Complétude

Exercice 1 [Relation d'ordre]. Etant donné deux langages $\mathcal{L} \subset \mathcal{A}^*$ et $\mathcal{L}' \subset \mathcal{B}^*$, on dit que $\mathcal{L}$ *se réduit à* $\mathcal{L}'$, noté $\mathcal{L} \leq_{\mathbf{P}} \mathcal{L}'$ si il existe un algorithme fonctionnant en temps polynomial $f : \mathcal{A}^* \to \mathcal{B}^*$ vérifiant :

$$x \in \mathcal{L} \iff f(x) \in \mathcal{L}'$$

1. Montrer que $\leq_{\mathbf{P}}$ est un pré-ordre, c'est à dire qu'elle est réflexive et transitive.
2. Une classe de complexité $\mathcal{C}$ est close par $\leq_{\mathbf{P}}$ si $\mathcal{L} \leq_{\mathbf{P}} \mathcal{L}'$ et $\mathcal{L}' \in \mathcal{C}$ implique que $\mathcal{L} \in \mathcal{C}$. Montrer que **P**, **NP** et **co-NP** sont closes pour $\leq_{\mathbf{P}}$.
3. Montrer que **Exp** et **NExp** sont closes pour $\leq_{\mathbf{P}}$.
4. On suppose que $\mathbf{P} = \mathbf{NP}$. Soit $\mathcal{L} \in \mathbf{P}$ tel que $\mathcal{L} \neq \emptyset$ et $\mathcal{L} \neq \mathcal{A}^*$, montrer que pour tout $\mathcal{L}' \in \mathbf{NP}$ on a $\mathcal{L}' \leq_{\mathbf{P}} \mathcal{L}$ (autrement dit $\mathcal{L}$ est **NP**-complet). Pourquoi doit-on exclure les langages triviaux ?
5. Soit $\mathcal{L}$ un problème **NP**-complet. Montrer que si $\mathcal{L}' \in \mathbf{NP}$ et $\mathcal{L} \leq_{\mathbf{P}} \mathcal{L}'$ alors $\mathcal{L}'$ est **NP**-complet.

Exercice 2 [Exemples de problèmes complets]. On s'intéresse aux problème de complétude suivant :

1. Montrer que le langage $\{(\langle \mathcal{N} \rangle, x, 1^t)$: la machine non déterministe $\mathcal{N}$ accepte x en temps $\leq t\}$ est **NP**-complet.
2. Montrer que le langage $\{(\langle \mathcal{M} \rangle, x, t)$: la machine déterministe $\mathcal{M}$ accepte x en temps $\leq t\}$ est **Exp**-complet (t est coder en binaire).
3. Montrer que le langage $\{(\langle \mathcal{N} \rangle, x, t)$: la machine non déterministe $\mathcal{N}$ accepte x en temps $\leq t\}$ est **NExp**-complet (t est coder en binaire).

Exercice 3 [`SAT` est NP-complet]. Un *littéral* est soit une variable x_i soit sa négation $\neg x_i$. Une clause est une disjonction de littéraux $C = l_1 \vee \cdots \vee l_k$ où les l_i sont des littéraux. Une formule est dites sous *forme normale conjonctive* si elle est la conjonction de clause :

$$\varphi = \bigwedge_i C_i \quad \text{où} \quad C = \bigvee_j l_j \quad \text{est une clause}$$

Le problème suivant est connu comme **NP**-complets.

`SAT` :

— *entrée :* Une formule booléenne φ écrite comme une conjonction de disjonction de littéraux.
— *question :* φ est-elle une satifaisable, c'est à dire existe t'il une affectation $(a_1, \ldots, a_n)$ telle que $\varphi(a_1, \ldots, a_n)$ soit vraie ?

`SAT` prend en entrée une formule booléenne sans quantificateur $\varphi(x_1, \ldots, x_n)$ est on cherche à savoir s'il existe une affectation des variable qui satisfait φ. Le problème `SAT` est connu comme **NP**-complets. A l'aide de la technique de réduction montrer que le problème suivant est **NP**-complet.

`3SAT` :

— *entrée :* Une formule φ formée de conjonction de disjonction de littéraux où chaque disjonction contient au plus 3 littéraux.
— *question :* φ est elle satisfaisable ?

Exercice 4. Le problème `Clique` s'énonce ainsi :

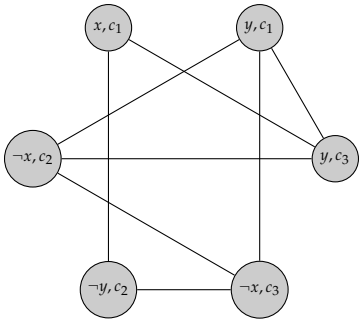
— *entrée :* Un graphe non orienté G et un entier k.

— *question :* Existe-t-il un ensemble de k sommets deux à deux adjacents ?

Nous allons monter que ce problème est **NP**-complet, en le réduisant depuis `SAT`. Pour cela nous allons suivre une preuve proposée par Richard Karp en 1972.

1. Montrer que `Clique` $\in$ **NP**.
2. Soit F une formule en forme normale conjonctive. On définit le graphe $G_F = (V_F, E_F)$ tel que :
   — les sommets de ce graphe sont les paires (l, c), où c est une clause de F et l est un littéral (c'est à dire une variable ou sa négation) qui apparaît dans c ;
   — les sommets (l, c) et (l', c') sont reliés dans le graphe si et seulement si $c \neq c'$ et l n'est pas la négation de l'.

   Par exemple si $F = c_1 \wedge c_2 \wedge c_3$ avec $c_1 = (x \vee y)$, $c_2 = (\neg x \vee \neg y)$ et $c_3 = (\neg x \vee \neg y)$, G_F est le graphe suivant :

   Montrez que si F est constituée de n clauses, alors F est satisfiable si et seulement si G_F contient une clique de cardinal n. On prouvera soigneusement les deux implications en construisant une clique dans un sens, et une interprétation des formules dans un autre.
3. On définit la taille $|F|$ d'une formule F par le nombre de littéraux qui la constituent (dans l'exemple ci-dessus, $|F| = 6$), et la taille $|G|$ d'un graphe comme la somme de son cardinal (nombre de sommets) et de son nombre d'arêtes. Bornez $|G_F|$ par un polynôme en $|F|$ (c'est-à-dire trouvez un polynôme p tel que $|G_F| \leq p(|F|)$).
4. En utilisant les résultats précédents, prouvez que `Clique` est **NP**-complet.
5. On considère le problème `Ensemble Indépendant` :
   — *entrée :* Un graphe non orienté G et un entier k.
   — *question :* Existe-t-il un ensemble de k sommets indépendants, c'est à dire tous non reliés deux à deux ?

   Montrer que le problème `Ensemble Indépendant` est **NP**-complet.

Exercice 5 [Coloriages]. Une k-coloration d'un graphe non orienté $G = (S, A)$ est une fonction $c : S \longrightarrow \{1, \ldots, k\}$ telle que $c(u) \neq c(v)$ si $(u, v) \in A$. Autrement dit $1, 2, \ldots, k$ représentent les k couleurs et deux sommets adjacents doivent avoir des couleurs différentes.
On considère les deux problèmes de décision suivant.

3-**Coloriage :**

- *entrée :* Un graphe non orienté G.
- *question :* Est-il possible de colorier les sommets de G avec 3 couleurs différentes de telle manière que deux sommets reliés par une arête aient des couleurs différentes ?

Coloriage :

- *entrée :* Un graphe non orienté G et un entier k codé en binaire.
- *question :* Est-il possible de colorier les sommets de G avec k couleurs différentes de telle manière que deux sommets reliés par une arête aient des couleurs différentes ?

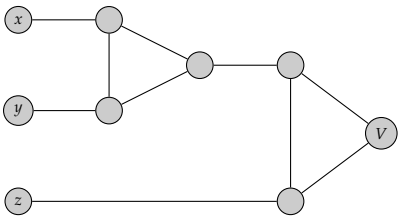
Partie I : **Quelques questions simples.**

1. Montrer que 3−`Coloriage` $\in$ **NP** et `Coloriage` $\in$ **NP**.
2. On suppose que 3−`Coloriage` est **NP**-complet. Est ce qu'on peut en déduire que `Coloriage` est **NP**-complet ? Justifiez votre réponse.

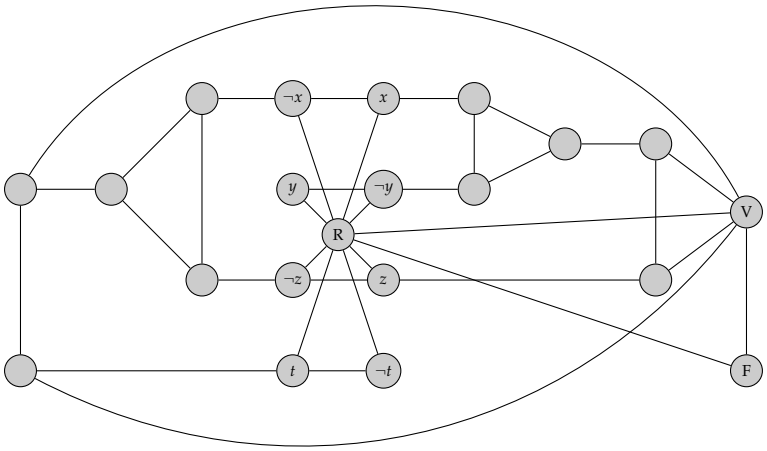
Partie II : **Montrons que 3−`Coloriage` $\in$ NP-complet.**

Pour démontrer que 3−`Coloriage` $\in$ **NP**-complet on utilise une réduction à 3−`SAT`. On admettra que 3−`SAT` est **NP**-complet. Etant donnée une formule φ sur n variables formée par une conjonction de m clauses tel que chaque clause est une disjonction de trois variables ou négation de variables, on associe un graphe $G_\varphi = (S_\varphi, A_\varphi)$.
L'ensemble des sommets S_φ est composé d'un sommet pour chaque variable, d'un sommet pour chaque négation de variable, 5 sommets pour chaque clause et de trois sommet spéciaux noté V, F et R (pour vrai, faux et rouge).
L'ensemble des arêtes A_φ est construit de la manière suivante :

— pour chaque variable on relie deux à deux le sommet qui lui correspond, le sommet qui correspond à sa négation et le sommet R pour former un triangle ;
— on relie deux à deux les trois sommets spéciaux pour former un triangle ;
— pour chaque clauses, on associe 10 arêtes qui relient les cinq sommets associés à cette clause, les trois variables ou négation de variables associé à cette clause et le sommet spécial V de telle sorte à avoir le sous graphe suivant (on suppose ici que l'on considère la clause $x \vee y \vee z$) :

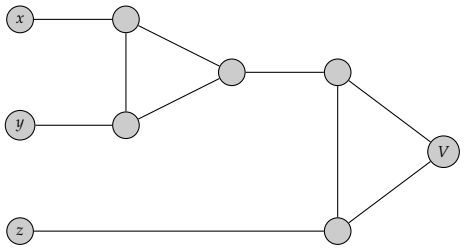


Par exemple la formule $\psi = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg z \vee t)$ va être coder par le graphe G_ψ suivant :



1. Donner un 3-coloriage de G_ψ (on pourra colorier directement le graphe donné) et en déduire une affectation des variables de ψ qui montre qu'elle est satisfaisable.
2. Nous allons faire le cas général. Soit φ une formule et c un coloriage associé. Montrez que chaque sommet associé à sa variable ou à sa négation est colorié par $c(V)$ ou $c(F)$, les couleurs des sommets spéciaux V et F.
3. Dans le graphe suivant, montrer que si on colore le sommet V avec la couleur 1 et les sommets x, y et z par les couleurs 1 ou bien 2 alors il est possible de compléter le coloriage

pour obtenir un 3-coloriage si et seulement si au moins un des sommets x, y ou z est colorié avec la couleur 1.



4. Montrer que pour toute interprétation de φ il existe une 3-coloration de G_φ.
5. Montrer que s'il existe une 3-coloration de G_φ alors φ admet une interprétation valide.
6. Montrer qu'il existe un algorithme polynomial qui transforme une formule φ en graphe G_φ. En particulier on donnera le nombre de sommets et d'arêtes de G_φ.
7. Conclure que 3-`Coloriage` $\in$ **NP**-complet.

Partie III Un établissement contient 3 salles. On cherche à construire un algorithme qui prend en argument les plages horaires où l'on a besoin d'une salle et répond oui si il existe une affectation possible et non sinon (c'est à dire que l'on a besoin de plus de 4 salles pour satisfaire les contraintes).
Votre directeur vous demande de réaliser un algorithme qui résout ce problème en temps polynomial. Qu'est ce que vous lui répondez ?

Exercice 6. Montrer que si un problème $\mathcal{L}$ est **NP**-complet alors $\overline{\mathcal{L}}$ est **co-NP**-complet.
En déduire que le problème suivant est **co-NP**-complet.

`TAUTOLOGIE` :

— *entrée :* Une formule booléenne φ.
— *question :* φ est-elle une tautologie, c'est à dire $\varphi(a_1, \ldots, a_n)$ est vraie pour toute affectation $(a_1, \ldots, a_n)$?

Exercice 7 [Algorithme polynomial pour résoudre `SAT` si P = NP]. Le but de cet exercice est de donner un algorithme polynomial pour résoudre `SAT` si **P** = **NP**. Soit $(\mathcal{M}_i)_{i\in\mathbb{N}}$ une énumération effective des machines de Turing fonctionnant en temps polynomial. On considère l'algorithme suivant :

Algorithm 1: Algo polynomial pour les grandes instances de `SAT` si **P** = **NP**

Data: Une formule φ de taille n
Result: `accepter` ou `rejeter`

---

for *i=1 to n* **do**
  simuler $\mathcal{M}_i(\psi)$ pour toute formule ψ de taille $\leq \log(n)$;
  si pour tout ψ, $\mathcal{M}_i$ donne toujours une affectation valide lorsque $\psi \in$ `SAT`, sortir de la boucle;
simuler $\mathcal{M}_i(\varphi)$ produisant une affectation a des variables φ ; si $\varphi(a) = 1$ `accepter` sinon `rejeter` ;

Montrer que si **P** = **NP** alors l'algorithme précédent décide `SAT` en temps polynomial pour des instances suffisamment grandes.