TD 07 – Complexité: Réduction

Exercice 1. *Définition*

1. Pour deux problèmes Q et Q', que signifie $Q \leq_m^P Q'$?
   ☞ Cela signifie qu'il existe une fonction f :
   — **calculable en temps polynomial** ;
   — qui transforme les instances de Q en instances de Q' ;
   — en respectant la condition $w \in Q \Leftrightarrow f(w) \in Q'$.
   Intuitivement, $Q \leq_m^P Q'$ signifie "Q est plus facile que Q'".

   Remarque : cette définition de réduction diffère de $Q \leq_m^T Q'$ qu'on utilise en calculabilité car on impose que f soit calculable en temps polynomial alors qu'avec $Q \leq_m^T Q'$, on impose simplement que f soit calculable (peu importe en combien de temps).

2. Pourquoi si $Q \leq_m^P Q'$ et que $Q' \in P$, alors $Q \in P$?
   ☞ On peut faire un algorithme qui décide si w est une instance positive de Q de la façon suivante. On utilise une réduction polynomiale f entre Q et Q'
   — On calcule $w' = f(w)$. Ça prend un temps polynomial (par définition de f) et donc w' est de taille polynomiale par rapport à w.
   — On vérifie si $w' \in Q'$. Ça prend un temps polynomial en $|w'|$ et donc un temps polynomial en $|w|$. Ça nous dit également si $w \in Q$ par définition de f.

3. Que signifie qu'un problème Q est NP-difficile ?
   ☞ Ça signifie que pour tout problème Q' dans NP, $Q' \leq_m^T Q$. Intuitivement, Q est plus difficile que tous les problèmes dans NP.

4. Que signifie qu'un problème Q est NP-complet ?
   ☞ Ça signifie que Q est dans NP et qu'il est NP-difficile

5. Qu'est ce que cela implique si un problème NP-difficile est dans P ?
   ☞ Alors, tous les problèmes de NP sont plus faciles qu'un problème dans P donc ils sont tous dans P et P = NP.

Exercice 2. *vrai ou faux ou "on ne sait pas"*

Pour chaque affirmation, dire si elle est vraie, fausse ou si elle correspond à un problème ouvert.

1. Pour tout $Q \in$ P, $Q \leq_m^P$ PARITÉ.

   PARITÉ
   entrée : Un entier n encodé en binaire
   question : n est-il pair ?

   ☞ C'est vrai. N'importe quel problème dans P peut être réduit en temps polynomial à n'importe quel problème non trivial Q'. Un problème Q' est non trivial s'il a une instance positive w^1 et une instance négative w^0. On peut définir la réduction f de Q vers Q' comme suit :

   $$f(w) = \begin{cases} w^1 & \text{si } w \text{ est une instance positive de } Q; \\ w^0 & \text{sinon.} \end{cases}$$

   Autrement dit, comme Q est trop facile, on peut calculer directement dans f si w est une instance positive ou non. Les réductions polynomiales sont trop puissantes dans la classe P pour être intéressantes...

2. Pour tout $Q \in$ NP, $Q \leq_m^P$ PARITÉ.
   ☞ C'est vrai ssi P = NP. Intuitivement, l'énoncé signifie que tout problème dans NP "est plus facile" que le problème PARITÉ qui est dans P. Ça revient à dire que NP $\subseteq P$ et donc que P = NP.

3. Q est NP-complet ssi $\bar{Q}$ est coNP-complet

   ☞ Vrai. (Et c'est vrai en général pour toutes les classes, pas juste NP)

   Par définition, Q est dans NP ssi son complémentaire est dans coNP.

   Maintenant, supposons que $\bar{Q}$ est coNP-difficile et montrons que Q et NP-difficile. Q est NP-difficile signifie que pour tout $Q' \in \mathsf{NP}$, $Q' \leq_m^P Q$ (intuitivement, tout problème dans NP est plus facile que Q). Soit $Q' \in \mathsf{NP}$. Donc son complémentaire $\bar{Q}'$ est dans coNP. Donc, comme $\bar{Q}$ est coNP-difficile, il existe une réduction polynomiale f de $\bar{Q}'$ vers $\bar{Q}$. Donc w est une instance de positive de $\bar{Q}'$ ssi $f(w)$ est une instance positive de $\bar{Q}$. Donc w est une instance de négative de Q' ssi $f(w)$ est une instance négative de Q. Donc f est également une réduction polynomiale entre Q' et Q.

   On peut faire le même argument pour montrer que si Q est NP-difficile, alors $\bar{Q}$ et NP-difficile.

4. Il existe des problèmes NP-difficile mais pas dans NP.

   ☞ Vrai, tous les problèmes NEXP-complets par exemple.

Exercice 3. *3-TAUTOLOGIE*

Une formule CNF (pour *conjunctive normal form*) est une formule de la forme $\mu_1 \wedge \cdots \wedge \mu_m$ où pour tout $1 \leq j \leq m$, μ_j est appelée une clause et est une disjonction de littéraux positifs λ_i ou négatifs $\bar{\lambda_i}$. Exemple de formule CNF : $(\lambda_1 \vee \bar{\lambda_2} \vee \bar{\lambda_3} \vee \lambda_4) \wedge (\bar{\lambda_1} \vee \lambda_2 \vee \lambda_3 \vee \lambda_4) \wedge (\bar{\lambda_2} \vee \lambda_3)$.
Une formule 3-CNF est une formule CNF dont les clauses sont de taille au plus 3.

> **Problème 3-SAT :**
> *entrée :* Une formule 3-CNF ϕ.
> *question :* ϕ est-elle satisfiable ?

Une formule DNF (pour *disjunctive normal form*) est une formule de la forme $\mu_1 \vee \cdots \vee \mu_m$ où pour tout $1 \leq j \leq m$, μ_j est appelée une clause et est une conjonction de littéraux positifs λ_i ou négatifs $\bar{\lambda_i}$. Exemple de formule DNF : $(\lambda_1 \wedge \bar{\lambda_2} \wedge \bar{\lambda_3} \wedge \lambda_4) \vee (\bar{\lambda_1} \wedge \lambda_2 \wedge \lambda_3 \wedge \lambda_4) \vee (\bar{\lambda_2} \wedge \lambda_3)$.
Une formule 3-DNF est une formule DNF dont les clauses sont de taille au plus 3.

> **3-TAUTOLOGIE :**
> *entrée :* Une formule 3-DNF ϕ.
> *question :* ϕ est-elle une tautologie ?

1. En utilisant le fait que le problème 3-SAT est NP-complets, prouver que 3-TAUTOLOGIE est coNP-complets.

   ☞

   Pour montrer que 3-TAUTOLOGIE est coNP-complets, il suffit de montrer que son complémentaire est NP-complets.

   Le problème 3-tautologie peut être formulé de la façon suivante. Entrée : une formule ϕ en forme 3-DNF avec n variables $\lambda = \{\lambda_1, \ldots, \lambda_n\}$ et m clauses $\mu = \{\mu_1, \ldots, \mu_m\}$. Sortie : est-ce ϕ est une tautologie ? Autrement dit :

   $$\forall v : \lambda \mapsto \{\bot, \top\}, v(\phi).$$

   ou

   $$\forall v : \lambda \mapsto \{\bot, \top\}, v(\bigvee_{j \in [1,m]} \mu_j)$$

   ou

   $$\forall v : \lambda \mapsto \{\bot, \top\}, v(\bigvee_{j \in [1,m]} \bigwedge_{\mu_{j,k} \in \mu_j} \mu_{j,k})$$

   La négation de cette sortie est :

   $$\exists v : \lambda \mapsto \{\bot, \top\}, v(\bar{\phi}).$$

   ou

   $$\exists v : \lambda \mapsto \{\bot, \top\}, v(\bigwedge_{j \in [1,m]} \bar{\mu_j})$$

   ou

   $$\exists v : \lambda \mapsto \{\bot, \top\}, v(\bigwedge_{j \in [1,m]} \bigvee_{\mu_{j,k} \in \mu_j} \bar{\mu_{j,k}})$$

   Le complémentaire de 3-TAUTOLOGIE est donc exactement le problème 3-SAT : étant donnée une formule 3-CNF, est-ce qu'elle est satisfiable ?

Chaque instance de 3-SAT correspond exactement à une instance de 3-TAUTOLOGIE : on remplace juste chaque littéral par sa négation.

Exercice 4. *Cas particuliers dans P*

Montrer que les problèmes suivants sont dans P :

1. **2-COLORATION :**
entrée : Un graphe non orienté $G = (V, E)$.
question : G est 2-coloriable ?

Un graphe est k-coloriable ssi on peut associer une couleur (un nombre entre 1 et k) à chaque sommet de façon à ce que 2 sommets voisins n'aient pas la même couleur.

☞ C'est facile une fois qu'on comprend que si un sommet est colorié d'une certaine couleur alors tous ses voisins doivent être de la couleur opposée. Dans chaque composante connexe on choisit arbitrairement la couleur d'un sommet et ça nous impose la couleur de tous les autres sommets.

Exemple d'algorithme (on n'est pas obligé de l'expliciter) :

```
def est_2_Colorable(G):
        couleurs = { v dans G : None }
        Pour tout v dans G:
                si couleurs[v] == None:
                        si non colorier(G, v, couleurs, 1):
                                return Faux
        return Vrai

def colorier(G, v, couleurs, couleur ):
        couleurs[v] = couleur
        pour tous voisin u de v dans G:
        si couleurs[u] == couleur:
            return Faux  # Le graphe n'est pas 2-colorable
        sinon si couleurs[u] == None:
                        si colorier(G, u, 3 - couleur):  # Alterner les couleurs 1 et 2
                                return Faux
    return Vrai
```

L'algorithme est récursif mais on peut se convaincre que son temps d'exécution et polynomial (même linéaire en $O(n+m)$) et donc le problème est dans P.

2. Une clause de Horn est une clause comportant au plus un littéral positif (par exemple $\lambda_2 \vee \overline{\lambda_3} \vee \overline{\lambda_7} \vee \overline{\lambda_8}$. Une formule de Horn est une formule propositionnelle en forme normale conjonctive, où toutes les clauses sont des clauses de Horn.

HORN-SAT :
entrée : Une formule de Horn μ
question : μ est-elle satisfiable ?

☞ Si une clause n'a pas de littéral négatif alors elle est de la forme $\mu_j = \lambda_i$ (car il y a un seul littéral positif au plus). Ceci impose que la valuation de λ_i va être positive. On peut donc simplifier le problème en supprimant toutes les clauses ou λ_i apparaît (positivement) et en supprimant tous les littéraux $\bar{\lambda_i}$ des clauses où λ_i "apparaît négativement". (Si $\bar{\lambda_i}$ était le dernier littéral de la clause alors cela signifie la formule n'est pas satisfiable). On recommence l'opération jusqu'à ce qu'il n'existe plus aucune clause sans littéral positif. À ce moment-là, on peut satisfaire toutes les clauses restantes en fixant toutes les variables restantes à faux.

Exercice 5. *Cas particuliers encore NP-complets*

Montrer que les problèmes suivants sont NP-complets :

1. Une clause μ_j est dit **monotone** si soit tous ses littéraux sont positif ($\lambda_1 \wedge \lambda_2 \wedge \lambda_3$ par exemple), soit ils sont tous négatifs ($\bar{\lambda_1} \wedge \bar{\lambda_2} \wedge \bar{\lambda_3}$ par exemple). Une formule CNF μ est dite **monotone** si toutes ses clauses sont monotones.

SAT_MONOTONE :
entrée : Une formule CNF μ monotone.
question : μ est-elle satisfiable ?

☞ Réduction SAT $\leq_m^P$ SAT_MONOTONE.

Idée : avec notre réduction f, on remplace chaque variable λ_i par deux variables λ'_i et λ'_{i+n} qui représentent respectivement λ_i et $\bar{\lambda_i}$. On ajoute des clauses $\lambda'_i \vee \lambda'_{i+n}$ et $\bar{\lambda'_i} \vee \bar{\lambda'_{i+n}}$ pour s'assurer que la valuation de λ'_i est bien l'opposée de celle de λ'_{i+n}. Les clauses d'origines sont transformées en des clauses monotones positives. Par exemple $\bar{\lambda_1} \vee \bar{\lambda_2} \vee \lambda_3$ devient $\lambda'_{1+n} \vee \lambda'_{2+n} \vee \lambda'_3$. Clairement, une instance μ de SAT est positive si $f(\mu)$ est une instance positive de SAT_MONOTONE et la réduction proposée est clairement calculable en temps polynomial.

Donc SAT $\leq_m^P$ SAT_MONOTONE et on sait par le théorème de Cook que SAT est NP-difficile. Donc SAT_MONOTONE est NP-difficile également. De plus SAT_MONOTONE est un cas particulier de SAT qui est dans NP. Donc SAT_MONOTONE est NP-complet.

Preuve rigoureuse de la réduction :

On considère une instance μ de SAT avec n variables $\lambda = \{\lambda_1, \ldots, \lambda_n\}$ et m clauses $\mu = \{\mu_1, \ldots, \mu_m\}$.

Notre réduction f transforme cette instance μ de SAT en une instance μ' de SAT_MONOTONE avec $2n$ variables $\lambda' = \{\lambda'_1, \ldots, \lambda'_{2n}\}$ et $m + 2n$ clauses $\mu' = \{\mu'_1, \ldots, \mu'_{m+2n}\}$.

Les clauses μ'_j avec $1 \leq j \leq m$ sont similaires aux clauses μ_j mais en remplaçant les littéraux positifs λ_i par des littéraux positifs λ'_i et les littéraux négatifs $\bar{\lambda_i}$ par des littéraux positif λ'_{n+i}. Les $2n$ autres clauses sont de la forme $\lambda'_i \vee \lambda'_{i+n}$ et $\bar{\lambda'_i} \vee \bar{\lambda'_{i+n}}$ pour tout $1 \leq i \leq n$.

On note que toutes les clauses qu'on a définies sont monotones positives sauf les n de la forme $\bar{\lambda'_i} \vee \bar{\lambda'_{i+n}}$ qui sont monotones négatives.

La réduction est clairement calculable en temps polynomial. Maintenant, montrons que μ est une instance positive de SAT si et seulement si $f(\mu) = \mu'$ est une instance positive de SAT_MONOTONE.

Supposons d'abords que μ est une instance positive de SAT. Il existe donc une valuation v qui satisfait μ. On peut définir v' qui satisfait μ' de la façon suivante. Pour tout $1 \leq i \leq n$, $v'(\lambda'_i) = v(\lambda_i)$ et $v'(\lambda'_{i+n}) = \overline{v(\lambda_i)}$. Notons que toutes les $2n$ dernières clauses sont satisfaites par cette définition. Comme v satisfait μ, pour tour tout $1 \leq j \leq m$, il y a deux cas :

— soit il existe λ_i dans μ_j avec $v(\lambda_i) = 1$ et dans cas là il existe λ'_i dans μ'_j avec $v'(\lambda'_i) = v(\lambda_i) = 1$;
— soit il existe $\bar{\lambda_i}$ dans μ_j avec $v(\lambda_i) = 0$ et dans cas là il existe λ'_{i+n} dans μ'_j avec $v'(\lambda'_{i+n}) = \overline{v(\lambda_i)} = 1$;

Ceci prouve que si μ est une instance positive de SAT alors $f(\mu)$ est une instance positive de SAT_MONOTONE.

Supposons maintenant que $f(\mu) = \mu'$ est une instance positive de SAT_MONOTONE. Il existe donc une valuation v' qui satisfait μ'. Pour tout $1 \leq i \leq n$, les clauses de la forme $\lambda'_i \vee \lambda'_{i+n}$ et $\bar{\lambda'_i} \vee \bar{\lambda'_{i+n}}$ imposent que $v'(\lambda'_i) = \overline{v'(\lambda'_{i+n})}$. On définit une valuation v de μ comme suit : $v(\lambda_i) = v'(\lambda'_i)$ pour tout $1 \leq i \leq n$.

Comme v' satisfait μ', pour tour tout $1 \leq j \leq m$, il y a deux cas :

— soit il existe λ'_i ($1 \leq i \leq n$) dans μ'_j avec $v'(\lambda'_i) = 1$ et dans cas là il existe λ_i dans μ_j avec $v(\lambda_i) = v'(\lambda'_i) = 1$;
— soit il existe λ'_{i+n} dans μ'_j avec $v'(\lambda'_{i+n}) = 1$ et dans cas là il existe $\bar{\lambda_i}$ dans μ_j avec $v(\lambda_i) = v'(\lambda'_i) = \overline{v'(\lambda'_{i+n})} = 1$;

Ceci prouve que si $f(\mu) = \mu'$ est une instance positive de SAT_MONOTONE alors μ est une instance positive de SAT.

Ceci conclut la preuve de la réduction.

2.

ENSEMBLE_INDÉPENDANT :
entrée : un graphe non orienté G et un entier k ;
question : G a un ensemble indépendant de taille k (tous non reliés deux à deux) ?

☞ On prouve que 3_SAT $\leq_m^P$ ENSEMBLE_INDÉPENDANT. (On supposera que les clauses sont exactement de taille 3 pour la simplicité de la démonstration, mais ça ne change pas grand-chose sinon).

On définie la réduction de 3_SAT vers ENSEMBLE_INDÉPENDANT $f : \phi \mapsto (G, k)$ où G est un graphe avec $2n + 3m$ sommets où chaque sommet représente une variable $\lambda_i \in \lambda = \{\lambda_1, \ldots, \lambda_n\}$, sa négation $\bar{\lambda_i}$ ou un littéral $\mu_{j,k}$ (avec $j \in [1, m]$ et $k \in [1, 3]$) de ϕ. Le graphe va avoir les arêtes suivantes :

— Pour chaque variable λ_i, une arête $\{\lambda_i, \bar{\lambda_i}\}$;
— Pour chaque clause μ_j, 3 arêtes $\{\mu_{j,1}, \mu_{j,2}\}$, $\{\mu_{j,1}, \mu_{j,3}\}$ et $\{\mu_{j,2}, \mu_{j,3}\}$;
— Pour chaque littéral positif $\mu_{j,k}$ correspondant à la variable λ_i, l'arête $\{\mu_{j_k}, \bar{\lambda_i}\}$;
— Pour chaque littéral négatif $\mu_{j,k}$ correspondant à la négation de la variable $\bar{\lambda_i}$, l'arête $\{\mu_{j_k}, \lambda_i\}$.

Clairement cette réduction est polynomiale en temps. Maintenant, montrons qu'elle fonctionne bien.

D'abord, montrons qu'une solution de 3-SAT nous donne une solution de ENSEMBLE_INDÉPENDANT.

Supposons que v est une valuation satisfiant ϕ. Définissons l'ensemble V' (qu'on va prouver indépendant). Pour chaque variable λ_i, nous mettons dans V' le sommet λ_i ou $\bar{\lambda_i}$ en fonction de la valeur de $v(\lambda_i)$. Si $v(\lambda_i) = \top$, nous choisissons λ_i ; sinon, nous choisissons $\bar{\lambda_i}$. De cette manière, nous prenons exactement un sommet par variable, et ces n sommets sont indépendants entre eux.

Si v satisfait ϕ, cela signifie que chaque clause a au moins un littéral satisfait par v. Par conséquent, pour chaque clause μ_j, au moins un des trois littéraux $\mu_{j,1}$, $\mu_{j,2}$, ou $\mu_{j,3}$ est satisfait par v (et donc connecté à un sommet que nous n'avons

pas pris dans V'). Donc, pour chaque clause μ_j, nous pouvons choisir l'un des trois littéraux correspondants (disons $\mu_{j,k}$) qui est satisfait par v, et nous ajoutons ces sommets $\mu_{j,k}$ à notre ensemble indépendant. Cela garantit que les m sommets correspondant à chaque clause sont indépendants les uns des autres car nous ne prenons qu'un seul sommet par clause mais également indépendant des sommets pris dans les variables. En somme, nous avons créé un ensemble indépendant de taille $k = n + m$.

Maintenant, montrons l'implication inverse, c'est-à-dire qu'une solution pour ENSEMBLE_INDÉPENDANT nous donne une solution pour 3-SAT.

Supposons que nous ayons un ensemble indépendant de taille $k = n + m$. Cet ensemble doit contenir exactement un sommet par clause (car ce sont des cliques de taille 3) et un sommet par variable (car ce sont des cliques de taille 2).

Pour chaque variable λ_i, nous avons choisi soit λ_i soit $\bar{\lambda_i}$, mais pas les deux. Si λ_i est dans notre ensemble indépendant, cela signifie que λ_i est vrai, sinon $\bar{\lambda_i}$ est vrai.

Pour chaque clause μ_j, nous avons choisi un sommet $\mu_{j,k}$ correspondant, ce qui signifie que $\mu_{j,k}$ est le seul sommet choisi dans la clause, garantissant qu'au moins un littéral dans cette clause est vrai.

Ainsi, nous avons une interprétation des variables qui rend chaque clause vraie, ce qui signifie que notre ensemble indépendant correspond à une solution valide pour 3-SAT.

En conclusion, nous avons montré que 3-SAT se réduit à ENSEMBLE_INDÉPENDANT en temps polynomial, ce qui implique que ENSEMBLE_INDÉPENDANT est NP-difficile. ENSEMBLE_INDÉPENDANT est en outre NP-complet car un certificat valide est l'ensemble des sommets de taille k qui forme un ensemble indépendant.

3.

CLIQUE :
entrée : un graphe non orienté G et un entier k ;
question : G a une clique de taille k ?

☞ On va montrer que ENSEMBLE_INDÉPENDANT $\leq_m^P$ CLIQUE.

On définit la réduction polynomiale $f : (G, k) \mapsto (\overline{G}, k)$ où $\bar{G}$ est le complément du graphe G. Le complément d'un graphe G est un graphe ayant les mêmes sommets que G, mais deux sommets dans le complément sont adjacents si et seulement s'ils ne sont pas adjacents dans G. f est clairement calculable en temps polynomial.

(G, k) instance positive de ENSEMBLE_INDÉPENDANT $\Leftrightarrow$ G a un ensemble indépendant E de taille k $\Leftrightarrow$ $\bar{G}$ a une clique E de taille k (le même E) $\Leftrightarrow$ $(\bar{G}, k)$ est une instance positive de CLIQUE.

En conclusion, ENSEMBLE_INDÉPENDANT $\leq_m^P$ CLIQUE et CLIQUE est donc NP-difficile. En outre on peut montrer que CLIQUE est dans NP (soit en montrant avec la même réduction f que CLIQUE $\leq_m^P$ ENSEMBLE_INDÉPENDANT, soit en montrant que la clique de taille k peut servir de certificat). CLIQUE est donc NP-complet.