TD 3 : Réductions et complétude

Théorie de la complexité S5.

2017-2018

Exercice 1 — *Réductions simples*

1. Montrer que (SIZE) $\preccurlyeq$ (INS).

► **Correction**

Cette réduction est un peu tordue, il est conseillé de ne pas la lire en premier.

On rappelle la définition formelle des deux problèmes :

(SIZE)Soit G un graphe non orienté et un entier k, G est-il de taille k ?

(INS) Soit G un graphe non orienté et un entier k, G possède-t-il un stable de taille k ? (k nœuds non reliés deux à deux).

On veut montrer que (SIZE) se réduit à (INS), c'est-à-dire qu'il est plus facile qu'(INS). C'est logique, il est clairement plus facile de trouver la taille d'un graphe (il suffit d'énumérer les nœuds) que de trouver un stable de taille k (il faut énumérer tous les ensembles de k nœuds). On va montrer que, en disposant d'une boîte noire pour résoudre (INS), on peut résoudre (SIZE) (de manière un peu tordue).

On va commencer par une première idée qui ne fonctionne pas. Puis une seconde qui marche mais qui courtcircuite complètement l'idée de réduction abordée dans le cours (Encore une fois ne lisez pas cette réduction en premier).

Première idée :

Soit $G = (V, E), k$ une instance de SIZE. Considérons la transformation suivante qui construit une instance (G', k') de (INS). Cette transformation consiste à supprimer toutes les arêtes de G. Le graphe résultat est G'. On pose $k' = k$.

Cette réduction est polynomiale, elle se fait en $O(|E| + \log(k))$.

Dans G', tous les nœuds ne sont pas reliés, donc un stable maximum est de taille égale à celle du graphe. Ainsi, (G, k) est une instance positive de (SIZE), alors (G', k') est une instance positive de (INS). Cependant, si (G, k) est une instance négative alors (G', k') peut être positive ou négative. En effet si la taille du graphe est inférieure à k, alors il n'existe pas de stable de taille k, donc l'instance est négative. Si maintenant la taille du graphe est supérieure à k, alors il existe bien un stable de taille k. En fait il existe un stable de taille 1, de taille 2, ..., jusqu'à la taille du graphe, en passant par la taille k. Donc la seconde instance sera positive. On a donc seulement la moitié des propriétés nécessaires à une réduction.

Deuxième idée :

Appliquons la transformation suivante : Poser $s = 0$, pour tout nœud de V ajouter 1 à s. Si $s = k$ alors renvoyer une instance (G', k') où G' ne contient qu'un seul nœud et $k' = 1$ sinon renvoyer une instance (G', k') où G' ne contient qu'un seul nœud et $k' = 2$.

Clairement on a résolu (SIZE) et on a construit une instance bidon de (INS). Si la réponse de (SIZE) est positive, alors on renvoie une instance positive triviale de (INS) et sinon on renvoie une instance négative triviale de (INS). Et cet algorithme est polynomial ; il tourne en $O(|V|)$. Cette transformation respecte donc bien les définitions de la réduction polynomiale de Karp. Mais on a l'impression de tricher. C'edst parce que (SIZE) est dans P, donc on peut le résoudre en temps polynomial. Comme on cherche une réduction "Polynomiale" de Karp, on peut se permettre, pendant la transformation de la réduction de résoudre le problème de départ. Généralement on travaille avec des problèmes de NP donc cette idée ne marche pas.

On aurait pu ici se demander s'il existe une réduction "logarithmique" de Karp qui travaille en temps logarithmique plutôt que polynomial ; cela aurait été plus intéressant.

2. Montrer que (SIZE) $\preccurlyeq$ (CHROMA).
3. Montrer que (INS) $\preccurlyeq$ (W-INS).

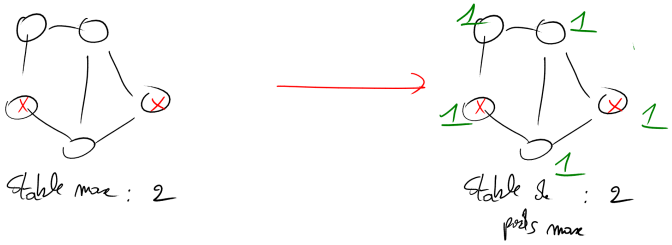
► **Correction**

(INS) Soit G un graphe non orienté et un entier k, G possède-t-il un stable de taille k ?

(WINS) Soit G un graphe non orienté, des poids $\omega : V \to \mathbb{N}$ sur les nœuds et un entier k, G possède-t-il un stable de dont la somme des poids dépasse k ?

Le second problème a l'air d'être une généralisation du premier. L'idée de cette réduction est de montrer qu'il existe des instances de (WINS) pour lesquels les réponses sont les mêmes que les instances d'(INS).

La transformation est la suivante, partant d'une instance $(G = (V, E), k)$ d'INS, on construit une instance (G', ω', k') de (WINS). Pour cela, on pose $G' = G$, $k' = k$ et $\omega(v) = 1$ pour tout nœud v.



Cette réduction se fait en temps linéaire $O(|V| + |E| + \log(k) + |V|)$, donc polynomial en, la taille de (G, k).

L'instance (G, k) est positive
si et seulement s'il existe dans G un stable de taille k
si et seulement s'il existe dans G' un stable de taille k'
si et seulement s'il existe dans G' un stable de poids k' puisque chaque poids est égal à 1.

Donc on a bien conservation de la positivité et la négativité est instances.

Donc il existe une réduction polynomiale de Karp de (INS) vers (WINS)

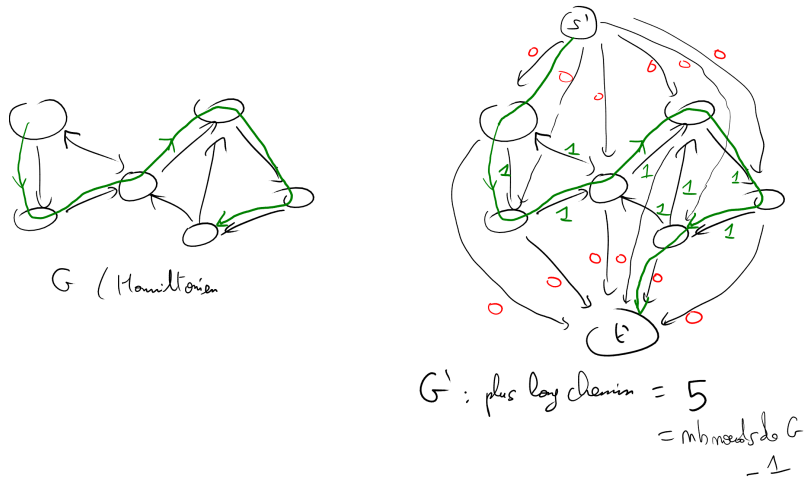
4. Montrer que (3-COL ?) $\preccurlyeq$ (SAT).
5. Montrer que (COHAM) $\preccurlyeq$ (TAU).
6. Montrer que (HAM ?) $\preccurlyeq$ (LOP).

► **Correction**

(HAM ?) Soit G un graphe orienté ; existe-t-il un chemin hamiltonien dans G ? (je prend la version orientée avec chemin c'est plus facile)

(LOP) Soit G un graphe orienté, des poids $\omega : \mathbb{E} \to \mathbb{N}$, un nœud s et un nœud t et un entier k, G possède-t-il un chemin élémentaire (ne passant qu'une fois au plus par chaque nœud) de s à t dont la somme des poids des arcs dépasse k ?

La transformation est la suivante, partant d'une instance $(G = (V, E))$ de (HAM ?), on construit une instance $(G', \omega', s', t', k')$ de (LOP). Pour cela, on pose $k' = |V| - 1$. Pour G', on commence par copier G, on ajoute deux sommets s' et t' et on relie s' à tous les sommets de V et tous les sommets de V à t'. On pose enfin $\omega(a) = 1$ pour tout arc a de E et 0 pour tout les arcs reliant s' et t' à V.



Cette réduction se fait en temps linéaire $O(\log(|V|) + (|V| + |E|) + 2|V| + (|E| + 2|V|))$, donc polynomial en, la taille de G.

Si l'instance (G, k) est positive, il existe dans G un chemin élémentaire $(u_1, u_2, \ldots, u_n)$ passant par tous les nœuds. Dans G', il existe donc le chemin $(s', u_1, u_2, \ldots, u_n, t')$ dans G'. Ce chemin utilise exactement $|V| - 1$ arcs de G et 2 arcs (s', u_1) et (u_n, t'). Ce chemin est de poids $|V| - 1 \geq k'$. L'instance $(G', \omega', s', t', k')$ est donc positive.

Si l'instance $(G', \omega', s', t', k')$ est positive, il existe dans G', un chemin élémentaire reliant s' à t' de poids au moins k'. Montrons que ce chemin possède exactement k' arcs de poids 1 (ça peut sembler évident mais il faut le prouver). Ce chemin s'écrit donc $(s', u_1, u_2, \ldots, u_p, t')$ Puisqu'il est de poids au moins k' et que les arcs sont de poids 1 ou 0, il emprunte au moins k' arcs de poids 1. Puisque le chemin est élémentaire, il ne peut passer par plus de $|G'| = |V| + 2$ nœuds et donc $|V| + 1$ arcs de poids 1. Puisqu'enfin les arcs issus de s' et allant en t' sont de poids 0, le chemin ne peut posséder plus de $|V| - 1 = k'$ arcs de poids 1. Donc ce chemin possède exactement k' arcs de poids 1.

Les arcs de poids 1 relient $u_1, u_2, \ldots, u_p$. Donc $p = k' + 1 = |V|$. Donc le chemin $u_1, u_2, \ldots, u_p$ est hamiltonien dans G.

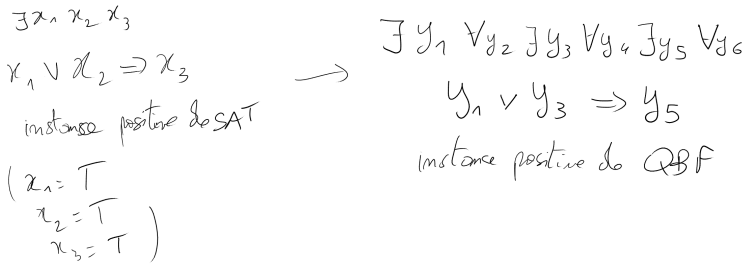
Donc on a bien conservation de la positivité et la négativité est instances.

Donc il existe une réduction polynomiale de Karp de (HAM ?) vers (LOP)

7. Montrer que (HAM ?) $\preccurlyeq$ (TSP).
8. Montrer que (SAT ?) $\preccurlyeq$ (3-SAT ?).
9. Montrer que (TAU ?) $\preccurlyeq$ (3-TAU ?).
10. Montrer que (SAT ?) $\preccurlyeq$ (QBF ?).

► **Correction**

(SAT ?) Soit une formule φ avec n variables $x_1, x_2, \ldots, x_n$, est-ce que $\exists x_1, x_2, \ldots, x_n | \varphi(x_1, x_2, \ldots, x_n)$?
(QBF ?) Soit une formule φ avec n variables $x_1, x_2, \ldots, x_n$ est-ce que $\exists x_1 \forall x_2 \exists \ldots \forall x_n | \varphi(x_1, x_2, \ldots, x_n)$.
La transformation est la suivante, partant d'une instance $(\varphi, x_1, x_2, \ldots, x_n)$ de (SAT ?), on construit une instance $(\varphi', y_1, y_2, \ldots, y_m)$ de (QBF ?). Pour cela, on pose $m = 2n$ avec $\varphi'(y_1, y_2, \ldots, y_m) = \varphi(y_1, y_3, y_5, \ldots, y_{2n-1})$.



Cette transformation est polynomiale puisqu'elle ne prend que le temps de recopier la formule.
Si $\exists x_1, x_2, \ldots, x_n | \varphi(x_1, x_2, \ldots, x_n)$ alors, $\exists y_1, y_3, \ldots, y_{2n-1} | \varphi(y_1, y_3, y_5, \ldots, y_{2n-1})$ (on renomme les variables).
Puisque $y_2, y_4, \ldots, y_{2n}$ n'interviennent pas dans cette formule, on a

$$\begin{aligned}
&\exists y_1, y_3, \ldots, y_{2n-1} | \varphi(y_1, y_3, y_5, \ldots, y_{2n-1}) \\
&= \forall y_2, y_4, y_{2n} \exists y_1, y_3, \ldots, y_{2n-1} | \varphi(y_1, y_3, y_5, \ldots, y_{2n-1}) \\
&= \exists y_1 \forall y_2 \exists \ldots \forall y_{2n} | \varphi(y_1, y_3, y_5, \ldots, y_{2n-1}) \\
&= \exists y_1 \forall y_2 \exists \ldots \forall y_{2n} | \varphi'(y_1, y_2, y_3, \ldots, y_{2n-1})
\end{aligned}$$

Donc $(\varphi', y_1, y_2, \ldots, y_m)$ est une instance positive.
Inversement, si $(\varphi', y_1, y_2, \ldots, y_m)$ est une instance positive, alors, puisque $y_2, y_4, \ldots, y_{2n}$ n'interviennent pas dans φ', on a toujours
$\exists y_1 \forall y_2 \exists \ldots \forall y_{2n} | \varphi'(y_1, y_2, y_3, \ldots, y_{2n-1}) = \exists y_1, y_3, \ldots, y_{2n-1} | \varphi(y_1, y_3, y_5, \ldots, y_{2n-1})$
Et donc $\exists x_1, x_2, \ldots, x_n | \varphi(x_1, x_2, \ldots, x_n)$ donc $(\varphi, x_1, x_2, \ldots, x_n)$ est une instance positive.
Donc on a bien conservation de la positivité et la négativité est instances.
Donc il existe une réduction polynomiale de Karp de (SAT ?) vers (QBF ?)

11. Montrer que (TAU ?) $\preccurlyeq$ (QBF ?).
12. Montrer que (CONNECTIVITY ?) $\preccurlyeq$ (MSPT).
13. Montrer que (SUDOKU ?) $\preccurlyeq$ (CHROMA).
14. Montrer que (INS) $\preccurlyeq$ (ILP).
15. Montrer que (BIPARTI ?) $\preccurlyeq$ (2-COL ?).
16. Montrer que (2-COL ?) $\preccurlyeq$ (BIPARTI ?).
17. Montrer que (SUBSET SUM ?) $\preccurlyeq$ (PARTITION).
18. Montrer que (SUBSET SUM ?) $\preccurlyeq$ (KNAPSACK).

► **Correction**

(SUBSETSUM ?) Soit $x_1, x_2, \ldots, x_n$ et B des entiers, est-ce qu'il existe un sous-ensemble des entiers de $x_1, x_2, \ldots, x_n$ dont la somme fait B ? On note alors I les indices de ces entiers.
(KNAPSACK) Soit n objets de volume $v_1, v_2, \ldots, v_n$ et de poids $p_1, p_2, \ldots, p_n$, soit deux entiers V et P, existe-t-il un sous-ensemble $J \subset [\![1; n]\!]$ de sorte que $\sum_{i \in J} v_i \leq V$ et $\sum_{i \in J} p_i \geq P$?
La transformation est la suivante, partant d'une instance $(x_1, x_2, \ldots, x_n, B)$ de (SUBSET-SUM ?), on construit une instance $(v_1, v_2, \ldots, v_m, p_1 p_2, \ldots, p_m, V, P)$ de (KNAPSACK). Pour cela, on pose $m = n$, $v_i = p_i = x_i$ et $V = P = B$.
Cette transformation est polynomiale puisqu'elle ne prend que le temps de recopier le entiers 2 fois chacun : $O(\sum \log(x_i) + \log(B))$ ce qui est bien polynomial en la taille de l'instance (elle même égale à $\sum \log(x_i) + \log(B)$).

Si $(x_1, x_2, \ldots, x_n, B)$ est une instance positive alors il existe $I \subset [\![1; n]\!]$ tel que $\sum_{i \in I} x_i = B$. Donc $\sum_{i \in I} v_i = V$ et $\sum_{i \in I} p_i = P$. Posons donc $J = I$ et on a bien une réponse positive à l'instance $(v_1, v_2, \ldots, v_m, p_1 p_2, \ldots, p_m, V, P)$.

Si $(v_1, v_2, \ldots, v_m, p_1 p_2, \ldots, p_m, V, P)$ est une instance positive alors il existe $J \subset [\![1; n]\!]$ de sorte que $\sum_{i \in J} v_i \leq V$ et $\sum_{i \in J} p_i \geq P$. Donc $\sum_{i \in J} x_i \leq B$ et $\sum_{i \in J} x_i \geq B$. Donc $\sum_{i \in J} x_i = B$. Posons donc $I = J$ et on a bien une réponse positive à l'instance $(x_1, x_2, \ldots, x_n, B)$.

Donc on a bien conservation de la positivité et la négativité est instances.

Donc il existe une réduction polynomiale de Karp de (SUBSETSUM ?) vers (KNAPSACK).

19. Montrer que (SET COVER) $\preccurlyeq$ (DST).

▶ **Correction**

(SET COVER) Soit un ensemble X, un ensemble S de parties de X (donc chaque élément $s \in S$ est un sous-ensemble de X) et un entier k, existe-t-il un sous-ensemble C de S couvrant X (chaque élément de X est dans au moins un ensemble de C) et de taille inférieure à k ?

(DST) Soit G un graphe orienté, un nœud r, des poids $\omega : \mathbb{E} \to \mathbb{N}$, un ensemble Y de nœuds de G et un entier k', G possède-t-il un sous-graphe de G possédant un chemin de r vers chaque nœud de Y et possiblement d'autres nœuds dont la somme des poids des arcs est inférieure à k' ? (j'ai un peu simplifié la définition par rapport à la correction du TD1, pour éviter des détails inutiles dans la preuve)

La transformation est la suivante, partant d'une instance (X, S, k) de (SET COVER), on construit une instance (G, r, Y, ω, k') de (DST). Le graphe G et les poids ω sont les suivants :

— ajouter un nœud r

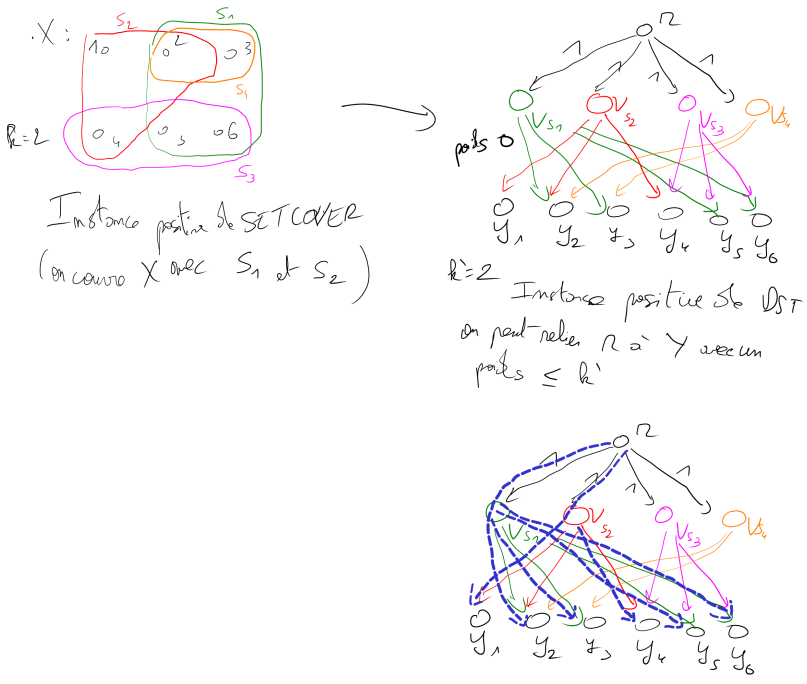
— ajouter un nœud v_s par ensemble $s \in S$

— ajouter un nœud y_x par élément $x \in X$.

— relier r à tous les sommets v_s avec un arc (r, v_s) de poids 1

— relier v_s à chaque sommet y_x tel que $x \in s$ avec un arc de poids 0

On pose ensuite $Y = \{y_x, x \in X\}$ et $k' = k$.



Cette transformation est polynomiale : construire G prend un temps $O(1+|S|+|X|+|S|+|S||X|)$. Construire Y prend un temps $O(|X|)$ et construire k' prend un temps $\log(k)$. La transformation est donc quadratique en la taille de (X, S, k).

Si (X, S, k) est une instance positive alors il existe un sous-ensemble C de S de taille inférieure ou égale à k qui couvre X. Pour chaque élément x de X, il existe un ensemble s_x de C qui contient x. Considérons maintenant le sous-graphe T de G qui contient tous les arcs (r, v_s) pour $s \in C$, ainsi que tous les arcs (v_{s_x}, y_x) pour $x \in X$. Puisque $s_x \in C$, alors (r, v_{s_x}) est dans T, donc il existe un chemin de r vers y_x dan sT. Le poids de T est $|C| \leq k = k'$. Donc (G, r, Y, ω, k') est une instance positive de (DST).

Si (G, r, Y, ω, k') est une instance positive de (DST), il existe un sous-graphe T de G possédant un chemin de r vers chaque nœud de Y et possiblement d'autres nœuds dont la somme des poids des arcs est inférieure à k'. Puisque seuls les arcs (r, v_s) sont de poids strictement positifs (égaux à 1), alors T contient au plus k' arcs de la sorte, donc contient au plus k' nœuds de type v_s. Posons $C = \{s | v_s \in T\}$. On a donc $|C| \leq k' = k$. Soit $x \in X$, montrons qu'il existe $s \in C$ tel que $c \in s$. On sait que T contient un chemin de r vers y_x. Ce chemin passe nécessirement par un nœud v_s, sous la forme (r, v_s, y_x). Par construction, cela signifie : que $s \in C$ et $x \in s$. Donc x est bien couvert par C. Donc (X, S, k) est une instance positive.

Donc on a bien conservation de la positivité et la négativité est instances.

Donc il existe une réduction polynomiale de Karp de (SETCOVER) vers (DST).

20. Montrer que (SET COVER) $\preccurlyeq$ (UST).

► **Correction**

Celle-ci est plus complexe. On utilise une réduction très similaire à la précédente mais on travaille dans un graphe non orienté, il faut donc faire attention au fait qu'on a des arêtes qui peuvent aller dans les deux sens et pas seulement des arcs qui ne peuvent aller que de r vers les nœuds de Y.

On sait que (SAT) est NP-Complet et que (TAU) est Co-NP-Complet, pour quels problèmes pouvez vous affirmer qu'ils sont (Co-)NP-Complet ou (Co-)NP-Difficile.

► **Correction**

La 4 prouve que (3COL?) est dans NP et la 5 prouve que (COHAM) est dans Co-NP. La 8 et la 9 prouvent que (3SAT ?) et (3TAU ?) sont respectovivement NP-Difficiles et Co-NP-Difficiles. La 10 e la 11 prouvent que QBF est à la fois NP-Difficile et Co-NP-Difficile.

Exercice 2 — *(SUBSET SUM est NP-Complet)*

On rappelle que (SUBSET SUM) est le problème suivant : soit Y un ensemble d'entiers et $s \in \mathbb{N}$, existe-t-il un sous-ensemble Z de Y dont la somme fait s ?

On veut démontrer que (SUBSET SUM) est NP-Complet.

1. Montrez que le problème est dans NP.

► **Correction**

(SUBSET SUM) est résolu par l'algorithme non déterministe suivant : pour chaque entier $y \in Y$, décider de manière non déterministe de mettre y dans Z ou non. Vérifier ensuite si la somme des entiers de Z fait s.

Cet algorithme est polynomial car il y a $|Y|$ choix non déterministe en temps constant suivi d'une somme des entiers de Z en temps $O(\sum_{y \in Z} \log(y))$ et une comparaison en $O(\max(\max_{y \in Y}(\log(y)), \log(s)))$.

Il résout bien l'instance. Si la réponse est OUI, alors il existe un ensemble Z dont la somme fait s. Si l'algorithme fait ses choix non détertministe de sorte à construire cet ensemble, il répondra OUI. Cette instance est donc faiblement acceptée. Dans le cas où la réponse est NON, quel que soit ses choix, la somme des entiers de Z sera différente de s et l'instance est fortement refusée.

2. Soit $\mathcal{I}$ une instance de (3-SAT ?), on veut transformer $\mathcal{I}$ en une instance $\mathcal{J}$ de (SUBSET SUM ?) en temps polynomial telle que $\mathcal{I}$ est positive si et seulement si $\mathcal{J}$ est positive. On pose Y l'ensemble des entiers de $\mathcal{J}$ et s la somme cible à atteindre.
   - On suppose que $\mathcal{I}$ a n variables $x_1, \ldots, x_n$ et m clauses $C_1, C_2, \ldots, C_m$
   - Les nombres de $\mathcal{J}$ ont $n+m$ chiffres et seront donc compris entre 0 et 10^{n+m}.
   - Pour chaque variable x_i de $\mathcal{I}$, on ajoute un entier x_i à $\mathcal{J}$ tel que le i^{e} chiffre est 1 et tel que le $n+j^{\text{e}}$ chiffre est 1 si et seulement si x_i est dans la j^{e} clause de $\mathcal{I}$. Les autres chiffres sont 0.
   - Pour chaque variable x_i de $\mathcal{I}$, on ajoute un entier $\bar{x_i}$ à $\mathcal{J}$ tel que le i^{e} chiffre est 1 et tel que le $n+j^{\text{e}}$ chiffre est 1 si et seulement si $\bar{x_i}$ est dans la j^{e} clause de $\mathcal{I}$. Les autres chiffres sont 0.
   - Pour chaque clause C_j de $\mathcal{I}$, on ajoute à $\mathcal{J}$ deux nombres égaux r_j et s_j où le $n+j^{\text{e}}$ chiffre est 1. Les autres chiffres sont 0.
   - s est le nombre où les n premiers chiffres sont 1 et les autres sont 3.

   (a) Décrivez $\mathcal{J}$ si $\mathcal{I} = (\bar{x_1} \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3})$.

   ▶ **Correction**

   On a $n = 3$ et $m = 4$. On a alors les entiers suivants :

x_1	1000011
$\bar{x_1}$	1001100
x_2	0101100
$\bar{x_2}$	0100011
x_3	0011000
$\bar{x_3}$	0010111
r_1	0001000
s_1	0001000
r_2	0000100
s_2	0000100
r_3	0000010
s_3	0000010
r_4	0000001
s_4	0000001
s	1113333

   (b) Décrivez $\mathcal{J}$ si $\mathcal{I} = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$.

   ▶ **Correction**

   On a $n = 2$ et $m = 4$. On a alors les entiers suivants :

x_1	101010
$\bar{x_1}$	100101
x_2	011100
$\bar{x_2}$	010011
r_1	001000
s_1	001000
r_2	000100
s_2	000100
r_3	000010
s_3	000010
r_4	000001
s_4	000001
s	113333

   (c) Montrez que la transformation se fait en temps polynomial.

▶ **Correction**

Pour écrire un entier x en décimal, cela prend un temps $\log_1 0(x)$. Les entiers étant compris entre 0 et 10^{n+m}, la complexité en temps pour écrire les entiers de l'instance est $O(n+m)$. Puisqu'on écrit $2n+2m+1$ entiers, la complexité en temps de la transformation est $O((n+m)\cdot(2n+2m+1))$, donc bien polynomiale en la taille de l'instance de 3SAT (*i.e.* $n+m$).

(d) En utilisant les deux exemples, montrez que, si $\mathcal{I}$ est une instance positive alors $\mathcal{J}$ est une instance positive.

▶ **Correction**

Si $\mathcal{I}$ est une instance positive, alors il existe une affectation des variables $x_1, x_2, \ldots, x_n$ qui rende les clauses $C_1, C_2, \ldots, C_m$ vraies. Créons un sous-ensemble Z des entiers de $\mathcal{J}$ dont la somme fait s.

Soit a_i l'affectation VRAI ou FAUX de x_i. Si a_i est VRAI alors on ajoute l'entier x_i à Z et sinon on ajoute l'entier $\bar{x_i}$ à Z.

Soit $C_j = (l_1 \vee l_2 \vee l_3)$ une clause. On sait que la clause est vérifiée par l'affectation, donc l_1 ou l_2 ou l_3 est vrai. Si un seul de ces trois littéraux est vrai, on ajoute r_j et s_j à Z. Si deux de ces trois littéraux sont vrais, on ajoute seulement r_j à Z. Et si les trois sont vrais, alors on n'ajoute aucun entier parmi r_j et s_j.

On peut noter que, dans tous les cas, le nombre d'entiers dans Z où le $n+j^{\text{e}}$ chiffre est 1 est exactement 3 : les entiers correspondant aux littéraux vrais et r_j ou s_j s'ils ont été ajoutés. On peut également noter le nombre d'entiers où le i^{e} chiffre vaut 1, pour $i \leq n$, est égal à 1.

On peut donc voir que, lorsqu'on additionne tous les nombres de Z, il n'y a aucune retenue et le nombre obtenu est $\underbrace{1111\ldots11}_{n \text{ fois}}\underbrace{3333\ldots333}_{m \text{ fois}}$. Donc l'instance $\mathcal{J}$ est positive.

(e) On suppose que $\mathcal{J}$ est une instance positive et on veut montrer que $\mathcal{I}$ est également positive. Il existe un sous-ensemble $Z \subset Y$ dont la somme des éléments est s.

i. Montrez que $x_i \in Z \Leftrightarrow \bar{x_i} \notin Z$.

▶ **Correction**

On peut tout d'abord remarquer que, même si on additionnait tous les nombres de Y, alors on n'aurait aucune retenue, quelque soit k, il n'y a jamais plus de 5 nombres dont le k^{e} chiffre est un 1, tous les autres étant 0. Ainsi, pour obtenir un 1 dans le i^{e} chiffre de la somme des entiers de Z, Z doit contenir exactement un nombre dont le i^{e} chiffre est un 1.

Puisque x_i et $\bar{x_i}$ sont les seuls nombres vérifiant cette propriété, $x_i \in Z \Leftrightarrow \bar{x_i} \notin Z$.

ii. Soit $C_j = (l_1 \vee l_2 \vee l_3)$ une clause de $\mathcal{I}$, montrez que l'un des entiers l_1 or l_2 or l_3 est dans X.

▶ **Correction**

En poursuivant le raisonnement débuté à la question précédente, pour obtenir un 3 dans le $n+j^{\text{e}}$ chiffre de la somme des entiers de Z, Z doit contenir exactement trois nombre dont le $n+j^{\text{e}}$ chiffre est un 1. Il n'y a que 5 nombres vérifiant cette propriété : $l_1, l_2, l_3, r_j et s_j$. Donc au moins un des entiers l_1, l_2 ou l_3 est dans Z.

iii. En déduire que $\mathcal{I}$ est satifiable.

▶ **Correction**

Considérons l'affectation suivante a_i des variables x_i : si l'entier x_i est dans Z alors a_i est VRAI. Sinon a_i est FAUX.

Considérons la clause $C_j = (x_1 \vee x_2 \vee x_3)$. D'après la question précédente, on sait que un des entiers x_1, x_2 ou x_3 est dans Z, donc a_1 ou a_2 ou a_3 est VRAI. Donc

C_j est VRAI. Ainsi toutes les clauses sont satisfaites par cette affectation. Donc $\mathcal{I}$ est satisfiable. Considérons la clause $C_j = (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$. D'après la question précédente, on sait que un des entiers $\bar{x_1}, \bar{x_2}$ ou $\bar{x_3}$ est dans Z, donc d'après les questions précédentes un des entiers x_1 ou x_2 ou x_3 n'est pas dans Z donc a_1 ou a_2 ou a_3 est FAUX. Donc C_j est VRAI.

Le raisonnement est similaire pour les clauses mélangeant littéraux positifs et négatifs. Ainsi toutes les clauses sont satisfaites par cette affectation. Donc $\mathcal{I}$ est satisfiable.

3. Déduire des questions précédentes que (SUBSET SUM) est NP-Complet.

► **Correction**

D'après les questions précédentes, on sait qu'il existe une réduction polynomiale de Karp de (3SAT ?) vers (SUBSET SUM). Puisque (3SAT ?) est NP-Complet, alors (SUBSET SUM) est NP-Difficile. Enfin, puisque (SUBSET SUM) est dans NP d'après la question 1, on sait qu'il est NP-Complet.

4. Pour quels problèmes de l'exercice 1 pouvez vous maintenant affirmer qu'ils sont NP-Complets ou NP-difficiles ?

Exercice 3 — *(SET COVER est NP-Complet)*

On rappelle que (SUBSET SUM) est le problème suivant : soit X un ensemble, S un ensemble de sous-ensembles de X et $K \in \mathbb{N}$, existe-t-il un sous-ensemble C de S de taille inférieure à K couvrant X ? (c'est à dire que pour tout $x \in X$, il existe $s \in C$ tel que $x \in s$).

On veut démontrer que (SET COVER) est NP-Complet.

1. Montrez que le problème est dans NP.

► **Correction**

(SET COVER) est résolu par l'algorithme non déterministe suivant : pour chaque ensemble $s \in S$, décider de manière non déterministe de mettre s dans C ou non. Vérifier ensuite si $|C| \leq K$, pour chaque $x \in X$, s'il existe $s \in C$ tel que $x \in s$.

Cet algorithme est polynomial car il y a $|S|$ choix non déterministe en temps constant suivi d'une comparaison en temps $O(\max(\log(|S|), \log(K)))$. Enfin pour la vérification de la couverture, on parcours C pour tout X. En supposant que l'appartenance de x à s peut se faire en $O(1)$, cette vérification se fait en $O(|C||X|) = O(|S||X|)$ ce qui est polynomial en la taille de l'instance de Set Cover.

Il résout bien l'instance. Si la réponse est OUI, alors il existe un ensemble C de taille inférieure à K couvrant X. Si l'algorithme fait ses choix non déterministe de sorte à construire cet ensemble C, il répondra OUI. Cette instance est donc faiblement acceptée. Dans le cas où la réponse est NON, quel que soit ses choix, l'ensemble C sera de taille supérieure stricte à K ou ne couvrira pas les éléments de X, et donc l'instance sera fortement refusée.

2. Soit $\mathcal{I}$ une instance de (3-SAT ?), on veut transformer $\mathcal{I}$ en une instance $\mathcal{J}$ de (SET COVER) en temps polynomial telle que $\mathcal{I}$ est positive si et seulement si $\mathcal{J}$ est positive. On pose X l'ensemble d'éléments de $\mathcal{J}$, S l'ensemble de parties de X et K le nombre d'ensemble que l'on peut utiliser.
   - On suppose que $\mathcal{I}$ a n variables $x_1, \ldots, x_n$ et m clauses $C_1, C_2, \ldots, C_m$
   - X possèdera $n + m$ éléments $e_1, e_2, \ldots, e_{n+m}$ et S possèdera $2n$ éléments.
   - Pour chaque variable x_i de $\mathcal{I}$, on ajoute deux ensembles x_i et $\bar{x_i}$ à S. Ces deux ensembles contiennent e_i.
   - Si x_i appartient à la clause C_j, l'ensemble x_i contient e_{n+j}.
   - Si $\bar{x_i}$ appartient à la clause C_j, l'ensemble $\bar{x_i}$ contient e_{n+j}.
   - $K = n$

   (a) Décrivez $\mathcal{J}$ si $\mathcal{I} = (\bar{x_1} \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3})$.

▶ **Correction**

On a $n = 3$ et $m = 4$.

On obtient l'instance suivante

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
x_1	×					×	×
$\bar{x_1}$	×			×	×		
x_2		×		×	×		
$\bar{x_2}$		×				×	×
x_3			×	×			
$\bar{x_3}$			×		×	×	×

On a enfin $K = 3$.

(b) Décrivez $\mathcal{J}$ si $\mathcal{I} = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$.

▶ **Correction**

On a $n = 2$ et $m = 4$.

On obtient l'instance suivante

	e_1	e_2	e_3	e_4	e_5	e_6
x_1	×		×		×	
$\bar{x_1}$	×			×		×
x_2		×	×	×		
$\bar{x_2}$		×			×	×

On a enfin $K = 2$.

(c) Montrez que la transformation se fait en temps polynomial.

▶ **Correction**

On construit l'ensemble X en $O(n+m)$ et l'ensemble S en $O(2n \cdot |X|) = O(2n \cdot (n+m))$. On construit l'entier K en $O(\log(n))$ donc la transformation est bien polynomiale en n et m, la taille de l'instance de (3SAT ?).

(d) En utilisant les deux exemples, montrez que, si $\mathcal{I}$ est une instance positive alors $\mathcal{J}$ est une instance positive.

▶ **Correction**

Si $\mathcal{I}$ est une instance positive, alors il existe une affectation des variables $x_1, x_2, \ldots, x_n$ qui rende les clauses $C_1, C_2, \ldots, C_m$ vraies. Créons un sous-ensemble C de S de taille inférieure à K couvrant X.

Soit a_i l'affectation VRAI ou FAUX de x_i. Si a_i est VRAI alors on ajoute l'ensemble x_i à C et sinon on ajoute l'ensemble $\bar{x_i}$ à C. On peut voir que l'ensemble C contient exactement $n = K$ ensembles et que les éléments e_i pour $i \leq n$ sont couverts.

Soit $C_j = (l_1 \vee l_2 \vee l_3)$ une clause. On sait que la clause est vérifiée par l'affectation, donc l_1 ou l_2 ou l_3 est vrai. Donc l'ensemble l_1 ou l_2 ou l_3 est dans C. Puisque ces 3 ensembles contiennent l'élément e_{n+j}, tous les éléments de X sont couverts. Donc l'instance $\mathcal{J}$ est positive.

(e) On suppose que $\mathcal{J}$ est une instance positive et on veut montrer que $\mathcal{I}$ est également positive. Il existe un sous-ensemble $C \subset S$ tels que chaque élément de X est dans au moins un ensemble de C et C contient au plus K ensembles.

i. Montrez que $x_i \in C \Leftrightarrow \bar{x_i} \notin C$.

▶ **Correction**

On sait que, pour tout $i \leq n$, e_i est couvert par C. Puisque seuls x_i et $\bar{x_i}$ couvrent cet élément, alors au moins un des deux est dans C. S'il contient les deux, alors C contiendra en tout au moins $n+1$ ensembles (x_i, $\bar{x_i}$ et un parmi x_j et $\bar{x_j}$ pour tout $j \neq i$). Puisque C ne peut contenir plus que $K = n$ ensembles, alors il ne peut y avoir qu'un seul ensemble parmi x_i et $\bar{x_i}$ dans C.

Donc C contient exactement un ensemble parmi ces deux là.

ii. Soit $C_j = (l_1 \vee l_2 \vee l_3)$ une clause de $\mathcal{I}$, montrez que $l_1 \in C$ ou $l_2 \in C$ ou $l_3 \in C$.

▶ **Correction**

Puisque seuls les ensembles l_1, l_2 et l_3 couvrent l'élément e_{n+j} et que C couvre X, au moins un de ces 3 là est dans C.

iii. En déduire que $\mathcal{I}$ est satifiable.

▶ **Correction**

Considérons l'affectation suivante a_i des variables x_i : si l'ensemble x_i est dans C alors a_i est VRAI. Sinon a_i est FAUX. Considérons la clause $C_j = (x_1 \vee x_2 \vee x_3)$. D'après la question précédente, on sait que un des ensembles x_1, x_2 ou x_3 est dans C, donc a_1 ou a_2 ou a_3 est VRAI. Donc C_j est VRAI. Considérons la clause $C_j = (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$. D'après la question précédente, on sait que un des ensembles $\bar{x_1}, \bar{x_2}$ ou $\bar{x_3}$ est dans C, donc d'après les questions précédentes un des ensembles x_1 ou x_2 ou x_3 n'est pas dans C donc a_1 ou a_2 ou a_3 est FAUX. Donc C_j est VRAI.

Le raisonnement est similaire pour les clauses mélangeant littéraux positifs et négatifs. Ainsi toutes les clauses sont satisfaites par cette affectation. Donc $\mathcal{I}$ est satisfiable.

3. Déduire des questions précédentes que (SET COVER) est NP-Complet.

▶ **Correction**

D'après les questions précédentes, on sait qu'il existe une réduction polynomiale de Karp de (3SAT ?) vers (SET COVER). Puisque (3SAT ?) est NP-Complet, alors (SET COVER) est NP-Difficile. Enfin, puisque (SET COVER) est dans NP d'après la question 1, on sait qu'il est NP-Complet.

4. Pour quels problèmes de l'exercice 1 pouvez vous maintenant affirmer qu'ils sont NP-Complets ou NP-difficiles ?

Exercice 4 — *(CHROMA est NP-Complet)*

On rappelle que (CHROMA) est le problème suivant : soit $G = (V, E)$ un graphe et $K \in \mathbb{N}$, $K \leq |V|$, peut-on colorier V avec K couleurs de sorte que deux nœuds voisins dans G n'aient pas la même couleur.

On veut démontrer que (CHROMA) est NP-Complet.

1. Montrez que le problème est dans NP.

▶ **Correction**

(CHROMA) est résolu par l'algorithme non déterministe suivant : on note $c_1, c_2, \ldots, c_K$ les K couleurs possibles, pour chaque noeud $v \in V$ et chaque couleur c_i, décider de manière non déterministe de colorier v avec c_i. Vérifier ensuite pour toute arête (v, w) si v et w n'ont pas la même couleur..

Cet algorithme est polynomial car il y a $|V|K \leq |V|^2$ choix non déterministe en temps constant et, pour la vérification de la coloration, on parcours E. Donc l'algorithme est en $O(|V|^2 + |E|)$.

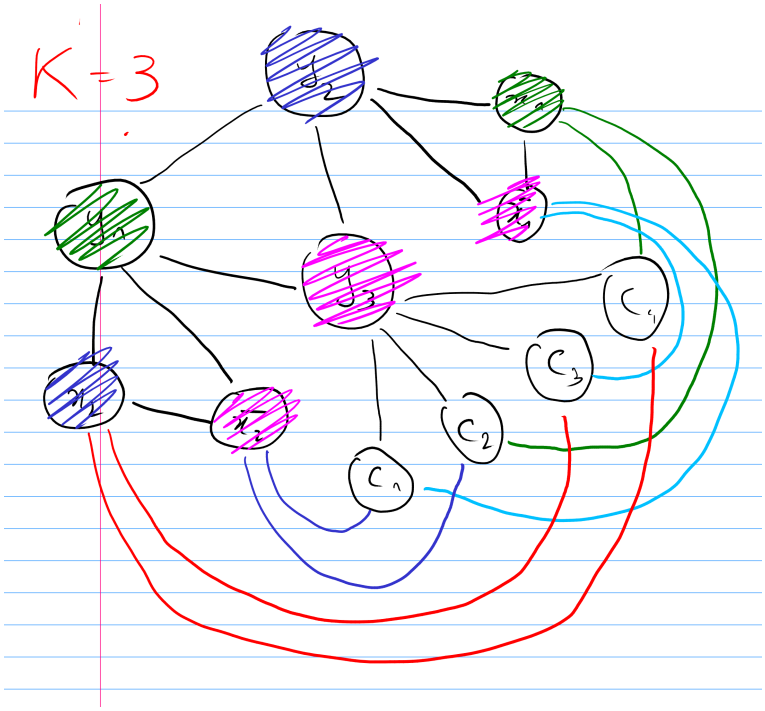
Il résout bien l'instance. Si la réponse est OUI, alors il existe une coloration valide de V avec au plus K couleurs. Si l'algorithme fait ses choix non déterministe de sorte à construire cette coloration, il répondra OUI. Cette instance est donc faiblement acceptée. Dans le cas où la réponse est NON, quel que soit ses choix, la coloration ne sera pas valide et l'instance sera fortement refusée.

2. Soit $\mathcal{I}$ une instance de (3-SAT ?), on veut transformer $\mathcal{I}$ en une instance $\mathcal{J}$ de (CHROMA) en temps polynomial telle que $\mathcal{I}$ est positive si et seulement si $\mathcal{J}$ est positive. On pose $G = (V, E)$ le graphe $\mathcal{J}$ et K le nombre de couleurs que l'on peut utiliser.
   - On suppose que $\mathcal{I}$ a n variables $x_1, \ldots, x_n$ et m clauses $C_1, C_2, \ldots, C_m$
   - G possèdera $3n + m + 1$ nœuds.

— Ajouter à G une clique de $n+1$ nœuds $y_1, y_2, \ldots, y_{n+1}$.
— Pour chaque variable x_i de $\mathcal{I}$, on ajoute deux nœuds x_i et $\bar{x_i}$ à V.
— Pour chaque clause C_j de $\mathcal{I}$, on ajoute à V un nœud C_j.
— On relie x_i et $\bar{x_i}$.
— Si $i \neq j$ et $j < n+1$, on relie x_i à y_j et $barx_i$ à y_j.
— On relie y_{n+1} et C_j pour tout j.
— Si x_i n'appartient pas à la clause C_j, relier x_i et C_j.
— Si $\bar{x_i}$ n'appartient pas à la clause C_j, relier $\bar{x_i}$ et C_j.
— $K = n+1$

(a) Décrivez $\mathcal{J}$ si $\mathcal{I} = (\bar{x_1} \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3})$.
(b) Décrivez $\mathcal{J}$ si $\mathcal{I} = (x_1 \vee x_2) \wedge (\bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2})$.

► **Correction**



On peut voir ci-dessus qu'il n'est pas possible de colorier le grpahe avec 3 couleurs.

(c) Montrez que la transformation se fait en temps polynomial.

► **Correction**

Le graphe G possède $3n+m+1$ nœuds et est donc construit en $O((3n+m+1)^2)$. On construit l'entier K en $O(\log(n+1))$ donc la transformation est bien polynomiale en n et m, la taille de l'instance de (SET COVER).

(d) En utilisant les deux exemples, montrez que, si $\mathcal{I}$ est une instance positive alors $\mathcal{J}$ est une instance positive.

► **Correction**

Si $\mathcal{I}$ est une instance positive, alors il existe une affectation des variables $x_1, x_2, \ldots, x_n$ qui rende les clauses $C_1, C_2, \ldots, C_m$ vraie. Créons un sous-ensemble C de S de taille inférieure à K couvrant X.
Soit a_i l'affectation VRAI ou FAUX de x_i. Construisons une coloration du graphe valide avec $K = n+1$ couleurs. On va appeler les couleurs $c_1 c_2 ... c_{n+1}$. On attribue la couleur c_i à y_i (pas de conflits dans la clique des y_i).

Si a_i est VRAI, on colorie x_i avec la couleur c_i et $\bar{x_i}$ avec c_{n+1}. Sinon a_i est FAUX, on colorie x_i avec la couleur c_{n+1} et $\bar{x_i}$ avec c_i. (Pas de conflit car x_i et $\bar{x_i}$ ne sont pas reliés à y_i et y_{n+1})

Soit C_j une clause. Supposons sans perte de généralité que $C_j = (x_1 \vee x_2 \vee x_3)$. On sait que cette clause est vraie donc a_1 est VRAI ou a_2 est VRAI ou a_3 est VRAI.

Prenons la couleur du noeud x_i pour lequel a_i est VRAI. Puisque a_i est VRAI, on sait que x_i est colorié avec c_i donc on colorie le noeud C_j avec la couleur c_i. Sans perte de généralité on suppose que x_1 est VRAI. Donc C_j est colorié avec c_1. Or le noeud C_j est relié à tous les x_i qui ne sont pas dans la clause et tous les $\bar{x_i}$ qui ne sont pas dans la clause. Donc à $\bar{x_1}$, $\bar{x_2}$, $\bar{x_3}$, x_4 et $\bar{x_4}$, x_5 et $\bar{x_5}$.

$\bar{x_1}$ est colorié avec la couleur c_{n+1} car a_1 est VRAI. Les autres noeuds x_i et $\bar{x_i}$ sont coloriés avec les couleurs c_i et c_{n+1} donc pas avec la couleur c_1. (On peut généraliser ce résultat à toutes les clauses quelque soit leurs littéraux)

Donc il est possible de colorier le graphe avec K couleurs. Donc l'instance $\mathcal{J}$ est positive.

(e) On suppose que $\mathcal{J}$ est une instance positive et on veut montrer que $\mathcal{I}$ est également positive. Il existe une coloration de G avec au plus K couleurs. Soit c_v la couleur du nœud v.

i. Montrez que $c_{x_i} = c_{y_i}$ et $c_{\bar{x_i}} = c_{y_{n+1}}$ ou $c_{x_i} = c_{y_{n+1}}$ et $c_{\bar{x_i}} = c_{y_i}$.

► **Correction**

On sait que x_i est relié à tous les y_j avec $j \neq i$ et $j < n+1$. Donc x_i ne peut pas être de la même couleur que ces noeuds.

Or il n'y a que K couleurs et il existe K noeuds y_j. Donc les seules couleurs possibles de x_i sont c_{y_i} et $c_{y_{n+1}}$. Pareil pour $\bar{x_i}$.

Puisque x_i et $\bar{x_i}$ sont reliés, il n'ont pas la même couleur donc l'un des deux est colorié avec c_{y_i} et l'autre avec $c_{y_{n+1}}$.

ii. Soit $C_j = (l_1 \vee l_2 \vee l_3)$ une clause de $\mathcal{I}$, montrez que $c_{C_j} = c_{l_1}$ ou $c_{C_j} = c_{l_2}$ ou $c_{C_j} = c_{l_3}$.

► **Correction**

C_j est relié à tous les autres noeuds que l_1, l_2 et l_3. Et à y_{n+1}. Donc C_j ne peut pas avoir la couleur $c_{y_{n+1}}$ ni les couleurs c_{x_i} avec $x_i \neq l_1, l_2, l_3$ ni les couleurs $c_{\bar{x_i}}$ avec $\bar{x_i} \neq l_1, l_2, l_3$.

Donc C_j ne peut être colorié qu'avec les couleurs de l_1, l_2 ou l_3.

iii. En déduire que $\mathcal{I}$ est satifiable.

► **Correction**

Soit l'affectation des variables suivantes :

si x_i est colorié avec c_{y_i} alors x_i est VRAI (dans ce cas $\bar{x_i}$ est colorié avec $c_{y_{n+1}}$ sinon x_i est FAUX (et il est colorié avec $c_{y_{n+1}}$) Montrons que cette affection satisfait la formule, donc chaque clause.

Soit une clause $C_j = (x_1 \vee x_2 \vee x_3)$ (sans perte de généralité). Montrons que x_1 ou x_2 ou x_3 est VRAI. C'est à dire montrons que x_1 est colorié avec c_{y_1} ou x_2 est colorié avec c_{y_2} ou x_3 est colorié avec c_{y_3}

D'après la question ii) on sait que C_j est colorié avec c_{x_1} ou c_{x_2} ou c_{x_3}. on sait aussi que C_j n'est pas colorié avec $c_{y_{n+1}}$.

Or si ni x_1 ni x_2 ni x_3 ne sont VRAI alors ils sont tous coloriés avec $c_{y_{n+1}}$. Il y aurait contradiction donc l'une des 3 variables est vraie.

Donc la formule est satisfiable.

3. Déduire des questions précédentes que (CHROMA) est NP-Complet.

▶ **Correction**

D'après les questions précédentes, on sait qu'il existe une réduction polynomiale de Karp de (3SAT ?) vers (CHROMA). Puisque (3SAT ?) est NP-Complet, alors (CHROMA) est NP-Difficile. Enfin, puisque (CHROMA) est dans NP d'après la question 1, on sait qu'il est NP-Complet.

4. Pour quels problèmes de l'exercice 1 pouvez vous maintenant affirmer qu'ils sont NP-Complets ou NP-difficiles ?

Exercice 5 — *Quelques preuves*

1. Montrer que la relation de réduction est transitive.

▶ **Correction**

Supposons que $\Pi_1 \preccurlyeq \Pi_2$ et $\Pi_2 \preccurlyeq \Pi_3$. Montrons que $\Pi_1 \preccurlyeq \Pi_3$.

On sait qu'il existe un algorithme $\mathcal{A}$ qui transforme les instances x_1 de Π_1 en instance x_2 de Π_2 en temps polynomial de sorte que x_1 est positive si et seulement si x_2 est positive.

On sait qu'il existe un algorithme $\mathcal{B}$ qui transforme les instances x_2 de Π_2 en instance x_3 de Π_3 en temps polynomial de sorte que x_2 est positive si et seulement si x_3 est positive.

Considérons l'algorithme $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$. Cet algorithme transforme les instances x_1 de Π_1 en instance x_3 de Π_3. L'instance $\mathcal{A}(x_1)$ de Π_2 est positive si et seulement si l'instance x_1 est positive. De plus l'instance $x_3 = \mathcal{B}(\mathcal{A}(x_1))$ de Π_3 est positive si et seulement si l'instance $\mathcal{A}(x_1)$ est positive. Donc x_3 est positive si et seulement si x_1 est positive.

$\mathcal{A}$ étant polynomial, il existe une constante c telle que, pour toute instance x_1 de Π_1, le temps de calcul de $\mathcal{A}$ ne dépasse pas $|x_1|^c$ opérations. Puisque, pour écrire chaque bit de x_2, il faut une opération, on sait que x_2 ne peut avoir une taille plus grande que $|x_1|^c$. Puisque $\mathcal{B}$ est polynomial, il existe une constante d telle que, pour toute instance x_2 de Π_2, le temps de calcul de $\mathcal{B}$ ne dépasse pas $|x_2|^d$ opérations. Donc le temps de calcul de $\mathcal{C}$ ne dépasse pas $(|x_1|^c)^d$ ce qui est polynomial en la taille de x_1.

Donc il existe bien une réduction polynomiale de Karp de Π_1 vers Π_3.

2. Supposons qu'il existe un problème $\Pi = (\mathcal{L}, \mathcal{L}_Y, \mathcal{L}_N)$ tel que Π et $\Pi^c = (\mathcal{L}, \mathcal{L}_N, \mathcal{L}_Y)$ soient NP-Complets, montrer alors que NP = Co-NP

▶ **Correction**

Soit un problème de Π_2 de NP. Puisque Π^c est NP-Complet, $\Pi_2 \preccurlyeq \Pi^c$. Or, puisque Π^c est dans Co-NP, alors tout problème plus facile que Π^c au sens de la réduction polynomiale de Karp est dans Co-NP. Donc Π_2 est dans Co-NP. Donc NP $\subset$ Co-NP.

Soit un problème de Π_2 de Co-NP, alors Π_2^c est dans NP. Puisque Π^c est NP-Complet, $\Pi_2^c \preccurlyeq \Pi^c$. Or, puisque Π^c est dans Co-NP, alors tout problème plus facile que Π^c au sens de la réduction polynomiale de Karp est dans Co-NP. Donc Π_2^c est dans Co-NP. Donc Π_2 est dans NP. Donc Co-NP $\subset$ NP.

3. Un oracle pour Π est une machine capable de résoudre Π en temps constant. Que peut-on dire si on disposait d'un oracle pour résoudre (3-SAT ?) ?

▶ **Correction**

On aurait P = NP. En effet, on pourrait résoudre 3SAT en temps constant, donc tout autre problème de NP en temps polynomial (pas en temps constant, il faut le temps de transformer ce problème en 3SAT).

Exercice 6 — *Réduction de Turing*

1. Montrer que la réduction polynomiale de Karp est un cas particulier de la réduction polynomiale de Turing.

▶ **Correction**

On rappelle que la réduction polynomiale de Turing s'écrit ainsi : $\Pi_1 \preccurlyeq_T \Pi_2$ si, en disposant d'un algorithme $\mathcal{A}$ pour résoudre Π_2 en temps constant, on peut construire un algorithme polynomial pour résoudre Π_1.

Supposons qu'il existe une réduction polynomiale de Karp entre Π_1 et Π_2, alors on a un algorithme $\mathcal{B}$ qui transforme les instances de Π_1 en instance de Π_2 en temps polynomiale et qui conserve la positivité des instances. Si on dispose d'un algorithme $\mathcal{A}$ qui résout Π_2 en temps constant, alors l'algorithme $\mathcal{A} \circ \mathcal{B}$ résout Π_1 en temps polynomial.

2. Montrer que, pour tout problème Π de NP, il existe un problème de Co-NP tel qu'il existe une réduction polynomiale de Turing de ce problème vers Π et inversement.

▶ **Correction**

Considérons le problème complémentaire Π^c de Π (on inverse les réponses OUI et NON). Si on dispose d'un algorithme $\mathcal{A}$ qui résout Π^c en temps constant, alors l'algorithme qui, connaissant une instance $\mathcal{I}$ de Π, applique $\mathcal{A}(\mathcal{I})$ et renvoie la réponse inverse résout Π. Donc $\Pi \preccurlyeq_T \Pi^c$. De même $\Pi^c \preccurlyeq_T \Pi$ puisque la technique est symétrique.

3. En déduire que NP et Co-NP sont équivalent au sens de la réduction polynomiale de Turing.

▶ **Correction**

Supposons qu'on a un problème Π NP-Complet. Alors pour tout problème Π_2 de Co-NP, puisqu'on a $\Pi_2^c \in$ NP, alors $\Pi_2^c \preccurlyeq \Pi$. Donc $\Pi_2^c \preccurlyeq_T \Pi$. Et puis que $\Pi_2 \preccurlyeq_T \Pi_2^c$. Alors $\Pi_2 \preccurlyeq_T \Pi$. Donc Π est Co-NP Difficile au sens de la réduction de Turing.

De même tous les problèmes Co-NP Complets sont NP-Difficile au sens de la réduction de Turing. Donc NP est, au sens de la réduction de Turing, plus difficile et plus facile que Co-NP.

4. Montrer que si NP $\neq$ Co-NP, alors il n'existe pas de problème de Co-NP qui soit NP-Complet et inversement si on utilise la réduction polynomiale de Karp.

▶ **Correction**

Cf l'exercice précédent.