un pointeur vers un entier `lc` et renvoie un goulot maximal sous forme d'un tableau constitué des sommets de ce chemin. La fonction devra modifier l'entier pointé par `lc` pour y stocker la taille du chemin ainsi renvoyé.

V Approximation du problème

Étant donné un graphe pondéré $G = (S,A,f)$ et un sous-ensemble $X \subset S$, le **problème de l'arbre de Steiner** pondéré consiste à déterminer un sous-graphe $T = (Y,B,f)$ tel que $X \subset Y \subset S$,T est un arbre et $f(T)$ est minimal.

On peut voir que le problème de l'arbre de Steiner pondéré est plus difficile que le cas non pondéré, qui n'est qu'un cas particulier où tous les poids sont égaux à 1. Malgré sa difficulté, il est possible de calculer en temps polynomial un arbre de Steiner pondéré dont le poids est moins du double d'un arbre optimal.

En considérant $G = (S,A,f)$ un graphe pondéré non orienté connexe et $X \subset S$, la solution approchée pour le problème de l'arbre de Steiner est donnée par l'algorithme suivant :

- poser $H_1 = (X,\mathcal{P}_2(X),g)$: H_1 est un graphe complet entre les sommets de X, dont les pondérations sont données par les distances dans G (c'est-à-dire $g(st) = d_G(s,t)$) ;
- calculer $T_1 = (X,B_1,g)$ un arbre couvrant de poids minimal de H_1;
- pour chaque arête $st \in B_1 \backslash A$, la remplacer dans T_1 par une chaîne de poids minimal dans G, en rajoutant les sommets nécessaires. Le sous-graphe de G obtenu sera noté $H_2 = (Y,A_2,f)$.
- calculer et renvoyer $T = (Y,B,f)$ un arbre couvrant de poids minimal de H_2.

Q. 35 Montrer que l'arbre T est un arbre de Steiner de G ayant X pour sommets terminaux.

Q. 36 Montrer que si T possède des sommets de Steiner de degré 1, leur suppression permet d'obtenir un arbre de Steiner ayant X pour sommets terminaux de poids inférieur.

On pose T^* un arbre de Steiner de poids minimal parmi ceux ayant X comme sommets terminaux.

Q. 37 Montrer que $f(T) \leqslant g(T_1)$, puis en déduire que $f(T) \leqslant 2f(T^*)$.

• • • Fin • • •