TD5 – Réductions, NP-difficulté, NP-complétude

Rappel 1 : Une reduction many-one polynomiale de L_1 à L_2, est donnée par un algo f en temps poly qui transforme chaque mot sur L_1 en un mot sur L_2, et préserve l'acceptation :

$$x \in L_1 \iff f(x) \in L_2.$$

On note alors $L_1 \leq_m^p L_2$, car le problème L_2 est *au moins aussi difficile* que le problème L_1.

Rappel 2 : L_2 est NP-difficile si et seulement si pour tout $L_1 \in$ NP on a $L_1 \leq_m^p L_2$.
Rappel 3 : L_2 est NP-complet si et seulement si $L_2 \in$ NP et L_2 est NP-difficile.
Rappel 4 : Si L_1 est NP-difficile et $L_1 \leq_m^p L_2$ alors L_2 est NP-difficile.

Donc, pour répondre à un exercice de la forme

« montrer que le problème **Toto** est NP-complet »,

on pourra remplir le texte à trou suivant :

(a) **Toto** $\in$ NP, car ....
....
⟨*ici soit on donne un algo dans* NP *pour décider* **Toto***, soit on utilise la char. exist. de* NP⟩ ....
....

(b) Pour le problème **Tata** que l'on sait déjà être NP-difficile, on a **Tata** $\leq_m^p$ **Toto**, car il existe la transformation $f : \Sigma^*_{\mathbf{Tata}} \to \Sigma^*_{\mathbf{Toto}}$ définie par ....
....
⟨*ici on explique comment transformer les instances de* **Tata** *en des instances de* **Toto**⟩ ....
....
...., qui est :

i. calculable en temps polynomial, car ....
....
⟨*ici on peut en général argumenter simplement : objets de taille poly faciles à générer…*⟩ ......
....

ii. et telle que, pour tout $x \in \Sigma^*_{\mathbf{Tata}}$ on a $x \in \mathbf{Tata} \iff f(x) \in \mathbf{Toto}$. En effet :

i. pour tout $x \in \Sigma^*_{\mathbf{Tata}}$ on a $x \in \mathbf{Tata} \implies f(x) \in \mathbf{Toto}$, car ....
....
....
⟨*ici se trouve le cœur de la démonstration – ventricule gauche*⟩ ....
....
....
....

ii. pour tout $x \in \Sigma^*_{\mathbf{Tata}}$ on a $f(x) \in \mathbf{Toto} \implies x \in \mathbf{Tata}$, car ....
....
....
⟨*ici se trouve le cœur de la démonstration – ventricule droit*⟩ ....
....
....
....

Donc le problème **Toto** est NP-complet.

Exercice 1. *Problèmes* NP*-complets*

Les définitions des problèmes sont données ci-après. On supposera acquis que **3-SAT** et **Clique** sont NP-complets. Pour répondre à une question on pourra utiliser les réponses précédentes.

1. Montrer que **Ensemble indépendant** est NP-complet. *Indice : réduire depuis* **Clique**.
2. Montrer que **Node cover** est NP-complet. *Indice : réduire depuis* **Clique**.
3. Montrer que **Set packing** est NP-complet. *Indice : réduire depuis* **Ensemble indépendent**.
4. Montrer que **Set covering** est NP-complet. *Indice : réduire depuis* **Node cover**.
5. Montrer que **Feedback node set** est NP-complet. *Indice : réduire depuis* **Node cover**.
6. Montrer que **Exactly-1 3-SAT** est NP-complet. *Indice : réduire depuis* **3-SAT**.
7. Montrer que **0-1 integer programming** est NP-complet. *Indice : réduire depuis* **Ex-1 3-SAT**.

3-SAT
entrée : une formule propositionnelle ϕ en forme normale conjonctive, dont toutes les clauses sont de taille exactement trois.
question : y a-t-il une affectation qui satisfait ϕ ?

Clique
entrée : un graphe non-orienté $G = (V, E)$ et un entier $k \in \mathbb{N}$.
question : G contient-il une clique de taille k ?

Ensemble indépendant
entrée : un graphe non-orienté $G = (V, E)$ et un entier $k \in \mathbb{N}$.
question : G contient-il un ensemble indépendant de taille k ?

Node cover
entrée : un graphe $G = (V, E)$ et un entier $\ell \in \mathbb{N}$.
question : existe-t-il un sous ensemble $V' \subseteq V$ tel que $|V'| \leq \ell$ et toute arête de E a l'une de ses extrémités dans V' ?

Set packing
entrée : une famille $\{S_j\}_{j\in\{1,\dots,m\}}$ d'ensembles tels que $S_j \subseteq \{1,\dots,n\}$ pour tout $j \in \{1,\dots,m\}$, et un entier $\ell \in \mathbb{N}$.
question : $\{S_j\}$ contient-elle ℓ ensembles mutuellement disjoints ?

Set covering
entrée : une famille $\{S_j\}_{j\in\{1,\dots,m\}}$ d'ensembles tels que $S_j \subseteq \{1,\dots,n\}$ pour tout $j \in \{1,\dots,m\}$, et un entier $k \in \mathbb{N}$.
question : $\{S_j\}$ contient-elle k ensembles $\{S_{j_i}\}_{i\in\{1,\dots,k\}}$ tels que $\bigcup_{i\in\{1,\dots,k\}} S_{j_i} = \bigcup_{j\in\{1,\dots,m\}} S_j$?

Feedback node set
entrée : un graphe orienté $G = (V, A)$ et un entier $k \in \mathbb{N}$.
question : existe-t-il un sous ensemble $V' \subseteq V$ tel que $|V'| \leq k$ et tout cycle dirigé de G contienne un sommet dans V' ?

Exactly-1 3-SAT
entrée : une formule propositionnelle ϕ en forme normale conjonctive, dont toutes les clauses sont de taille exactement trois.
question : y a-t-il une affectation qui satisfait exactement un littéral pour chaque clause de ϕ ?

0-1 integer programming
entrée : Une matrice M de taille $m \times n$ avec $M_{i,j} \in \mathbb{Z}$ pour tout i, j, et un vecteur d de taille m avec $d_i \in \mathbb{Z}$ pour tout i.
question : existe-t-il un vecteur x de taille n avec $x_j \in \{0, 1\}$ pour tout j et $Mx = d$?