TD 07 – Deviner, charactérisation existentielle de NPet réductions

Exercice 1. *Faire* deviner *une MT non-déterministe*

En langage de haut-niveau on utilise l'instruction *deviner* pour écrire les algorithmes non-déterministes. Dans cet exercice nous allons voir comment convertir ces instructions en machines de turing non-déterministes.

Conventions : dans les schémas les cases vides contiennent des symboles blanc $B \in \Gamma$, et les entiers en binaires sont écrits avec le bit de poids fort à gauche.

1. En imaginant que l'on a un entier $n \in \mathbb{N}$ en entrée dont on veut savoir s'il est premier, convertir en machine de Turing l'instruction *deviner(un entier $x \in \{1, \dots, \sqrt{n}\}$).*

On pourra supposer que le code de machine de Turing pour calculer la racine carrée est déjà écrit, c'est-à-dire que l'on partira de la configuration suivante :

$$\cdots\boxed{1}\boxed{0}\boxed{\#}\underbrace{\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}}_{\sqrt{n}}\underset{\substack{\uparrow\\ q_1}}{\boxed{\#}}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\cdots$$

et que l'on veut arriver dans la configuration suivante :

$$\cdots\boxed{1}\boxed{0}\boxed{\#}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\underset{\substack{\uparrow\\ q_2}}{\boxed{\#}}\underbrace{\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{0}}_{x \text{ deviné}}\boxed{}\boxed{}\boxed{}\cdots$$

2. En imaginant que l'on a en entrée un graphe non-orienté $G = (V, E)$ à n sommets dont on veut savoir s'il contient une clique de taille k, convertir en machine de Turing l'instruction *deviner(un sous-ensemble de k sommets de V).*

On pourra supposer que l'on part de la configuration suivante :

$$\cdots\boxed{0}\boxed{0}\boxed{\#}\underbrace{\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}}_{n \text{ unaire}}\boxed{\#}\underbrace{\boxed{1}\boxed{1}\boxed{1}}_{k \text{ unaire}}\underset{\substack{\uparrow\\ q_1}}{\boxed{\#}}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\cdots$$

et que l'on veut arriver dans la configuration suivante :

$$\cdots\boxed{0}\boxed{0}\boxed{\#}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{\#}\boxed{1}\boxed{1}\boxed{1}\underset{\substack{\uparrow\\ q_2}}{\boxed{\#}}\underbrace{\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}}_{\text{sous-ensemble } \textit{deviné}}\boxed{}\boxed{}\boxed{}\cdots$$

3. En imaginant que l'on a une formule ϕ à n variables (l'ensemble des variables est X avec $|X| = n$) en entrée dont on veut savoir si elle est satisfaisable, convertir en machine de Turing l'instruction *deviner(une valuation $X \to \{\bot, \top\}$).*

On pourra supposer que l'on part de la configuration suivante :

$$\cdots\boxed{1}\boxed{0}\boxed{\#}\underbrace{\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{1}}_{n \text{ unaire}}\underset{\substack{\uparrow\\ q_1}}{\boxed{\#}}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\boxed{}\cdots$$

et que l'on veut arriver dans la configuration suivante :

$$\cdots\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1&0&\#&1&1&1&1&1&1&1&\#&1&0&0&1&0&1&0&&&\\\hline\end{array}\cdots$$

$$\underset{q_2}{\uparrow}\quad\underbrace{1001010}_{\text{valuation devinée}}$$

Exercice 2. *Charactérisation existentielle de* NP

On vous rapelle la charactérisation existentielle de la classe NP : un langage A appartient à NP si et seulement si il existe une machine de Turing déterministe M qui fonctionne en temps polynomial et un polynôme $p(n)$ tels que $x \in L \iff \exists y \in \{0,1\}^{p(n)}\ M(x,y)$ accepte.
Démontrer à l'aide de cette charactérisation que les problèmes suivants sont dans NP.

1. **CNF-SAT**
   entrée : une formule propositionnelle ϕ sous forme normale conjonctive.
   question : est-ce que $mod(\phi) \neq \emptyset$?

2. **Node cover**
   entrée : un graphe $G = (V, E)$ et un entier ℓ.
   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' ?

3. **Somme de sous-ensemble**
   entrée : un ensemble d'entiers relatifs $S = \{x_1, x_2, \ldots, x_n\}$ avec $x_i \in \mathbb{Z}$ pour $1 \leq i \leq n$.
   question : S contient-il un sous-ensemble d'entiers *non vide* dont la somme vaut zéro* ?
   *c'est-à-dire un sous-ensemble $S' \subseteq S$ tel que $\sum_{x \in S'} x = 0$.

Exercice 3. *Réduisons, many-one polynomialement !*

Une reduction many-one polynomiale de L_1 à L_2, est donnée par un algo f en temps polynomial qui transforme chaque question sur L_1 en une question 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.

3-SAT
entrée : une formule propositionnelle ϕ en forme normale conjonctive, dont toutes les clauses sont de taille exactement trois.
question : est-ce que $mod(\phi) \neq \emptyset$?

Clique
entrée : un graphe non-orienté $G = (V, E)$ et un entier k.
question : G contient-il une clique[1] de taille k ?

1. Montrer que **3-SAT** $\leq_m^p$ **SAT**. (c'est facile)
2. Montrer que **CNF-SAT** $\leq_m^p$ **3-SAT**, ou **CNF-SAT** est comme **SAT** mais les formules sont en forme normale conjonctive avec un nombre non borné de littéraux par clause. (Pensez à une solution plus simple par rapport à la réduction **SAT** $\leq_m^p$ **3-SAT** vue en cours, où les formules n'étaient pas dans une forme normale.)
3. Montrer que **Node cover** $\leq_m^p$ **SAT**.
4. Montrer que **Clique** $\leq_m^p$ **Node cover**.
5. Montrer que **Clique** $\leq_m^p$ **SAT**.