TD Option informatique $n°3$ – Ordonnancement et coloration

I. Graphes d'intervalles

On considère le problème concret suivant : des cours doivent avoir lieu dans un intervalle de temps précis (de 8h à 9h55, ...) et on chercher à attribuer une salle à chaque cours. On souhaite qu'à tout moment une salle ne puisse être attribuée à deux cours différents et on aimerait utiliser le plus petit nombre de salles possibles.

Ce problème d'allocation de ressources (ici les salles) en fonction de besoins fixes (ici les horaires des cours) intervient dans de nombreuses situations très diverses (allocation de pistes d'atterrissage aux avions, répartition de la charge de travail sur plusieurs machines, ...).

I.A. Représentation du problème

On modélise le problème ainsi :

— chaque besoin est représenté par un segment $[a, b]$ où $a, b \in \mathbb{N}$ et $a \leqslant b$;

— deux besoins I et J sont en conflit lorsqu'on a $I \cap J \neq \emptyset$.

La donnée du problème est une suite finnie $(I_0, \ldots, I_{n-1})$ de n segments où $n \in \mathbb{N}^*$.

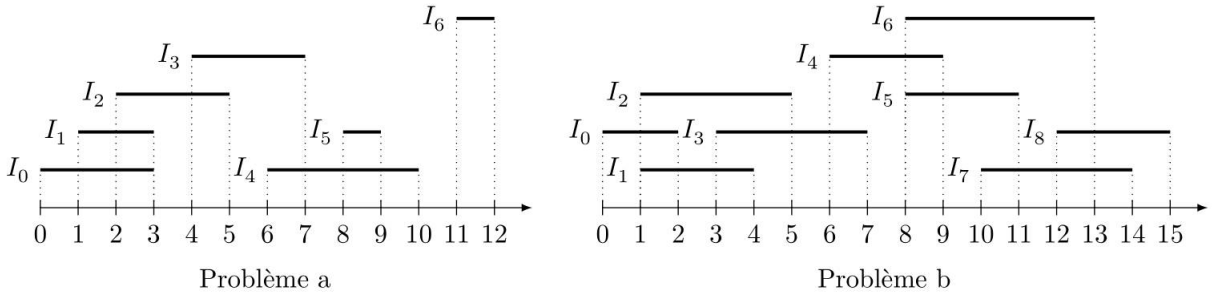


Figure 1 Deux exemples de problèmes

On représente un segment en Caml par un coupe d'entiers, la donnée du problème est une valeur du type `(int*int) array`. Le *problème a* de la **Figure 1** est représenté par le tableau suivant : `[|(0,3); (1,3); (4,7); (6,10); (8,9); (11,12)|]`.

1. Écrire une fonction `conflit : int*int → int*int → bool` telle que `conflit i j` renvoie `true` si et seulement si `i` et `j` sont en conflit.

I.B. Graphe d'intervalles

Les graphes considérés dans la suite sont non orientés et sans boucle, et leurs sommets sont étiquetés par les entiers de 0 à $n-1$, où n est son nombre de sommets. On les représente par listes d'adjacence. Par exemple, le graphe représenté en CAML par le tableau `[| [1;2;3], [0;2;3]; [0;1;3;4]; [0;1;2]; [2] |]` est représenté graphiquement dans la **Figure 2**.

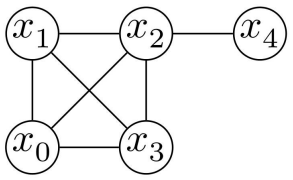


Figure 2

Dans toute la suite, on utilisera le type `graphe` suivant :

```
type graphe = int list array
```

Pour $\overline{I} = (I_0, \ldots, I_{n-1})$ une suite finie de segments, on appelle graphe d'intervalles associé à $\overline{I}$ le graphe $G(\overline{I})$

— dont les sommets sont les segments $I_0, \ldots, I_{n-1}$

— et pour lequel il existe une arête entre I_p et I_q, lorsque $p \neq q \in \{0, \ldots, n-1\}$ sont tels que I_p et I_q sont en conflit.

Le graphe d'intervalles correspondant au *problème a* de la **Figure 1** se représente graphiquement sur la **Figure 3**.

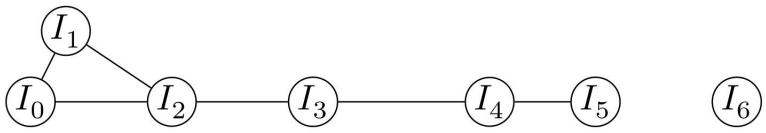


Figure 3

2. Donner une représentation graphique du graphe d'intervalles associé au *problème b* de la **Figure 1**.

3. Écrire une fonction `construit_graphe : (int*int) array → graphe` qui étant donné la suite de segments $\overline{I} = (I_0, \ldots, I_{n-1})$ énumérés dans cet ordre, et représentée en Caml par le tableau `[|i0;...;in-1|]`, renvoie la représentation en Caml du graphe $G(\overline{I})$.

I.C Coloration

Soit $G = (S, A)$ un graphe dont les sommets x_0, …, x_{n-1} sont représentés en Caml par les entiers 0, …, $n-1$.

On appelle coloration de G une suite finie d'entiers naturels $(c_0, \ldots, c_{n-1})$ telle que

$$\forall i, j \in \{0, \ldots, n-1\}^2,\ \{x_i, x_j\} \in A \Rightarrow c_i \neq c_j.$$

L'entier c_i est appelé la couleur du sommet x_i et la condition se traduit par le fait que deux sommets reliés ont toujours des couleurs distinctes. Dorénavant, le terme couleur est synonyme d'entier naturel.

La suite finie $(0, 1, 2, 3, 0)$ est une coloration du graphe de la **Figure 2**.

Lorsqu'une coloration utilise le plus petit nombre de couleurs distinctes possible, on dit qu'elle est optimale. On note alors $\chi(G)$ ce nombre minimum de couleurs, appelé nombre chromatique de G.

En associant une salle à chaque couleur, on peut répondre au problème inital à l'aide d'une coloration de son graphe d'intervalles associé.

4. Déterminer des colorations optimales pour les graphes d'intervalles associés aux deux problèmes de la **Figure 1**. On attribuera à chaque fois la couleur 0 à l'intervalle I_0.

5. *Couleur disponible.*
   (a) Écrire une fonction `appartient : int list → int → bool` telle que l'appel à `appartient l x` renvoie `true` si et seulement si l'entier `x` est présent dans la liste `l`. *Oui, je suis au courant que cette fonction existe déjà en* Caml *! Manifestement ces gens nous demandent de la réécrire...*
   (b) Écrire une fonction `plus_petit_absent : int list → int` telle que l'appel à `plus_petit_absent l` renvoie le plus petit entier naturel non présent dans `l`.
   (c) On considère ici une coloration progressive des sommets d'un graphe. Pour cela, une coloration partielle est un tableau `couleurs : int array` tel que `couleurs.(i)` contient la couleur de `i` s'il est coloré et `-1` sinon, ce qui ne pose pas de problème car les couleurs sont toujours positives.
   Écrire une fonction `couleur_voisins : graphe → int array → int → int list` telle que l'appel à `couleur_voisins g couleurs i` renvoie la liste des couleurs des voisins colorés du sommet d'indice `i` dans le graphe `g` où le tableau `couleurs` décrit une coloration partielle.
   (d) En déduire une fonction `couleur_disponible : graphe → int array → int → int list` telle que l'appel à `couleur_disponible g couleurs i` renvoie la plus petite couleur pouvant être attribuée au sommet `i` afin qu'il n'ait la couleur d'aucun de ses voisins dans le graphe `g`.

I.E. Cliques

Soit $G = (S, A)$ un graphe. Une partie $C \subset S$ est appelée une clique de G lorsqu'elle vérifie

$$\forall x, y \in C,\ x \neq y \Rightarrow \{x, y\} \in A$$

Autrement dit, une clique est l'ensemble des sommets d'un sous-graphe complet de G.

Le nombre d'éléments de C est appelé sa taille. La taille d'une plus grande clique de G (une clique qui possède le plus grand nombre d'éléments possible) est notée $\omega(G)$.

6. Déterminer $\chi(G)$ et $\omega(G)$ lorsque :
   (a) G ne possède pas d'arête (c'est-à-dire $A = \emptyset$).
   (b) G est un graphe complet à n sommets, c'est-à-dire $|S| = n$ et $\forall u, v \in S,\ \{u, v\} \in A$.

7. Comparer $\chi(G)$ et $\omega(G)$ pour un graphe G quelconque.

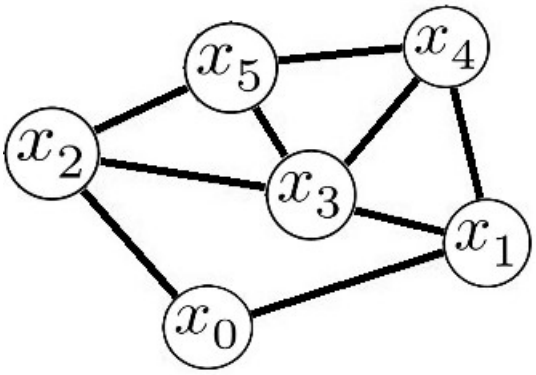
8. Écrire une fonction `est_clique : graphe → int list → bool` telle que `est_clique g xs` renvoie `true` si et seulement si la liste `xs` est une liste d'indices de sommets formant une clique dans le graphe `g`.

II. Algorithme glouton pour la coloration

Soit $G = (S, A)$ un graphe dont les sommets $x_0, \ldots, x_{n-1}$ sont représentés en CAML par les entiers $0, \ldots, n-1$. On propose l'algorithme suivant, qu'on appellera dans la suite "l'algorithme glouton" :
pour k variant de 0 à $n-1$, on colore le sommet x_k avec la plus petite couleur non encore utilisée dans la coloration des sommets x_j avec $0 \leqslant j < k$ qui sont voisins de x_k.

II.A. Deux exemples.

9. Déterminer la coloration renvoyée par l'algorithme glouton pour le graphe ci-dessous :



10. La coloration obtenue sur l'exemple précédent est-elle optimale ?

11. Déterminer la coloration donnée par l'algorithme pour le graphe d'intervalles du *problème b* de la **Figure 1**.

II.B. Implémentation.

12. Écrire une fonction `colore : graphe → int array` telle que l'appel à `colore g` détermine la coloration du graphe `g` obtenue en appliquant l'algorithme glouton et la renvoie sous la forme d'un tableau `c` où, pour chaque i compris entre 0 et $n-1$, l'entier `c.(i)` indique la couleur du $i^{\text{ième}}$ sommet.

II.C. Correction et complexité de l'algorithme.

On se propose maintenant de démontrer que lorsqu'il est appliqué à un graphe d'intervalles dont les sommets sont triés dans l'ordre croissant de leurs extrémités gauches, l'algorithme glouton fournit une coloration optimale de l'ensemble des segments. Soit k un entier entre 0 et $n-1$. On suppose qu'à la $k^{\text{ième}}$ étape de l'algorithme, le segment I_k reçoit la couleur c.

13. L'extrémité gauche du segment I_k appartient à un certain nombre de segments parmi I_0, I_1, …, I_{k-1}. Combien au moins ?

14. Prouver que l'ensemble constitué des I_k et ses voisins d'indice inférieur à k constitue une clique de taille au moins $c+1$ dans le graphe d'intervalles associé.

15. En déduire que le nombre de couleurs nécessaires à une coloration de l'ensemble des segments est au moins égal à $c+1$.

16. Conclure sur la correction de l'algorithme.

17. Déterminer la complexité de la fonction `coloration` en fonction du nombre m d'arêtes du graphe d'intervalles associé à la suite $\overline{I}$ et du nombre n d'intervalles (*i.e.* de sommets du graphe).

III. Graphes munis d'un ordre d'élimination parfait.

On introduit ici la notion d'ordre d'élimination parfait, dont on montre qu'il existe toujours pour un graphe d'intervalles, et qui permet de proposer un algorithme glouton pour le problème de la coloration d'un graphe.

Soient $G=(S,A)$ un graphe et $(x_0,\dots,x_{n-1})$ une énumération des sommets de G. Pour tout $i\in\{0,\dots,n-1\}$, on note $G_i=(S_i,A_i)$ où $S_i=\{x_0,\dots,x_i\}$ et $\forall p,q\in\{0,\dots,n-1\},\ \{x_p,x_q\}\in A_i \Rightarrow p\leqslant i$ et $q\leqslant i$ et $\{x_p,x_q\}\in A$.

Ainsi, G_i est le graphe déduit de G en se restreignant aux sommets de x_0 à x_i.

Une énumération $(x_0,\dots,x_{n-1})$ des sommets de G est appelée un ordre d'élimination parfait si pour tout $i\in\{0,\dots,n-1\}$ les voisins de x_i d'indices inférieurs à i forment une clique.

III.A. Un exemple.

18. Déterminer un ordre d'élimination parfait pour le graphe G donné par la représentation de la **Figure 4** :

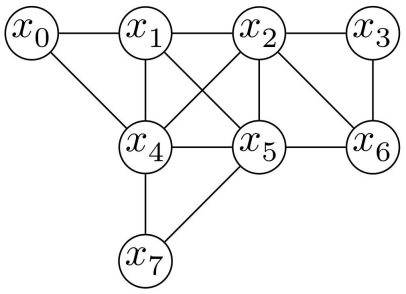


Figure 4

III.B. Vérification.

19. Écrire une fonction `voisins_inferieurs : graphe → int → int list` telle que `voisins_inferieurs g x` renvoie la liste des voisins du sommet d'indice `x` dont l'indice est strictement inférieur à `x`.

20. Écrire une fonction `est_ordre_parfait : graphe → bool` telle que `est_ordre_parfait g` renvoie `true` si et seulement si l'énumération associée au graphe `g` est un ordre d'élimination parfait.

III.C. Implémentation brutale de la recherche d'un ordre d'élimination parfait.

On considère une énumération partielle des sommets $(x_0, x_1, \dots, x_{p-1})$ d'un graphe $G = (S, A)$. Ainsi l'ensemble $\{x_0, x_1, \dots, x_{p-1}\}$ est une partie de S.

On dira d'une telle énumération partielle qu'il s'agit d'un ordre d'élimination parfait partiel ssi pour tout $i \in \{0, \dots, p-1\}$ les voisins de x_i d'indices inférieurs à i forment une clique.

Il est évident qu'un ordre d'élimination parfait n'est autre qu'un ordre d'élimination parfait partiel de longueur n, et que tout préfixe d'un ordre d'élimination parfait (partiel ou non) est un ordre d'élimination partiel.

Une énumération partielle sera représentée par un tableau d'entiers de longueur n, dont l'élément en place i vaudra x_i si $i < p$ et -1 sinon (c'est-à-dire si $p \leqslant i \leqslant n-1$).

21. Écrire une fonction `successeurs: graphe → int → int array → int array` telle que l'exécution de `successeurs g p oepp` où `g` est un graphe, où `p` est un entier et où `oepp` est un ordre d'élimination parfait partiel $(x_0, x_1, \dots, x_{p-1})$ renvoie la liste des sommets y tels que $(x_0, x_1, \dots, x_{p-1}, y)$ soit encore un ordre d'élimination parfait partiel.

22. Écrire une fonction récursive `prolonger_oepp : graphe → int → int array → int array list` telle que l'appel `prolonger_oepp g p oepp` renvoie la liste des ordres d'élimination parfaits du graphe `g` prolongeant l'ordre d'élimination parfait partiel `oepp`.

23. En déduire une fonction `lister_ordres_parfaits : graphe → int array list` renvoyant la liste des ordres d'élimination parfaits du graphe `g`.

III.D. Ordre d'élimination parfait pour un graphe d'intervalles.

24. Montrer que l'énumération des segments $(I_0, \dots, I_{n-1})$ obtenue en les triant par leur extrémités gauches en ordre croissant est un ordre d'élimination parfait de leur graphe d'intervalles.

III.E. Coloration.

On considère un graphe dont $(x_0, \dots, x_{n-1})$ est une énumération des sommets.

On colore ce graphe à l'aide de l'algorithme glouton vu précédemment, c'est-à-dire que pour i allant de 0 à $n-1$ on colore x_i avec la plus petite couleur qui ne soit pas utilisée par un de ses voisins déjà colorés.

25. Appliquer cet algorithme de coloration au graphe G de la **Figure 4** muni :
    (a) de l'ordre $(x_0, \dots, x_7)$;
    (b) d'un ordre d'élimination parfait.

26. Montrer que si l'énumération des sommets utilisée est un ordre d'élimination parfait, alors l'algorithme glouton donne une coloration optimale.

IV. Graphe cordal.

On s'intéresse ici à une nouvelle condition nécessaire et suffisante pour qu'un graphe admette un ordre d'élimination parfait qui s'exprime en considérant les cycles de longueur au moins égale à 4 du graphe considéré.

Un graphe G est dit cordal lorsque pour tout cycle $C = (v_0, v_1, \ldots, v_{n-1}, v_0)$ de G de longueur $n \geqslant 4$, il existe i, j distincts entre 0 et $n-1$ tels que les sommets v_i et v_j soient reliés dans le graphe G mais non successifs dans le cycle. Une telle arête $\{v_i, v_j\}$ est appelée une corde du cycle C. Autrement dit, le graphe G est cordal lorsque tout cycle de G de longueur supérieure ou égale à 4 possède une corde.

IV.A. Sens direct

27. Justifier qu'un graphe ayant un ordre d'élimination parfait est toujours cordal.

Dans la suite, on va s'intéresser à la réciproque.

IV.B. Cycles de longueur 4 dans un graphe d'intervalles

Soit G un graphe d'intervalles. Dans cette question, on se propose de démontrer par l'absurde que tout cycle de longueur 4 de G possède une corde. On suppose à cet effet que G contient un 4-cycle sans corde.

On dispose donc de 4 segments I_0, I_1, I_2, I_3 tels que $I_0 \cap I_1 \neq \emptyset$, $I_1 \cap I_2 \neq \emptyset$, $I_2 \cap I_3 \neq \emptyset$, $I_3 \cap I_0 \neq \emptyset$, et $I_0 \cap I_2 \neq \emptyset$, $I_1 \cap I_3 \neq \emptyset$. On supposera pour simplifier que les extrémités des segments sont toutes distinctes.

28. Montrer qu'aucun des segments I_k pour $k \in \{0, 1, 2, 3\}$ n'est inclus dans un autre de ces segments.

29. Justifier qu'à renommage près, on a I_0 et I_2 disjoints, de même que I_1 et I_3, puis qu'on a $\min I_0 < \min I_1 < \max I_0 < \max I_1$.

30. Montrer qu'on a alors $\min I_1 < \min I_2 < \max I_1 < \max I_2$.
Donner les inégalités correspondantes pour I_2 et I_3.

31. Conclure à une contradiction.

IV.C. Cordalité des graphes d'intervalles

32. Montrer plus généralement que tout graphe d'intervalles est cordal.

33. On va montrer que la réciproque est fausse.
    (a) Justifier qu'un arbre (au sens graphe connexe sans cycle) est toujours un graphe cordal.
    (b) Dans un graphe d'intervalles, montrer que si J_1, J_2 et J_3 sont des segments deux à deux disjoints mais voisins d'un même segment I, alors l'un au moins d'entre eux est inclus dans I.
    (c) Trouver un graphe cordal qui ne soit pas un graphe d'intervalles.

IV.D. Parenthèse : une enquête policière

Six personnes sont entrées dans la bibliothèque le jour où un livre rare y a été volé. Chacune d'entre elles est entrée une seule fois dans la bibliothèque, y est restée un certain temps, puis elle en est sortie. Si deux personnes étaient ensemble dans la bibiothèque à un instant donné, alors au moins l'un des deux a vu l'autre. À l'issue de l'enquête, les témoignages recueillis sont les suivants : Albert dit qu'il a vu Bernard et Édouard dans la bibliothèque. Bernard a vu Albert et Fanny. Charlotte affirme avoir vu Didier et Fanny. Didier dit qu'il a vu Albert et Fanny. Édouard certifie avoir vu Bernard et Charlotte. Fanny dit avoir vu Charlotte et Édouard.

Seul le coupable a menti, en affirmant avoir vu quelqu'un qui n'était pas là.

34. Qui est le coupable ?

IV.E. Sommets simpliciaux et ordre d'élimination parfait.

Un sommet v d'un graphe G est dit simplicial lorsque l'ensemble des voisins de v est une clique.

Étant donnés un graphe $G = (S, A)$ et $S' \subset S$ un ensemble de sommets de G, le sous-graphe de G induit par S' est le graphe $H = (S', A')$ où $Q' \subset A$ est l'ensemble des arêtes de G dont les extrémités appartiennent à S'.

On représente en CAML un sous-graphe induit d'un graphe G possédant n sommets par le couple `(g,sg)`de type `graphe*bool array` où `g` est une description du graphe G et `sg` un tableau de taille `n` tel que `sg.(i)` vaut `true` si le sommet d'indice `i` est un sommet du sous-graphe induit et `false` sinon.

35. Justifier que si un sommet y du graphe $G = (S, A)$ est simplicial, et si $(x_0, x_1, \ldots, x_{n-2})$ est un ordre d'élimination parfait pour le sous-graphe de G induit par $S \setminus \{y\}$, alors $(x_0, x_1, \ldots, x_{n-2}, y)$ est un ordre d'élimination parfait pour le graphe G.

36. En déduire un algorithme permettant de déterminer un ordre d'élimination parfait pour un graphe G dont on suppose que tout sous-graphe induit admet au moins un sommet simplicial.
37. Écrire une fonction `simplicial : (graphe*bool array) → int → bool` telle que l'appel à `simplicial (g,sg) k` où le sommet d'indice `k` est supposé appartenir au sous-graphe induit J décrit par `(g,sg)`, renvoie `true` si le sommet d'indice `k` est simplicial dans H et `false` sinon.

38. Déterminer la complexité de la fonction `simplicial`.

39. Écrire une fonction `trouver_simplicial : (graphe*bool array) → int` telle que l'appel à `trouver_simplicial (g,sg)` renvoie, s'il en existe, un sommet simplicial du sous-graphe induit décrit par `(g,sg)`, et provoque une exception dans le cas contraire.

40. Déterminer la complexité de la fonction `trouver_simplicial`.

41. On souhaite écrire une fonction `ordre_parfait : graphe → int list` telle que l'appel à `ordre_parfait g` renvoie un ordre d'élimination parfait du graphe `g` s'il en existe un. On implémentera l'algorithme donné précédemment permettant de déterminer un ordre d'élimination parfait pour un graphe G dont on suppose que tout sous-graphe induit admet au moins un sommet simplicial (ce qui ne nous fait pas perdre en généralité, car on prouvera plus loin que c'est bien le cas de tout graphe admettant un ordre d'élimination parfait).
42. Déterminer la complexité de la fonction `ordre_parfait`.

IV.F. Coupures minimales dans un graphe cordal.

Étant donné un graphe G on appelle coupure de G tout ensemble $C \subset S$ de sommets de G telle que certains sommets reliés par un chemin dans le graphe G ne le sont plus dans le sous-graphe de G induit par $S \setminus C$.

Autrement dit, c'est un ensemble de sommets qui, lorsqu'on le retire du graphe, augmente le nombre de composantes connexes du graphe.

On se donne dans cette question un graphe cordal $G = (S, A)$. Soit C une coupure de G de cardinal minimal dont on suppose qu'il est supérieur ou égal à 2. Soit H le sous-graphe de G induit par $S \setminus C$. Soient a et b deux sommets de G déconnectés par la coupure, et soient G_1 et G_2 les composantes connexes de a et b dans le graphe H. Soient enfin x et y deux sommets distincts de la coupure C.

43. Montrer que x est voisin dans le graphe G d'un sommet de G_1 et d'un sommet de G_2, et de même pour y.

44. Montrer qu'il existe un chemin $P_1 = (x, a_1, \ldots, a_p, y)$ dont tous les sommets hormis x et y sont des sommets de G_1, et un chemin $P_2 = (y, b_1, \ldots, b_p, x)$ dont tous les sommets hormis x et y sont des sommets de G_2.

45. On prend deux tels chemins P_1 et P_2 de longueur minimale. En considérant un cycle formé à partir des chemins P_1 et P_2, montrer que x et y sont reliés dans le graphe G.

46. Montrer que C est une clique du graphe G.

IV.G. Sommets simpliciaux dans un graphe cordal

On se propose de montrer que tout graphe cordal G possède la propriété $\mathcal{P}(G)$ suivante :

$\mathcal{P}(G)$: G possède un sommet simplicial, et même deux sommets simpliciaux non voisins si G n'est pas complet.

On se donne dans les questions 47 à 49 un graphe cordal G.

47. Montrer que si G est complet alors tous ses sommets sont simpliciaux.

48. Montrer que la propriété $\mathcal{P}(G)$ est vérifiée si G possède 1, 2 ou 3 sommets.

49. On suppose dans cette question que G n'est pas complet et possède au moins trois sommets, et que la propriété $\mathcal{P}(G')$ est vérifiée pour tous les graphes cordaux G' ayant strictement moins de sommets que G. Soit C une coupure de G de cardinal minimal.Soient a et b deux sommets de G déconnectés par la coupure C, et G_1 et G_2 les composantes connexes de a et b dans le sous-graphe de G induit par $S \setminus C$. Soit S_1 (respectivement S_2) l'ensemble des sommets de G_1 (respectivement G_2). Soit enfin H_1 (respectivement H_2) le sous-graphe de G induit par $S_1 \cup C$ (respectivement $S_2 \cup C$).
    (a) Justifier que H_1 est cordal.
    (b) On suppose que H_1 est complet. Montrer que S_1 contient un sommet simplicial du graphe H_1. Prouver que ce sommet est aussi simplicial comme sommet du graphe G.
    (c) On suppose que H_1 n'est pas complet. Montrer que $S_1 \cup C$ contient deux sommets simpliciaux non voisins du graphe H_1. Montrer qu'au moins l'un de ces deux sommets est dans S_1 et que ce sommet est aussi simplicial comme sommet du graphe G.
    (d) en déduire $\mathcal{P}(G)$.

50. Finalement, justifier que tout graphe cordal G vérifie la propriété $\mathcal{P}(G)$.

IV.H. Conclusion

51. Montrer que tout graphe cordal possède un ordre d'élimination parfait.