

---

**TD 06 – Classes de complexité et non déterminisme**


---

**Exercice 1.**

1. Soit  $f: \mathbb{N} \rightarrow \mathbb{N}$  et  $L \in \text{DTIME}(f(n))$ . Montrer que  $\bar{L} = \Sigma^* - L$  est aussi dans  $\text{DTIME}(f(n))$ .
2. Y a-t-il de fonctions  $f: \mathbb{N} \rightarrow \mathbb{N}$  qui sont, à la fois,  $\mathcal{O}(n^2)$  et  $\Omega(n^3)$ ? Justifier la réponse.
3. Soit  $f: \mathbb{N} \rightarrow \mathbb{N}$ . Est-ce que  $\text{DTIME}(f(n)) \subseteq \text{NTIME}(f(n))$ ? Justifier la réponse.

**Exercice 2.**

1. Donner la définition de P avec des mots.
2. Donner la définition formelle de EXP.
3. Montrer que le problème suivant est dans P :

**Quatre ennemis**

*entrée* : un graphe orienté  $G = (V, E)$

*question* : le graphe  $G$  contient-il un ensemble  $V'$  de quatre sommets tels que aucune paire de sommets dans  $V'$  n'est relié par un arc ?

**Exercice 3.**

1. Donner la définition de NP avec des mots.
2. Donner la définition formelle de coNEXP.
3. Montrer que le problème suivant est dans NP :

**Plus long chemin**

*entrée* : un graphe orienté  $G = (V, E)$  et un entier  $k$

*question* :  $G$  contient-il un chemin *simple* (c'est-à-dire, sans jamais passer deux fois par le même arc) de longueur au moins  $k$  qui part de n'importe quel sommet ?

**Exercice 4.**

Pour chacune des affirmations suivantes, cocher une case pertinente.

1.  $\text{NP} \neq \text{coNP}$ .      ☐ vrai    ☐ faux    ☐ si je savais le démontrer je gagnerais 1 000 000 \$
2.  $\text{P} \neq \text{EXP}$ .      ☐ vrai    ☐ faux    ☐ si je savais le démontrer je gagnerais 1 000 000 \$
3.  $\text{P} \subsetneq \text{NP}$ .      ☐ vrai    ☐ faux    ☐ si je savais le démontrer je gagnerais 1 000 000 \$
4.  $\text{NP} = \text{NEXP}$ .    ☐ vrai    ☐ faux    ☐ si je savais le démontrer je gagnerais 1 000 000 \$

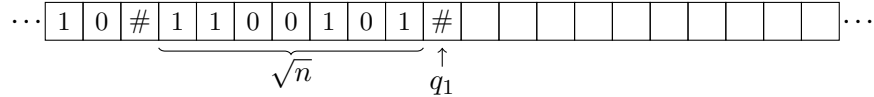
### Exercice 5.

Faire deviner une MT non-déterministe

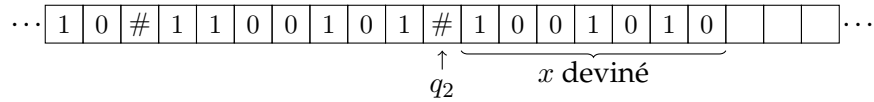
En langage de haut-niveau on utilise l'instruction *deviner* pour écrire les algorithmes non-déterministes. Dans cet exercice nous allons voir comment convertir ces instructions en machines de Turing non-déterministes.

*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.

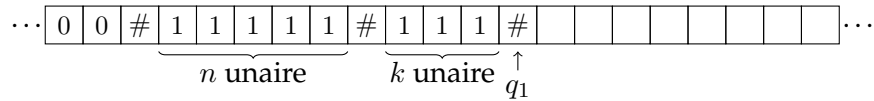
1. En imaginant que l'on a un entier  $n \in \mathbb{N}$  en entrée dont on veut savoir s'il est premier, convertir en machine de Turing l'instruction *deviner*(un entier  $x \in \{1, \dots, \sqrt{n}\}$ ).  
On pourra supposer que le code de machine de Turing pour calculer la racine carrée est déjà écrit, c'est-à-dire que l'on partira de la configuration suivante :



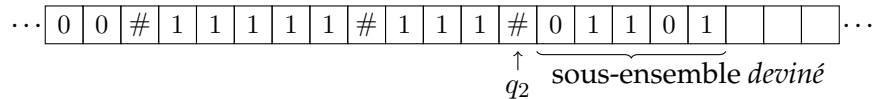
et que l'on veut arriver dans la configuration suivante :



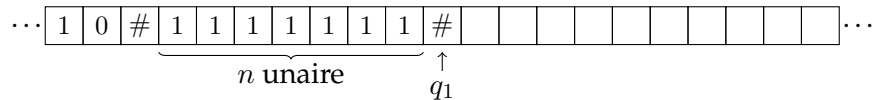
2. En imaginant que l'on a en entrée un graphe non-orienté  $G = (V, E)$  à  $n$  sommets dont on veut savoir s'il contient une clique de taille  $k$ , convertir en machine de Turing l'instruction *deviner*(un sous-ensemble de  $k$  sommets de  $V$ ).  
On pourra supposer que l'on part de la configuration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



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



3. En imaginant que l'on a une formule  $\phi$  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  $X \rightarrow \{\perp, \top\}$ ).  
On pourra supposer que l'on part de la configuration suivante :



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

