Classes de complexité en temps non déterministe

Exercice 1. Parmi les problèmes de décision suivants, déterminer s'ils sont dans la classe **NP** ou **NExp** :

1. $k-$`Color` :

   Entrée : Un graphe orienté $G = (V, E)$.

   Question : Est il possible de colorier les sommet de V avec k couleurs de telle sorte que deux sommets adjacents soient de couleurs différentes ?

2. $k-$`Clique` :

   Entrée : Un graphe $G = (V, E)$.

   Question : Est ce qu'il est possible de trouver une clique de taille k (c'est à dire k sommets tel que le sous-graphe induit est un graphe complet) ?

3. `Clique` :

   Entrée : Un graphe $G = (V, E)$ et un entier K coder en binaire.

   Question : Est ce qu'il est possible de trouver une clique de taille k (c'est à dire k sommets tel que le sous-graphe induit est un graphe complet) ?

4. `Somme Partielle`

   Entrée : Une liste d'entiers $a_1, \ldots, a_m$ et une cible entière t.

   Question : Existe t'il $S \subset \{1, \ldots, m\}$ tel que $\sum_{i \in S} a_i = t$?

5. `Arrêt Non Deterministe`

   Entrée : Une machine de Turing non déterministe $\mathcal{N}$ et un entier k en binaire.

   Question : Est ce que $\mathcal{N}$ s'arrête en k étapes ?

Exercice 2 [La classe Ntime $(t(n))$]. Considérons une fonction $t : \mathbb{N} \to \mathbb{N}$, un langage $\mathcal{L}$ appartient à la classe Ntime $(t(n))$ s'il existe une machine de Turing non déterministe $\mathcal{N}$ et une constante α telles que sur toute entrée x de taille n, il existe une exécution de $\mathcal{N}$ sur l'entrée x qui accepte x en temps inférieur à $\alpha\, t(n)$.

1. Montrer que s'il existe n_0 tel que pour tout $n \geq n_0$ on a $f(n) \leq g(n)$ alors Ntime $(f(n)) \subset$ Ntime $(g(n))$.
2. Soit $t(n) \geq n$ pour tout $n \in \mathbb{N}$. Monter que si $\mathcal{L}_1, \mathcal{L}_2 \in$ Ntime $(t(n))$ alors :
   — $\mathcal{L}_1 \cup \mathcal{L}_2 \in$ Ntime $(t(n))$;
   — $\mathcal{L}_1 \cap \mathcal{L}_2 \in$ Ntime $(t(n))$;
3. Soit $t(n) \geq n$, montrer que

$$\text{Dtime}\,(t(n)) \subset \text{Ntime}\,(t(n)) \subset \bigcup_{C \in \mathbb{N}} \text{Dtime}\left(2^{C\,t(n)}\right)$$

Exercice 3.

1. Montrer qu'un langage $\mathcal{L}$ est dans **NP** si et seulement si il existe un polynôme p et un langage $\mathcal{L}' \in \mathbf{P}$ tels que

$$x \in \mathcal{L} \iff \exists y \in \{0,1\}^{p(|x|)}\ (x, y) \in \mathcal{L}'$$

2. Montrer qu'un langage $\mathcal{L}$ est dans **NExp** si et seulement si il existe un polynôme p et un langage $\mathcal{L}' \in \mathbf{P}$ tels que

$$x \in \mathcal{L} \iff \exists y \in \{0,1\}^{2^{p(|x|)}} \ (x,y) \in \mathcal{L}'$$

Exercice 4 [`Prime` $\in$ NP $\cap$ co-NP]. En 2002, Agrawzl, Kayal et Saxena ont trouvé un algorithme qui décide la primalité d'un entier naturel en temps polynomial autrement dit `Prime` $\in$ **P**. Le but de cet exercice est de s'intéresser à la complexité du problème avant 2002 et de montrer que `Prime` $\in$ **NP** $\cap$ **co-NP**.
Dans cet exercice un entier est codé en binaire et on s'intéresse au langage où les mots sont associés aux entiers premiers. Plus précisément :

$$\mathcal{L}_2 \quad = \quad \{u \in \{0,1\}^* : u \text{ est le codage binaire d'un nombre premier}\}.$$

Un algorithme décide la primalité d'un entier écrit en binaire s'il prend en entrée un mot de $\{0,1\}^*$, accepte le mot s'il appartient à $\mathcal{L}_2$ et le refuse sinon. On étudie la complexité de tels algorithmes en fonction de la taille de l'entrée, c'est à dire la longueur du mot décrivant l'entier. Pour simplifier, on admettra que seules les opérations arithmétiques ont de l'importance. On admettra que la complexité des opérations arithmétiques multiplication, division et multiplication modulo un entier, de deux entiers écrits codés sur k bits est M(k)=$O(k\log(k)\log(\log(k)))$.

1. Donner un algorithme simple pour décider si un mot de $\{0,1\}^*$ est dans $\mathcal{L}_2$ et donner sa complexité. En déduire que `Prime` $\in$ **Exp**.
2. Montrer que $\mathcal{L}_2$ est dans la classe **co-NP**.
3. On admet le critère de Lehmer suivant : Un entier $n > 1$ est premier si et seulement s'il existe un élément $1 < a < n$ tel que $a^{n-1} \equiv 1 \mod n$ et $a^{\frac{n-1}{q}} \not\equiv 1 \mod n$ pour tout q premier diviseur de $n-1$.
   (a) A l'aide de ce théorème montrer que 2,3 et 13 sont premiers. En déduire un certificat pour être premier
   (b) Montrer que le nombre de facteurs premiers de $n-1$ est majoré par $\log(n)$.
   (c) Montrer par récurrence que P_n est codé à l'aide de $O(\log^2(n))$ bits.
   (d) Montrer que lorsque a,b et n sont donnés avec $a, b \leq n$, le coût du calcul de $a^b \mod n$ est en $O(\log(n)M(\log(n)))$.
   (e) Montrer que la vérification se fait en temps polynomial.
   (f) En déduire que `Prime` $\in$ **NP**.