Informatique I

Correction.

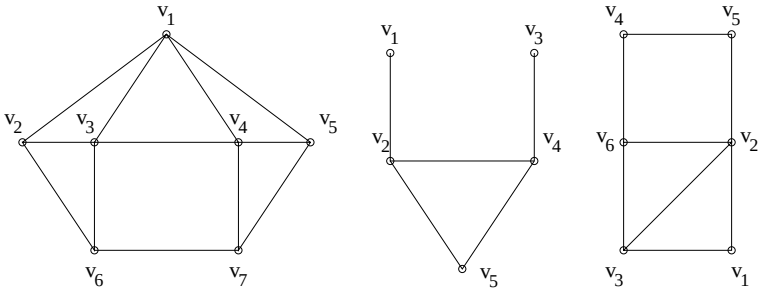
Le problème porte sur le problème de coloriage de graphes, et une application à la sécurité informatique.

Notations et définitions

La notation $\hat{=}$ désigne l'*égalité par définition*, pour la distinguer de l'égalité $=$.

Dans tout le problème, un *graphe* désigne un graphe non orienté, c'est-à-dire un couple $G\hat{=}(V, E)$, où V est un ensemble fini de *sommets*, et E est un ensemble de paires $\{v_1, v_2\}$ de sommets appelées *arêtes*. Si $\{v_1, v_2\} \in E$, on dit que v_1 et v_2 sont *adjacents*, et qu'ils sont les *extrémités* de l'arête $\{v_1, v_2\}$. Le *degré* $d_G(v)$ d'un sommet v de G est le nombre de sommets adjacents à v dans G. Le *degré* $d(G)$ du graphe G est $\max_{v \in V} d_G(v)$.

On notera souvent les graphes par des dessins, par exemple :



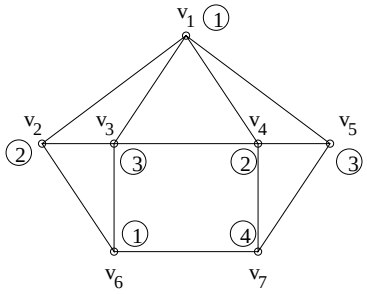
Pour tout entier $k \in \mathbb{N}$, un *k-coloriage* de G est une application c de V vers $[1, k]$ telle que $c(v_1) \neq c(v_2)$ pour tous sommets adjacents v_1, v_2. De façon imagée, on appelle $c(v)$ la *couleur* du sommet v de G dans c. Si G a un k-coloriage, on dit que G est *k-colorable*. Le plus petit entier k tel que G est k-colorable est le *nombre chromatique* de G.

Pour tout ensemble fini S, $|S|$ désigne le cardinal de S.

1 Algorithmes de coloriage

1. Montrer que le nombre chromatique du graphe [1] est exactement 4.

   Un 4-coloriage possible est le suivant, où on a indiqué les couleurs dans des bulles circulaires :



   Réciproquement, il s'agit de montrer qu'on ne peut pas colorier [1] avec trois couleurs. Supposons le contraire. Les sommets v_1, v_3 et v_4 doivent avoir des couleurs différentes deux à deux. Sans perte de généralité, supposons que la couleur de v_1 est 1, celle de v_3 est 3, celle de v_4 est 2. Comme la couleur de v_2 est différente de celles de v_1 et de v_3, c'est nécessairement 2. De même, celle de v_4 est nécessairement 3. Comme la couleur de v_6 est différente de celles de v_2 et de v_3, c'est 1. De même, la couleur de v_7 est différente de celles de v_4 et de v_5, c'est donc 1 aussi. Mais ceci contredit le fait que les couleurs de v_6 et de v_7 devraient être différentes.

2. On considère l'algorithme glouton suivant, où les sommets de $G \hat{=} (V, E)$ sont supposés numérotés de 1 à $|V|$, et $V[i]$ est la liste des sommets adjacents au sommet i, et où c est un tableau de $|V|$ entrées :

```
1 fonction COL (V, E) {
2         couleur := 0; coloriés := 0;
3         pour i = 1..|V| { c[i] := 0; }
4         tant que coloriés < |V| {
5               couleur := couleur+1;
6               pour i = 1..|V| {
7                     si c[i] = 0 et pour tout j ∈ V[i] c[j] ≠ couleur
8                        alors { c[i] :=couleur; coloriés := coloriés +1; }
9                     }
10              }
11        retourner (couleur, c);
```

12 }

L'idée est que pour tout graphe (V, E), COL (V, E) retourne un couple (k, c) tel que c est un k-coloriage de (V, E). Pour chacun des graphes (V, E) dessinés plus haut, [1], [2] et [3], calculer COL (V, E). On supposera que les sommets sont listés dans l'ordre v_1, v_2, ..., comme indiqué sur la figure.

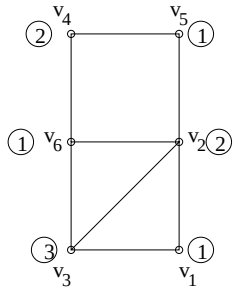
Pour [1], on trouve v_1 et v_6 de couleur 1, v_2 et v_4 de couleur 2, v_3 et v_5 de couleur 3, et v_7 de couleur 4, donnant $k = 4$.
Pour [2], on trouve v_1, v_3 et v_5 de couleur 1, v_2 de couleur 2, et v_3 de couleur 3, donnant $k = 3$.
Pour [3], on trouve v_1 et v_4 de couleur 1, v_2 de couleur 2, v_3 et v_5 de couleur 3, et v_6 de couleur 4, donnant $k = 4$.

3. Montrer que COL (G) termine en temps polynomial en la taille $|V| + |E|$ de $G\hat{=}(V, E)$, et retourne effectivement une paire (k, c) où c est un k-coloriage de G.

On note que le fait que $c[i] = 0$ si et seulement si le sommet i n'a pas encore reçu de couleur. Un invariant de l'algorithme après la boucle d'initialisation de la ligne 3 est que coloriés est le nombre d'entrées $c[i]$ dans le tableau c qui sont différentes de 0, c'est-à-dire le nombre de sommets déjà coloriés. Maintenant, dans la boucle interne 6–8, le test de la ligne 7 réussit au moins une fois. En effet, si on rentre dans la boucle, c'est que coloriés $< |V|$, donc qu'il existe au moins un i entre 1 et $|V|$ tel que $c[i] = 0$; et le premier i tel que $c[i] = 0$ est nécessairement tel que pour tout $j \in V[i]$, $c[j] \neq$couleur. En effet, à ce point tous les sommets sont coloriés par des entiers $<$ couleur. Donc la boucle 6–8 colorie au moins un nouveau sommet. Donc la boucle 4–10 est exécutée au plus $|V|$ fois. En supposant que l'on a calculé $|V|$ en temps $O(|V|)$ une seule fois au début de l'algorithme, le temps d'exécution de l'algorithme est $O(|V| + |V|^2|E|)$, qui est au plus cubique en la taille de G. (On peut faire mieux, mais ce n'est pas demandé.)
Un autre invariant de l'algorithme (lignes 4–11) est que $c[i]$ est, pour tout i, un entier entre 0 et couleur, et que $c[i] \neq c[j]$ dès que i et j sont adjacents. Comme en ligne 11 coloriés$= |V|$, aucun i n'est tel que $c[i] = 0$, donc c est un k-coloriage avec $k =$couleur.

4. En s'aidant de la question 2, montrer que COL n'est pas optimal, autrement dit il existe un graphe G, un entier k et un k-coloriage c de G tels que COL $(G) = (k, c)$ et k est strictement supérieur au nombre chromatique de G.

Considérons le graphe [3]. Il est 3-colorable:



mais on a vu à la question précédente que l'algorithme COL *donnait $k = 4$.*
Noter cependant que COL *est optimal sur le graphe [1]. Si l'on représente [1] par:*

i	$V[i]$
1	2, 3, 4, 5
2	1, 3, 6
3	1, 2, 4, 6
4	1, 3, 5, 7
5	1, 4, 7
6	2, 3, 7
7	4, 5, 6

alors COL *attribue la couleur* 1 *à* 1 *et* 6, *puis la couleur* 2 *à* 2 *et* 4, *puis la couleur* 3 *à* 3 *et* 5, *puis la couleur* 4 *à* 7.
De même, il est optimal sur [2] (exercice).

5. Montrer cependant qu'il existe toujours une permutation π de $\{1,\ldots,|V|\}$ tel que COL retourne un résultat optimal sur le graphe permuté $\pi(G)$. (Formellement, $\pi(G)$ est le graphe dont les sommets sont ceux de G, et dont les arêtes sont les $\{\pi(v),\pi(v')\}$ lorsque $\{v,v'\}$ parcourt les arêtes de G.)

   Soit k le nombre chromatique de G, et c un k-coloriage de G. On peut toujours permuter les sommets de G de sorte que les sommets de couleur 1 *viennent en premier, ceux de couleur* 2 *en second, ..., ceux de couleur k en dernier. Il est intuitif que si l'on fournit le graphe permuté à* COL, COL *fournira exactement le coloriage c. Formellement, il est relativement facile de voir que lorsque la variable couleur vaut p dans* COL, *à la ligne 9, alors au moins tous les sommets de couleurs $\leq p$ dans c ont déjà été coloriés de couleurs $\leq p$. (Par contre, il est en général faux de réclamer que tout sommet sera colorié par la couleur qu'il a dans c.) Il s'ensuit lorsque $p = k$, que* COL *terminera en ayant trouvé un k-coloriage de G.*

6. Écrire un algorithme énumérant toutes les permutations sur $\{1,\ldots,|V|\}$ sans répétition. À titre d'indication, on montrera que toute permutation π sur $\{1,\ldots,N\}$ s'écrit de façon unique comme la composée $(N,i_N)\circ(N-1,i_{N-1})\circ\ldots\circ(2,i_2)\circ(1,i_1)$, où $1\leq i_1\leq N$, $2\leq i_2\leq N$, ..., $N\leq i_n\leq N$, et où la notation (i,j) dénote la transposition qui échange i et j si $i\neq j$, et la fonction identité sinon.

   On démontre le lemme sur la présentation de π par récurrence sur N. Si $N=0$, c'est évident : π est l'identité, et s'écrit de façon unique comme la composée de 0 *transposition. Sinon, soit $i_1\hat{=}\pi^{-1}(1)$. Alors $\pi\circ(1,i_1)$ est une permutation de $\{1,\ldots,N\}$ qui envoie* 1 *vers* 1. *Par hypothèse de récurrence, la restriction de $\pi\circ(1,i_1)$ à $\{2,\ldots,N\}$ s'écrit de façon unique $(N,i_N)\circ(N-1,i_{N-1})\circ\ldots\circ(2,i_2)$, où $2\leq i_2\leq N$, ..., $N\leq i_n\leq N$. Donc, comme $(1,i_1)$ est son propre inverse, π s'écrit de façon unique sous la forme prescrite.*
   Par ce lemme, il suffit d'énumérer toutes les valeurs possibles pour i_1 (de 1 *à N) d'abord, puis toutes celles de i_2 (de* 2 *à N), et ainsi de suite jusqu'à i_N. Ceci se fait aisément récursivement :*

```
1 fonction ENUMERE_PERMS (p, m, n, f) {
2         si m ≥ n
3           alors f(p)
4         sinon pour tout i = m..n { (* i énumère les i_m possibles. *)
5                   aux := p[m]; p[m] := p[i]; p[i] := aux;
6                   ENUMERE_PERMS (p, m + 1, n, f);
7               }
8         }
```

   Partant d'un tableau p initialisé par $p[i] := i$ pour tout $i = 1..n$, ENUMERE_PERMS $(p,1,n,f)$ *appelle f sur chaque permutation de $\{1,\ldots,n\}$ à son tour.*

7. En déduire un algorithme en temps exponentiel COLORIAGE qui prend en entrée un graphe G, et retourne son nombre chromatique k et un k-coloriage de G.

   Par la question 5, pour trouver le nombre chromatique de $G\hat{=}(V,E)$, il suffit de savoir énumérer les permutations π de $\{1,\ldots,|V|\}$, et pour chaque, de permuter G par π, et d'appeler COL *sur le graphe permuté. On retient à chaque itération le meilleur résultat de* COL. *Chaque itération de la boucle prend un temps polynomial, et il y a $|V|!$ permutations à tester. Or $|V|!\leq|V|^{|V|}=e^{|V|\log|V|}$. Donc l'algorithme tourne en temps $P(|V|+|E|)e^{|V|\log|V|}$, où P est un polynôme, c'est-à-dire en temps exponentiel. On rappelle que le temps exponentiel en n est par définition le temps $O(e^{n^k})$ pour une constante $k\geq 1$ fixée.*

8. Montrer que le nombre chromatique d'un graphe G quelconque est $\leq d(G)+1$. On exhibera un algorithme COLDEG de k-coloriage en temps polynomial de tout graphe tel que $d(G)+1\leq k$. (Indication : combien de couleurs sont utilisées au maximum par les sommets adjacents à un sommet fixé v ?)

L'idée est de parcourir tous les sommets de G, et de les colorier au fur et à mesure d'une couleur différente de tous les sommets adjacents déjà coloriés. Ceci est toujours possible car chaque sommet a $\leq k-1$ sommets adjacents, qui ne peuvent donc consommer que $k-1$ couleurs au plus; il en reste donc toujours une de libre pour colorer le sommet lui-même. On écrit donc:

```
1  fonction COLDEG (V, E) {
2          pour i = 1..|V| { c[i] := 0; }
3          pour i = 1..|V| {
4                  si c[i] = 0
5                     alors {
6                             j := 1
7                             tant que j ∈ V[i] { j := j + 1; }
8                             c[i] := j;
9                     }
10                 }
11         }
```

9. Montrer que le fait qu'un graphe G soit k-colorable ou non reste inchangé si on supprime de G un sommet v quelconque de degré $\leq k-1$, autrement dit G est k-colorable si et seulement si $G-\{v\}$ est k-colorable. On montrera d'autre part comment obtenir un k-coloriage de G à partir d'un k-coloriage quelconque de $G-\{v\}$.

   $G-\{v\}$ est k-colorable si et seulement si G l'est, pour la même raison qu'à la question précédente : si G est k-colorable, $G-\{v\}$ l'est trivialement. Réciproquement, si $G-\{v\}$ est k-colorable, il y aura toujours au moins une couleur de libre parmi k pour colorier les sommets ayant degré au plus $k-1$, qui ne peuvent utiliser qu'au plus $k-1$ couleurs.

10. En déduire qu'il existe un algorithme en temps polynomial qui prend en entrée un graphe G, et retourne un sous-graphe G' dont tous les sommets sont de degré $\geq k$, qui est k-colorable si et seulement si G l'est, et tel que l'on peut reconstruire en temps polynomial un k-coloriage de G à partir de tout k-coloriage de G'.

    Par récurrence sur le nombre de sommets n de G. Soit e le nombre d'arêtes. L'algorithme cherché fonctionnera en $\leq k'n^2e$ opérations élémentaires, où k' est une constante. L'algorithme de reconstruction fonctionnera lui en $\leq k''ne^2$ opérations, où k'' est une constante.

    Détecter si G contient un sommet de degré $\leq k-1$ se fait clairement en $\leq k_1ne$ opérations élémentaires, par un parcours du graphe et de ses listes d'adjacence. De plus, s'il en existe un, v, on peut le retourner, ainsi que $G-\{v\}$, en le même nombre d'opérations. Par hypothèse de récurrence, on peut construire en temps $\leq k'(n-1)^2e$ un sous-graphe G' de $G-\{v\}$ (donc de G), dont tous les sommets de degré $\geq k$, qui est k-colorable si et seulement si $G-\{v\}$ l'est, et tel que l'on peut reconstruire en temps $\leq k''(n-1)e^2$ un k-coloriage de $G-\{v\}$ à partir de tout coloriage de G'.

    Le temps mis pour fabriquer G' est $\leq k_1ne + k'(n-1)^2e$, donc $\leq k'(ne+(n-1)^2e)$ si on choisit $k' \geq k_1$, donc $\leq k'n^2e$.

    Finalement, on peut construire une couleur non prise parmi les $\leq k-1$ couleurs des sommets adjacents à v en au plus k_2e^2 opérations. Choisissant $k'' \geq k_2$, on peut donc reconstruire un k-coloriage de G à partir d'un k-coloriage de G' en temps au plus $k''(n-1)e^2 + k_2e^2 \leq k''ne^2$.

2 Complexité du coloriage

On rappelle que le problème 3-SAT de la satisfiabilité d'ensembles de 3-clauses est NP-complet. Un *littéral* L est soit une variable propositionnelle x soit une négation $\overline{x}$ d'une variable propositionnelle x. Une 3-*clause* est une disjonction de trois littéraux, non nécessairement distincts. Un *ensemble de 3-clauses* S est vu comme une conjonction de ses 3-clauses. Une *valuation* ρ est une application qui à chaque variable x

associe un booléen, VRAI ou FAUX; ρ *satisfait* une 3-clause si elle contient x tel que $\rho(x) =$VRAI ou $\overline{x}$ tel que $\rho(x) =$FAUX; ρ *satisfait* un ensemble S si ρ satisfait toute 3-clause de S. S est *satisfiable* si et seulement si S est satisfait par au moins une valuation ρ.

1. Soit G un graphe 2-colorable. Soit n la couleur du sommet v. Quelles sont les couleurs des sommets adjacents à v ?

   Les couleurs des sommets adjacents à v sont différentes de n, mais il n'y a que deux couleurs disponibles en tout, donc ils ont tous la même couleur, à savoir $3 - n$, ou encore 2 si $n = 1$ et 1 si $n = 2$.

2. Montrer que G est k-colorable si et seulement si toutes ses composantes connexes sont k-colorables. On rappelle qu'une composante connexe de G est un sous-graphe maximal de G dans lequel il existe un chemin entre v et v' pour tous sommets v et v'.

   Si G est k-colorable, alors il est clair que toute composante connexe de G l'est aussi.
   Réciproquement, supposons que toutes les composantes connexes G_1, ..., G_n de G sont k-colorables, et soient c_1, ..., c_n respectivement des k-coloriages de ces composantes connexes. Les domaines de définition de c_1, ..., c_n sont disjoints, considérons donc l'application $c \hat{=} c_1 \uplus \ldots \uplus c_n$. Il s'agit d'un k-coloriage. En effet, si v et v' sont deux sommets adjacents, ils sont dans la même composante connexe par définition, disons G_i, et $c[v] = c_i[v] \neq c_i[v'] = c[v']$.

3. Déduire des questions précédentes que le problème de la 2-colorabilité d'un graphe G donné en entrée est décidable en temps polynomial. On donnera un algorithme 2COL qui prend en entrée un graphe G et qui retourne en sortie soit un 2-coloriage de G dans un tableau c, soit la réponse NON-2-COLORABLE si G n'est pas 2-colorable.

   Supposons G 2-colorable. Choisissons un sommet quelconque de G. On peut supposer sans perte de généralité (échanger les couleurs 1 et 2 au besoin), qu'il est colorié 1. Alors tous les sommets adjacents sont coloriés 2, les sommets adjacents à ces derniers sont coloriés 1, et ainsi de suite. En somme, si G est connexe, G n'a qu'un 2-coloriage, à échange de couleurs près. Si G n'est pas connexe, notons que G est 2-coloriable si et seulement si toutes ses composantes connexes le sont. Ceci mène à l'algorithme :

```
1  fonction 2COL (V, E) {
2          pour i = 1..|V| { c[i] := 0; } (* initialisation. *)
3          tant que VRAI { (* on explore les composantes connexes une par une. *)
4                  i := 1; tant que i ≤ |V| et  c[i] ≠ 0 { i := i + 1; }
5                  si i > |V| (* si pas de composante connexe restante, *)
6                     alors retourner c; (* alors on a trouvé. *)
7                  (* sinon, on passe à la composante connexe suivante. *)
8                  si 2COL_CONNEXE (V, E, c, i, 1) =FAUX
9                     alors retourner NON-2-COLORIABLE;
10                 }
11         }
12
13 fonction 2COL_CONNEXE (V, E, c, i, couleur) (* colorie i par couleur, et récurse. *)
14         si c[i] = 0 (* si i n'est pas encore colorié, on le colorie, et on récurse. *)
15            alors { c[i] := couleur;
16                    pour tout j ∈ V[i] {
17                            si 2COL_CONNEXE (V, E, c, j, 3 − couleur) =FAUX
18                               alors retourner FAUX;
19                            }
20                    }
21         sinon retourner c[i] = couleur;
22         }
```

4. Montrer que le graphe [2] a la propriété suivante : dans tout 3-coloriage de ce graphe, si v_1 et v_3 sont de couleurs autres que 3, et si v_5 est de couleur 1, alors v_1 ou v_3 est de couleur 1. (Le graphe [2], intuitivement, code donc une disjonction.)

   Pour montrer que v_1 ou v_3 est de couleur 1, sachant qu'ils ne sont pas de couleur 3, il suffit de montrer que v_1 et v_3 ne peuvent pas être tous les deux de couleur 2. En effet, si cela était le cas, alors v_2 et v_4 seraient de couleurs différentes de 2, et différentes entre elles, c'est-à-dire de couleurs 1 et 3, ou 3 et 1. Comme v_5 est de couleur différente de ces deux dernières, il serait de couleur 2, ce qui contredit le fait qu'il est de couleur 1.

5. Trouver un graphe ayant au moins deux sommets v_1 et v_2, tel que dans tout 3-coloriage, si v_1 et v_2 sont de couleurs différentes de 3, alors v_2 est de couleur 1 si et seulement si v_1 n'est pas de couleur 1. (Ceci code une négation.)

   Le graphe ayant juste v_1 et v_2 comme sommets et exactement une arête entre les deux convient clairement, puisqu'ils ne peuvent avoir que les couleurs 1 ou 2 par hypothèse.

6. Si un graphe a deux sommets A de couleur 3 et B de couleur 2, dans un 3-coloriage donné, quelles sont les couleurs des sommets adjacents à la fois à A et à B ? (Si on code les formules logiques par des sommets, et que "être de couleur 1" signifie "être vrai", ceci code une conjonction.)

   Trivial : il ne reste qu'une couleur possible, 1.

7. Déduire des trois questions précédentes qu'il existe un algorithme en temps polynomial qui transforme tout ensemble S de 3-clauses en un graphe G qui est 3-colorable si et seulement si S est satisfiable. (Indication : on créera deux sommets distingués A et B dans G, adjacents; par symétrie, on supposera que A est de couleur 3 et B de couleur 2, et on connectera les sous-graphes des questions précédentes à l'un ou l'autre de ces sommets ou même aux deux.)

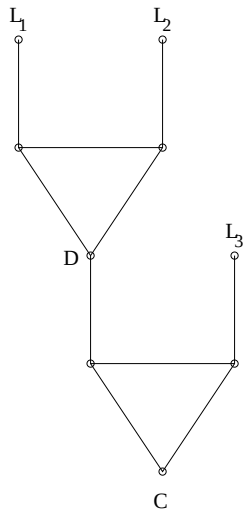
   Intuitivement, il faut pouvoir coder la disjonction et la négation, et le fait d'être VRAI. *Utilisons pas convention la couleur 1 pour représenter* VRAI, *et 2 pour représenter* FAUX. *D'après la question 4, le graphe [2] code la disjonction, à ceci près qu'il faut pouvoir garantir que les entrées v_1 et v_3 sont de couleurs différentes de 3. Pour cela, on crée un sommet spécial A dans le graphe G à venir. Par symétrie, on pourra toujours supposer qu'il sera de couleur 3. Pour assurer que v_1 et v_3 sont de couleurs différentes de 3, il suffira donc de poser des arêtes reliant A à v_1 et v_3.*

   Pour coder la négation, de même, il suffit d'exprimer que la négation d'un sommet est un sommet qui lui est relié par une arête, et qui est aussi relié à A, par la question 5.

   Pour coder le fait qu'une série de sommets, chacun représentant une clause, doivent tous être coloriés 1, on ne peut pas se contenter de les relier à A, ce qui imposerait seulement qu'ils seraient de couleurs 1 ou 2. Par contre, on peut créer un nouveau sommet B, relié à A, et qui par symétrie pourra être supposé de couleur 2. Si on relie tous les sommets représentant les clauses à la fois à A et à B, ceci imposera que tous ces sommets seront coloriés 1, par la question 6.

   Définissons formellement cette construction. Pour tout ensemble S de 3-clauses, on construit un graphe G comme suit :

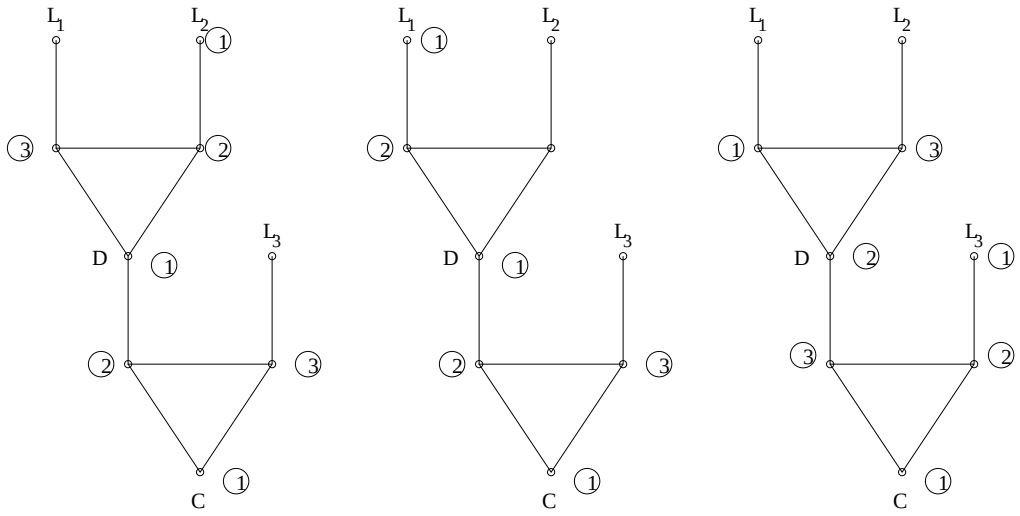
   a. *D'abord, G contient deux sommets distingués A et B, et une arête entre A et B.*
   b. *Ensuite, pour toute variable x dans S, on crée deux sommets x et $\overline{x}$, une arête entre x et $\overline{x}$, et deux arêtes reliant A à x et à $\overline{x}$. (On a ainsi naturellement, pour chaque littéral L, un sommet correspondant à L.)*
   c. *Pour chaque clause $C \hat{=} L_1 \vee L_2 \vee L_3$ dans S, on crée un sommet dénoté C, plus 5 autres sommets en combinant deux graphes de la forme [2] :*



[4]

d. On relie finalement tous les sommets C ainsi obtenus à l'étape précédente à la fois à A et à B, et les sommets D à A.

Montrons que si S est satisfiable, alors G est 3-coloriable. Soit donc ρ une valuation satisfaisant S, et assimilons 1 à VRAI, *2 à* FAUX. *Colorions chaque sommet x par $\rho(x)$, et $\overline{x}$ par la couleur correspondant à la négation de $\rho(x)$. On colorie A par 3, B par 2, ce qui colorie correctement les arêtes des points a et b de la construction. Noter que pour toute clause $C \hat{=} L_1 \vee L_2 \vee L_3$ de S, au moins un des littéraux L_1, L_2 ou L_3, vu en tant que sommet, a reçu la couleur 1. Colorions chaque graphe [4] par le graphe de gauche si L_1 est colorié 1, par celui du milieu si L_2 est colorié 1, par celui de droite si L_1 n'est pas colorié 1 mais L_3 l'est :*



Il s'ensuit que les arêtes du point c de la construction sont bien coloriées. Comme tous les sommets C sont coloriés 1, et tous les sommets D sont coloriés 1 ou 2, les arêtes du point d sont aussi bien coloriées.

Réciproquement, supposons G 3-colorié. Sans perte de généralité, on peut supposer que A est colorié 3, et B, qui doit avoir une couleur différente de A par le point a de la construction, est colorié 2. Par le point b, les sommets x et $\overline{x}$ sont de couleurs 1 ou 2, et de couleurs différentes. Notons ρ la valuation qui à x associe VRAI *si x est de couleur 1,* FAUX *si x est de couleur 2. Mais, pour toute clause $C \hat{=} L_1 \vee L_2 \vee L_3$, par le point d, C est de couleur 1. Ceci implique que D ou L_3 est colorié 1; en effet la question 4 s'applique parce que D et L_3 sont de couleurs différentes de 3 : dans le cas de D, c'est parce que D est relié à A par le point c, et pour L_3 c'est parce que c'est un x ou un $\overline{x}$, qui est relié à A par le point b. De plus, si D est colorié 1, alors par le point b et*

la question 4 encore, L_1 ou L_2 est colorié 1. Donc au moins un des L_i, $1 \leq i \leq 3$, est colorié 1. Mais par définition de ρ, L_i est colorié 1 si et seulement si L_i est VRAI *dans ρ. Donc ρ satisfait toutes les clauses C de S, donc S est satisfiable.*

Par la question 1.5, le problème de 3-colorabilitié est dans NP*. La construction ci-dessus montre que* 3-SAT *se transforme en temps polynomial en une instance de 3-colorabilité. Donc la 3-colorabilité est* NP*-complète.*

8. Que peut-on conclure de la question précédente ?

   On en conclut bien sûr que le problème de 3-colorabilité d'un graphe est NP*-complet.*

3 Mots de passe à connaissance nulle

On construit un système de mots de passe permettant à un utilisateur U de s'authentifier sans jamais divulguer son mot de passe, même à l'ordinateur O sur lequel il se connecte. Soit t_0 une durée quelconque dépassant le durée prévue d'utilisation du système (par exemple, 1000 ans).

On suppose pour ceci que l'on dispose d'un générateur de bits aléatoires, d'un mécanisme de chiffrement f, tel que pour tout suite M d'au moins N_0 bits (le *message en clair*), pour toute suite K d'au moins N_1 bits (la *clé*), $f(M,K)$ est une suite de $\geq N_0$ bits. On suppose que dans ces conditions, il existe une unique suite K' de bits (la *clé inverse*), contenant au moins N_1 bits, telle que $f(f(M,K),K') = M$. La fonction $M \mapsto f(M,K)$ est donc injective. On suppose aussi que l'image inverse M de $M' = f(M,K)$ n'est pas calculable en temps inférieur à t_0 en fonction de M' seul (en particulier, sans connaissance de K'). On supposera finalement que l'on sait produire des paires de clés (K,K') aléatoires, avec K inverse de K'.

On supposera d'autre part qu'il est impossible de trouver un 3-coloriage d'un graphe ayant au moins N_2 arêtes, où N_2 est une constante suffisamment grande, en un temps inférieur à t_0. (Ceci est une simplification, en réalité ce n'est vrai qu'avec probabilité proche de 1.)

Soit $\overline{i}$ la suite de deux bits codant l'entier $i \in \{0,1,2,3\}$ en binaire; réciproquement, $\underline{r}$, où r est une suite de deux bits, est l'entier représenté par r en binaire. On note ϵ la suite vide, et $s \cdot s'$ la concaténation des suites s et s'.

L'utilisateur U s'authentifie en utilisant comme nom d'utilisateur un graphe G aléatoire 3-colorable ayant au moins αN_2 arêtes, où α est une constante > 1, et comme mot de passe un 3-coloriage c de G. Quant à O, il maintient un ensemble fini $\mathcal{G}$ de noms d'utilisateurs autorisés, tous de taille $\geq N_2$. Soit n_0 un entier fixé. Une session d'authentification se déroule comme suit :

a. U fournit à O son nom G (dont les sommets sont $1, \ldots, N$).
b. O pose $n := 0$. Si G n'est pas dans $\mathcal{G}$, alors O répond NON et la session se termine.
c. Si $n > n_0$, alors O répond OUI et la session se termine. Sinon :
d. U tire une permutation aléatoire π des couleurs $\{1,2,3\}$, et envoie à O les suites de bits $f(\overline{\pi(c(i))} \cdot R_i, K_i)$, pour chaque sommet $i \in \{1,\ldots,N\}$, où R_i est une chaîne de bits aléatoires de longueur $\geq N_0 - 2$, et (K_i, K_i') est une paire de clés aléatoires de tailles $\geq N_1$, K_i' étant l'inverse de K_i.
e. Maintenant, O possède une description de G, et pour tout sommet i de G, une suite de bits M_i. O tire aléatoirement deux sommets i et j, $1 \leq i,j \leq N$, $i \neq j$, qui sont adjacents dans G, et fournit i et j à U.
f. U fournit les clés inverses K_i' et K_j' à O.
g. O calcule $\underline{r_i}$ et $\underline{r_j}$, où r_i est la suite formée des deux premiers bits de $f(M_i, K_i')$ et r_j de même pour $f(M_j, K_j')$. Si $\underline{r_i} \neq \underline{r_j}$ et $\underline{r_i} \neq 0$, $\underline{r_j} \neq 0$, alors O incrémente n et retourne à l'étape c. Sinon, O répond NON et la session se termine.

1. Montrer que si U possède effectivement un 3-coloriage c de G et l'utilise effectivement comme il est spécifié, alors O retourne OUI, mais que si U est un *intrus* et fournit un graphe G en-dehors de $\mathcal{G}$ ou n'en connaît pas de 3-coloriage, alors O retourne NON avec une probabilité tendant rapidement vers 1 lorsque n_0 tend vers $+\infty$, que l'on déterminera. (On pourra supposer que n_0 est négligeable devant la taille de G.)

Il est clair que O doit retourner OUI *si U obéit à la spécification, car le protocole termine après n_0 tours de boucle, et d'autre part les couleurs $\underline{r}_i$ et $\underline{r}_j$ de l'étape g sont nécessairement les couleurs $c(i)$ et $c(j)$ des deux sommets adjacents (à cause du choix de i et j dans l'étape e).*

Supposons maintenant que U est un intrus. S'il ne fournit pas un $G \in \mathcal{G}$ à l'étape a, alors O le détecte à l'étape b, de façon sûre. Sinon, l'intrus par définition n'en connaît pas de coloriage. Examinons la probabilité que O ne retourne pas NON *au cours d'un tour de la boucle c–g. Soit c'_ℓ, $1 \leq \ell \leq N$, la "couleur" fournie par U à O à l'étape d, c'est-à-dire $\underline{r}_\ell$, où r_ℓ est la suite des deux premiers bits de $f(M_\ell, K'_\ell)$. (Intuitivement, le schéma de l'étape d force U à fournir quelque chose qui ressemble à un coloriage, sans le révéler à O.) Comme U ne peut pas calculer de 3-coloriage des graphes avec N_2 arêtes, et que G a βN_2 arêtes avec $\beta \geq \alpha$, il y a au moins $(\beta - 1)N_2$ arêtes $\{\ell_1, \ell_2\}$ dont les extrémités ont mêmes couleurs ($c'_{\ell_1} = c'_{\ell_2}$) ou dont l'une des extrémités a couleur 0 ($c'_{\ell_1} = 0$ ou $c'_{\ell_2} = 0$). Donc la probabilité que le test de l'étape g réponde* NON *est au moins $1 - 1/\beta \geq 1 - 1/\alpha$. Comme les valeurs de i et de j tirées à l'étape e sont indépendantes, la probabilité que O réponde* NON *au cours d'une session est au moins $1 - 1/\alpha^{n_0}$.*

2. Montrer que le système ne permet à O d'obtenir aucune information sur le 3-coloriage c de G que U possède. Plus précisément, supposons que O est un *intrus* qui implémente un algorithme arbitraire plutôt que celui décrit plus haut, et retournant aussi soit OUI soit NON après un nombre fini de tours n_0. Considérons deux utilisateurs U et U', possédant le même nom G, et deux 3-coloriages différents c et c'.

   Montrer que O ne peut pas faire la différence entre U et U', au sens où la distribution des réponses de O en présence de l'un ou l'autre est la même, avec probabilité très proche de 1; on déterminera quels paramètres doivent tendre vers l'infini pour que cette probabilité tende vers 1.

   Comme la permutation π tirée par U à l'étape d, pour toute arête $\{i, j\}$ de G la distribution des couples $(\pi(c(i)), \pi(c(j)))$ est identique à la distribution uniforme sur les couples d'entiers distincts entre 1 et 3. Noter qu'à cause du fait que U ne fournit à l'étape f que deux clés K'_i et K'_j, l'intrus O ne peut récupérer au mieux que les valeurs $\pi(c(i))$ et $\pi(c(j))$ avec forte probabilité lors d'un tour de boucle. Cette probabilité p est celle que K'_i et K'_j soient différentes de tous les autres K'_ℓ. Donc $1 - p$ est la probabilité que K'_i est égale à une autre clé, ou que K'_j est égale à une autre clé, est est donc inférieure ou égal à $2(1 - p_0)$, où p_0 est la probabilité qu'une clé soit différente de $N - 2$ clés aléatoires, soit $p_0 \geq (1 - 1/2^{N_1})^{N-2}$. Donc $p \geq p_1 \hat{=} 2(1 - 1/2^{N_1})^{N-2} - 1$, qui est très proche de 1 dès que N_1 est assez grand.

   Il s'ensuit qu'avec probabilité au moins $p_1^{n_0}$, la distribution des (M_i, M_j) récupérés par O à l'étape e est la même que O dialogue avec U ou U', donc de même pour la réponse finale de O.

   Asymptotiquement, faire tendre n_0 vers $+\infty$ seul est ici désastreux, de même pour N. C'est N_1, la taille des clés, qui doit croître vers l'infini. On peut cependant faire croître toutes les quantités, N_1, N et n_0 vers l'infini à condition que $Nn_0 = o(2^{N_1})$. En effet, quand tous les paramètres tendent vers l'infini, $p_1 \sim 2(1 - N/2^{N_1}) - 1 = 1 - 2N/2^{N_1}$, donc $p_1^{n_0} \sim 1 - 2Nn_0/2^{N_1}$, qui tend vers 1.