Autres classes de complexité

Exercice 1 [Complexité en espace]. On introduit les classes de complexité suivantes :
- L : les langages reconnus par des machines de Turing en espaces logarithmique (l'entrée se situe sur un ruban annexe du ruban de travail) ;
- NL : les langages reconnus par des machines de Turing non déterministes en espaces logarithmique (l'entrée se situe sur un ruban annexe du ruban de travail) ;
- PSPACE : les langages reconnus par des machines de Turing en espaces polynomial ;
- NPSPACE : les langages reconnus par des machines de Turing non déterministes en espaces polynomial ;
- EXPSPACE : les langages reconnus par des machines de Turing en espaces exponentiel.

1. Montrer que

$$L \subset NL \subset P \subset NP \subset PSPACE \subset NPSPACE \subset EXP \subset NEXP \subset EXPSPACE.$$

   Remarques : On peut montrer que $PSPACE = NPSPACE$, c'est le théorème de Savitch. Mais on ne sais pas si $L = NL$ ou $P = PSPACE$!

2. Déterminer la plus petite classe auquel appartient les problèmes suivants :
   (a) `Palindrome` :

   Entrée : Un mot $u \in \{0,1\}^*$.

   Question : Est-ce que u est un palindrome ?

   (b) `Parenthésages` :

   Entrée : Un mot $u \in \{(,)\}^*$.

   Question : Est-ce que u est un bon parenthésage ?

   (c) `Accessibilité` :

   Entrée : Un graphe orienté G et deux sommet s et t.

   Question : Est-ce qu'il existe un chemin dans G allant de s à t ?

   (d) `Géographie` :

   Entrée : Un ensemble de nom de ville.

   Question : Quel joueur admet une stratégie gagnante au jeu géographie ? Dans ce jeu, on commence par une ville fixée, par exemple Lyon. Le premier joueur doit alors nommer une nouvelle ville (non encore citée) dont la première lettre est la dernière lettre dernière ville citée, par exemple Nancy. C'est alors au second joueur qui doit citer le nom d'une ville commençant par "Y". Le premier joueur dans l'impossibilité de jouer a perdu.

Exercice 2. Une *machine de Turing randomisée* $\mathcal{M}$ est une machine de Turing (déterministe !) qui prend deux entrées :
- x la suite de bits de l'instance du problème à résoudre,
- r une suite de bits aléatoires, tirés uniformément et indépendamment dans $\{0,1\}$.

La machine renvoie $\mathcal{M}(x,r) \in \{0,1\}$. Ainsi, au cours de son exécution la machine de Turing peut avoir recourt à la suite r pour choisir entre deux alternatives. La suite de bits aléatoires est différentes pour chaque exécution de la machine de Turing mais elle fera les même opérations si on lui soumet la même suite de bits aléatoires pour la même instance. De manière pratique, chaque bits de r est choisi avec une probabilité $1/2$ et de manière indépendante chaque fois que $\mathcal{M}$ en a besoin. On note $\mathbf{P}(\mathcal{M}(x,r) = 1)$ (resp. $\mathbf{P}(\mathcal{M}(x,r) = 0)$) la probabilité que $\mathcal{M}$ accepte (resp. refuse) l'entrée x suivant la suite de bits aléatoire.
Une machine de Turing randomisée *fonctionne en temps polynomial* s'il existe un polynôme $t : \mathbb{N} \to \mathbb{N}$ tel que pour toute entrée de taille n, l'algorithme randomisé s'arrête après $t(n)$ opérations (affectations, tests, opérations arithmétiques...) quel que soit la suite de bits aléatoires.

1. On se donne trois matrices A, B et C sur $\mathbb{Z}/2\mathbb{Z}$ de taille $n \times n$ et on cherche à savoir si $AB = C$. Donner en quelques mots un algorithme déterministe qui réalise cela et en donner la complexité (on compte le nombre d'opérations algébriques réalisées).
2. Considérons l'algorithme suivant :

Algorithm 1: Algorithme probabiliste pour décider si AB=C

Data: Trois matrices A, B et C de même taille

choisir aléatoirement de manière uniforme un vecteur $r = (r_1, r_2, \ldots, r_n)$;
calculer Br, puis $ABr = A(Br)$;
calculer Cr;
if $ABr = Cr$ **then**
  renvoyer `correct`;
else
  renvoyer `incorrect`;

Déterminer la complexité de cet algorithme. Avec quelle probabilité cet algorithme peut-il se tromper ?

3. Un langage $\mathcal{L} \subset \{0,1\}^*$ est dans la classe $RP(p)$ s'il existe un algorithme randomisé $\mathcal{M}$ fonctionnant en temps polynomial tel que
   — $x \in \mathcal{L} \Longrightarrow \mathbf{P}(\mathcal{M}(x,r) = 1) \geq p$,
   — $x \notin \mathcal{L} \Longrightarrow \mathcal{M}(x,r) = 0 \; \forall r \in \{0,1\}^*$.
   Montrer que $RP(p) = RP(p')$ pour tout $p, p' \in]0,1[$. On notera RP cette classe.
4. Un langage $\mathcal{L} \subset \{0,1\}^*$ est dans la classe $BPP(p)$ s'il existe un algorithme randomisé $\mathcal{M}$ fonctionnant en temps polynomial tel que
   — $x \in \mathcal{L} \Longrightarrow \mathbf{P}(\mathcal{M}(x,r) = 1) \geq p$,
   — $x \notin \mathcal{L} \Longrightarrow \mathbf{P}(\mathcal{M}(x,r) = 0) \geq p$.
   Montrer que $BBP(p) = BBP(p')$ pour tout $p, p' \in]0.5,1[$. On notera BPP cette classe.
5. Montrer que
$$P \subset RP \subset BBP \subset EXP.$$
6. Montrer que $RP \subset NP$.
7. Montrer que si $NP \subset BBP$ alors $P = NP$.
8. On ne sait pas si RP est clos par complémentaire. On dit qu'un langage $\mathcal{L} \in ZPP$ si $\mathcal{L} \in RP$ et $\{0,1\}^* \setminus \mathcal{L} \in RP$. Montrer qu'un langage est dans ZPP si et seulement s'il existe un algorithme randomisé ne se trompant jamais et dont le temps moyen d'exécution est polynomial.