TD 9 – Complexité en temps

1 Problèmes de factorisation

On considère les trois problèmes suivants.

— FindSmallestFactor : Étant donné $N \in \mathbb{N}_{>0}$ renvoie le plus petit $k \in \mathbb{N}$ avec $k \geq 2$ tel que k divise N.

— FindGreatestFactor : Étant donné $N \in \mathbb{N}_{>0}$ renvoie le plus grand $k \in \mathbb{N}$ avec $k < N$ tel que k divise N.

— HasFactor : Étant donné $(N, M) \in (\mathbb{N}_{>0})^2$ décide s'il existe un facteur non trivial de N plus petit que M.

Exercice 1. Montrer que si l'on sait résoudre un de ces problèmes en temps polynomial, alors on sait résoudre les trois. Attention, le temps polynomial se réfère à la taille de l'entrée qui est logarithmique en les nombres représentés.

Solution 1. Si l'on sait résoudre l'un des deux premiers problèmes, alors on sait factoriser N et donc résoudre les deux autres problèmes.

Pour le dernier problème, on va montrer qu'on peut s'en servir pour résoudre le premier. Il s'agit de travailler par dichotomie. On commence par $k_1 = 1$ et $k_2 = N$; ensuite, tant que $k_1 + 1 < k_2$ on choisit $M = \lfloor (k_1 + k_2)/2 \rfloor$ et on regarde s'il existe un facteur non trivial de N plus petit que M, si c'est le cas on pose $k_2 = M$ sinon on pose $k_1 = M$. Une fois que notre algorithme s'arrête, on a $k_1 + 1 = k_2$ et donc soit N n'a pas de facteur non trivial, soit k_1 est le plus petit facteur non trivial de N. À chaque étape, $k_1 - k_2$ perd un bit : on fait donc un nombre linéaire d'appels à HasFactor.

Pour la suite, vous pouvez supposer que tester la primalité d'un nombre est dans P.

Exercice 2. Montrer que HasFactor est dans NP.

Solution 2. Ce problème peut être traité en considérant le certificat $(p_1, e_1), \ldots, (p_k, e_k)$: on vérifie que chacun des p_i est premier, que $N = \prod_{i=1}^{k} p_i^{e_i}$ et que le plus petit p_i est bien plus petit que M.

Exercice 3. Montrer que HasFactor est dans coNP.

Solution 3. De la même manière que dans l'exercice précédent, on considère le certificat $(p_1, e_1), \ldots,$ (p_k, e_k) : on vérifie que chacun des p_i est premier, que $N = \prod_{i=1}^{k} p_i^{e_i}$ et que tout p_i est strictement plus grand que M.

2 Coloriabilité

Étant donné un graphe non-orienté $G = (V, E)$, un *k-coloriage* de G, pour $k \in \mathbb{N}_{>0}$, est une application $c\colon V \to [\![1, k]\!]$ associant une couleur (un nombre) de 1 à k à chaque sommet de G, de telle manière que deux sommets x et y adjacents (c'est-à-dire, tels que $\{x, y\} \in E$) n'aient jamais la même couleur ; lorsqu'un tel k-coloriage existe, on dit que G est *k-coloriable.*

Exercice 4. Montrer que le langage $3\mathrm{COL} = \{\langle G \rangle \mid G$ graphe non orienté 3-coloriable$\}$ est dans NP.

Solution 4. Direct.

Exercice 5. Montrer que le langage $2\mathrm{COL} = \{\langle G \rangle \mid G$ graphe non orienté 2-coloriable$\}$ est dans P.

Solution 5. On peut montrer qu'un graphe non-orienté G est 2-coloriable si et seulement s'il ne contient pas de cycle de longueur impaire.

Le sens direct est facile par contraposée. Pour le sens indirect, étant donné un graphe non-orienté $G = (V, E)$ qui ne contient pas de cycle de longueur impaire, considérons pour chaque composante connexe C un sommet s distingué arbitraire. Observons que pour tout sommet x de C, tous les chemins simples de x à s sont soit de longueur impaire, soit de longueur paire, autrement il existerait nécessairement deux sommets x_1 et x_2 de C reliés par deux chemins simples disjoints ayant l'un une longueur paire, l'autre une longueur impaire, c'est-à-dire qu'il existerait un cycle de longueur impaire dans C. On peut donc colorier les sommets de C en fonction de la parité des longueurs des chemins simples à s, un coloriage bien défini et respectant la condition de 2-coloriabilité.

Ceci nous donne un moyen de facilement construire une MT testant la 2-coloriabilité en temps polynomial.

3 Propriétés de clôture

Exercice 6. Montrer que P est close par intersection, union, complémentation et étoile.

Solution 6. La clôture de P par intersection, union et complémentation est directe.

Étant donné $L \in \mathsf{P}$ un langage sur un alphabet Σ et une MT $\mathcal{M}$ le décidant en temps polynomial, on construit une MT $\mathcal{M}_*$ qui décide L^* en calculant, étant donné un mot en entrée $w \in \Sigma^*$, $|w| = n > 0$, les booléens $t_{i,j}$ pour tout $i \in [\![1, n]\!], j \in [\![i, n]\!]$ indiquant si le mot $w_i \cdots w_j$ appartient à L^* ou non. Formellement, étant donné un mot en entrée $w \in \Sigma^*$, $|w| = n > 0$, $\mathcal{M}_*$ agit selon l'algorithme suivant :

pour $i \leftarrow 1 \ldots n$ **faire**
  $t_{i,i} \leftarrow \mathcal{M}(w_i)$
fin pour
pour $d \leftarrow 1 \ldots n-1$ **faire**
  pour $i \leftarrow 1 \ldots n-d$ **faire**
    $t_{i,i+d} \leftarrow \mathcal{M}(w_i \cdots w_{i+d}) \vee \bigvee_{j=i}^{i+d-1} (t_{i,j} \wedge t_{j+1,i+d})$
  fin pour
fin pour
si $t_{1,n}$ **alors**
  accepter
sinon
  rejeter
fin si.

4 Quelques propriétés de classes

Étant donné une fonction $t\colon \mathbb{N} \to \mathbb{N}$, on note $\mathsf{DTIME}(t(n))$ la classe des langages reconnus par une MT $\mathcal{M}$ telle qu'il existe une constante $\alpha \in \mathbb{R}$ pour laquelle, sur toute entrée x, on a que $\mathcal{M}(x)$ fonctionne en temps au plus $\alpha \cdot t(|x|)$.

On peut alors définir $\mathsf{P} = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(n^k)$ et $\mathsf{EXP} = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(2^{n^k})$.

De même, étant donné une fonction $t\colon \mathbb{N} \to \mathbb{N}$, on note $\mathsf{NTIME}(t(n))$ la classe des langages reconnus par une MTN $\mathcal{M}$ telle qu'il existe une constante $\alpha \in \mathbb{R}$ pour laquelle, sur toute entrée x, on a que $\mathcal{M}(x)$ fonctionne en temps au plus $\alpha \cdot t(|x|)$, quels que soient les choix qu'elle fait lors de son exécution.

On définit alors, de manière similaire, $\mathsf{NP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(n^k)$ et $\mathsf{NEXP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(2^{n^k})$.

Rappelons en outre que, pour toute classe C de langages, la classe coC est donnée par $\mathsf{coC} = \{\bar{L} \mid L \in \mathsf{C}\}$.

Exercice 7. Montrer que si $\mathsf{P} = \mathsf{NP}$, alors $\mathsf{NP} = \mathsf{coNP}$.

Solution 7. Direct.

Exercice 8. Montrer que si $\mathsf{P} = \mathsf{NP}$, alors $\mathsf{EXP} = \mathsf{NEXP}$.

Solution 8. Soit $L \in \mathsf{NEXP}$ un langage sur un alphabet Σ. Cela signifie qu'il existe $k \in \mathbb{N}$ tel que $L \in \mathsf{NTIME}(2^{n^k})$.

Alors on construit le langage $\tilde{L} = \{w\#1^{2^{|w|^k}-|w|-1} \mid w \in L\}$ sur l'alphabet $\Sigma \cup \{1, \#\}$, avec $\#$ n'appartenant pas déjà à Σ. Étant donné une MTN $\mathcal{M}$ décidant L en temps $\mathrm{O}(2^{n^k})$, on construit une MTN $\tilde{M}$ qui, étant donné un mot $x \in (\Sigma \cup \{1, \#\})^*$ en entrée, se comporte selon l'algorithme suivant :

si x n'est pas de la forme $w\#1^m$ **alors**
  rejeter
fin si ▷ À partir d'ici on suppose que $x = w\#1^m$
écrire $1^{2^{|w|^k}}$ sur une bande, en vérifiant, en parallèle, que l'on n'écrit pas plus de symboles que $|x|$;
rejeter si c'est le cas
si $|x| \neq 2^{|w|^k}$ **alors**
  rejeter
fin si
si $\mathcal{M}(w)$ accepte **alors**
  accepter
sinon
  rejeter
fin si.

Sachant qu'étant donné $|w|$, on sait écrire successivement les 1^{2^i} pour i de 0 à $|w|^k$ en alternant entre 2 bandes avec un nombre constant d'aller-retours de tête pour chaque i, on a que $\tilde{M}$ décide $\tilde{L}$ en temps $\mathrm{O}(n)$, ce qui implique que $\tilde{L} \in \mathsf{NTIME}(n) \subseteq \mathsf{NP}$. Puisque, par hypothèse, $\mathsf{P} = \mathsf{NP}$, on a donc que $\tilde{L} \in \mathsf{DTIME}(n^c)$ pour un certain $c \in \mathbb{N}$.

On peut à présent construire une MT déterministe décidant L qui, étant donné $w \in \Sigma^*$ en entrée, écrit $w\#1^{2^{|w|^k}-|w|-1}$ sur une de ses bandes puis exécute la MT montrant que $\tilde{L} \in \mathsf{DTIME}(n^c)$ sur ce mot afin de décider si le mot appartient à $\tilde{L}$ ou non, et enfin répond la même chose. Cette machine fonctionne en temps $\mathrm{O}(2^{n^k} + 2^{cn^k}) \subseteq \mathrm{O}(2^{n^{k+1}})$. Il s'ensuit que $L \in \mathsf{EXP}$.

Exercice 9. Montrer que si tout langage unaire de NP est dans P, alors $\mathsf{EXP} = \mathsf{NEXP}$.

Solution 9. Soit $L \in \mathsf{NEXP}$ un langage sur un alphabet Σ. Cela signifie qu'il existe $k \in \mathbb{N}$ tel que $L \in \mathsf{NTIME}(2^{n^k})$.

L'idée est de définir un encodage $\langle\cdot\rangle$ qui à tout mot $w \in \Sigma^*$ associe un entier $\langle w\rangle$ dans $[\![0, |\Sigma|^n - 1]\!]$. De manière similaire à l'exercice précédent, on construit ensuite le langage $\tilde{L} = \{1^{2^{\lceil \log_2(|\Sigma|)\rceil |w|^k} + \langle w\rangle} \mid w \in L\}$. On peut ensuite montrer que $\tilde{L} \in \mathsf{NP}$ et donc en déduire que $\tilde{L} \in \mathsf{P}$, ce qui nous permet finalement de montrer que $L \in \mathsf{EXP}$.