Master Sciences-Technologies-Santé / Mention Informatique
Info0705 : Informatique théorique / Pascal Mignot
Année universitaire 2015-2016
Première session anticipée / Examen terminal

Université de Reims Champagne-Ardenne

Notes :
– Seul l'aide-mémoire est autorisé. Tout autre document est interdit.
– Toute propriété de l'aide mémoire utilisée devra être citée (même brièvement). Si une propriété utilisée n'est pas dans l'aide-mémoire, alors cette propriété devra être démontrée.

Questions indépendantes

1. Y-a-t-il un rapport entre la décidabilité d'un langage et la classe de complexité temporelle dans laquelle il se trouve.
2. Y-a-t-il un rapport entre le fait qu'un langage soit en espace linéaire et la classe de complexité dans laquelle il se trouve ?
3. Démontrer que $\mathbf{P} = co\mathbf{P}$.
4. Si on démontre qu'un langage de **NP** est dans **P**, qu'aura-t-on montré ?
5. Choisir un langage L de **NP** qui n'est pas trivialement dans **P**. Puis, démontrer qu'il est effectivement dans **NP** en utilisant une machine de Turing non déterministe.
6. Pour le langage L donné à la question précédente, quel est le langage reconnu en permutant accepte et rejette ?
7. Donner une machine de Turing déterministe qui décide SAT, puis donner sa complexité temporelle.
8. En utilisant le théorème de Rice, donner un moyen de construire un langage non trivial (et non vu en cours) qui n'est pas récursivement énumérable.
9. Quel est l'argument permettant d'affirmer qu'il existe des langages qui ne sont reconnus pas aucune machine de Turing ?
10. Montrer que pour tout langage régulier L, alors $L \in \mathtt{SPACE}(\log(n))$.

Exercice 1: $L = \{0^n 1^n 0^n 1^n \mid n \geq 0\}$

1. Ce langage est-il régulier ? Si oui, donner son automate déterministe fini, si non, le prouver en utilisant le lemme de l'étoile pour les langages réguliers.
2. Ce langage est-il libre du contexte ? Si oui, donner sa grammaire libre du contexte. Si non, le prouver en utilisant le lemme de l'étoile pour les grammaires libres du contexte.
3. Ce langage est-il récursivement énumérable ? Si oui, donner sa machine de Turing. Si non, le démontrer.
4. Proposer une machine de Turing (éventuellement multi-bandes) appartenant à $\mathtt{TIME}(n)$ reconnaissant L.
5. Proposer une machine de Turing appartenant à $\mathtt{SPACE}(\log(n))$ reconnaissant L. Si la machine donnée à la question précédente est déjà dans $\mathtt{SPACE}(\log(n))$, il suffira de démontrer que sa complexité spatiale est bien celle recherchée.

Exercice 2: Automate Linéaire Borné

On appelle un automate linéaire borné (ou ABL) une machine de Turing avec une bande de taille fixe (*i.e.* bornée à gauche et à droite). La tête de lecture reste bloquée si l'on tente de dépasser l'une des extrémités de la bande, à savoir bloqué à droite (resp. à gauche) si l'on est déjà à l'extrémité droite (resp. gauche) de la bande. On note Σ l'alphabet du langage et p la taille de la bande (dépend de l'ABL). On note $A_{\mathtt{ABL}}$ le langage $A_{\mathtt{ABL}} = \{< A > \mid A \text{ est un ABL qui accepte } \langle w \rangle \}$

1. Montrer que A_{ABL} est récursivement énumérable.
2. Rappeler ce qu'est une configuration pour une machine de Turing.
3. Que peut-on dire d'une machine de Turing qui, lors de son exécution, passe par deux configurations identiques ?
4. Le nombre de configurations possibles pour un ABL est-il fini, dénombrable ou infini ? S'il est fini, on donnera alors le nombre de configurations possibles ; s'il est dénombrable, on le mettra en bijection avec $\mathbb{N}$; s'il est infini, on utilisera la méthode de diagonalisation ?
5. Est-on dans les conditions de l'application du théorème de Rice. On justifiera.
6. A_{ABL} est-il décidable ?
7. Peut-on en conclure quelque chose d'applicable au fonctionnement d'un ordinateur ?

Exercice 3: Primalité et complexité

1. Soit un entier n écrit sur la bande d'entrée d'une machine de Turing. Quel est la taille de cette entrée en supposant que l'on utilise en codage raisonnable de n ?
2. Donner la fonction totalement calculable permettant d'effectuer la multiplication deux entiers.
3. Quelle est la complexité temporelle de cette multiplication ?
4. On dit que deux nombres a et b sont premiers entre eux si aucun diviseur $d > 1$ de a ne divise b, et vice-versa. Soit $\texttt{RELPRIME} = \{\langle a, b\rangle \mid a \text{ et } b \text{ sont premiers entre eux}\}$. Montrer $\texttt{RELPRIME} \in \mathbf{P}$.
5. Donner la fonction totalement calculable permettant de calculer la factorielle d'un nombre n.
6. Quelle est la complexité temporelle de cette fonction ? Si besoin, on rappelle que $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ (Stirling).
7. Soit $\texttt{PRIME} = \{\langle n\rangle \mid n \text{ est un nombre premier}\}$. Donner une machine de Turing qui décide $\texttt{PRIME}$.
8. Quelle est la complexité temporelle de l'algorithme proposé à la question précédente ?
9. Il a été démontré en 2002 que $\texttt{PRIME} \in \mathbf{P}$. Si la complexité trouvée à la question précédente est différente, qu'est-ce-que cela prouve ?
10. Que peut-on en déduire sur la classe de complexité spatiale de $\texttt{PRIME}$?
11. Est-ce que la machine de Turing décidant $\texttt{PRIME}$ donnée à la question 7 est dans $\mathbf{NL}$?

Exercice 4: SET_PARTITION

Soit $\texttt{SET\_PARTITION} = \{\langle S\rangle \mid S$ est un ensemble d'entiers, tel que $S = A \cup \overline{A}$ et $\sum_{x_i \in A} x_i = \sum_{x_i \in \overline{A}} x_i\ \}$ (*i.e.* S peut être partagé en deux ensembles dont la somme est égale).

1. Montrer que $\texttt{SET\_PARTITION} \in \mathbf{NP}$ en utilisant une machine de Turing non déterministe.
2. Montrer que $\texttt{SET\_PARTITION} \in \mathbf{NP}$ en utilisant une vérificateur.
3. Prenons un mot $\langle S, t\rangle \in \texttt{SUBSET\_SUM}$. Il existe donc un sous-ensemble A de S tel que $t = \sum_{x_i \in A} x_i$. Notons $q = \sum_{x_i \in S} x_i$ la somme de tous les éléments dans A. Quel entier r faut-il ajouter au complémentaire de A tel que $\sum_{x_i \in A} x_i = \sum_{x_i \in \overline{A} \cup \{r\}} x_i$?
4. Déduire de la question précédente une transformation f en temps polynomial qui transforme un problème de $\texttt{SUBSET\_SUM}$ en un problème de $\texttt{SET\_PARTITION}$.
5. Prouver que la transformation f définie à la question précédente est une réduction.
6. Peut-on en déduire que $\texttt{SET\_PARTITION}$ est $\mathbf{NP}$-complet ? On justifiera précisément la réponse.