1. Machines de Turing

Exercice 1.1 Pour chacune des tâches suivantes, construire une machine de Turing en respectant les notations du cours :

1. Recopier un mot sur un alphabet à une lettre.
2. Additionner en unaire.
3. Décider du langage de Dyck $S \to SS|(S)|\varepsilon$
4. Calculer $x \mapsto x + 1$ en binaire.
5. Calculer $x \mapsto x \dot{-} 1$ en binaire.

Exercice 1.2 On note ${}^t w$ le mot *miroir* du mot w : si

$$w = w[0] \cdots w[i] w[i+1] \cdots w[\ell - 1]$$

alors

$${}^t w = w[\ell - 1] \cdots w[i+1] w[i] \cdots w[0].$$

1. Construire une machine de Turing qui calcule $w \mapsto {}^t w$ sur l'alphabet $\{a, b\}$.
   Par exemple si $w = calcul$ alors ${}^t w = luclac$.
   Un mot w est un *palindrome* ssi il est égal à son image-miroir. Par exemple *radar* et *babbcbbab* sont des palindromes.
2. Construire une machine de Turing à *deux bandes* qui décide du langage des mots palindromes sur l'alphabet $\Sigma = \{a, b\}$.
3. Construire une machine de Turing à *une bande* qui décide du langage des mots palindromes sur l'alphabet $\Sigma = \{a, b\}$.

Exercice 1.3 Construire une machine de Turing à trois bandes pour l'opération $(x, y) \mapsto x + y$ en binaire.

Exercice 1.4 Construire une machine de Turing à deux bandes pour la transformation de unaire en binaire (exemple : $1111111111111 \mapsto 1101$).

Exercice 1.5 Construire une machine de Turing pour décider chacun des langages suivants :

1. $\{a^n b^n \mid n \in \mathbb{N}\}$
2. $\{a^{2^n} \mid n \in \mathbb{N}\}$
3. $\{a^n b^n c^n \mid n \in \mathbb{N}\}$
4. $\{u \in \{a, b, c\}^* \mid |u|_a = |u|_b = |u|_c\}$

Exercice 1.6 Construire une machine de Turing avec une bande infinie dans les deux sens, dont la tête visite toutes les cases de la bande.

NB : Dans la suite on admet la thèse de Church et une description informelle des algorithmes sera suffisante pour la preuve de la calculabilité.

Terminologie : Un algorithme A (ou une machine de Turing T) *énumère* un ensemble E, si A (ou T sans entrée) travaille sans arrêt, et renvoie au fur et à mesure tous les éléments de E (et uniquement ceux-là).

Exercice 1.7 Écrire une machine de Turing à une bande qui énumère tous les mots binaires (on pourra les écrire bit de poids faible d'abord ou utiliser un ruban bi-infini).

Exercice 1.8 Les ensembles considérés sont soit une partie de $\mathbb{N}$ ou un langage sur un alphabet fini.

1. Soit E un ensemble accepté par une machine de Turing M : pour tout $w \in E$, sur l'entrée w, M s'arrête dans l'état q_{yes}. Pour tout $w \notin E$, sur l'entrée w, M s'arrête dans un état $\neq q_{yes}$ ou ne s'arrête pas. Comment énumérer E ?
2. Montrer que si E admet un algorithme énumérant, alors il existe une machine de Turing qui l'accepte.
3. Soit $L \subseteq \Sigma^*$ un langage. Montrer que L est décidable ssi L et son complémentaire $\Sigma^* \setminus L$ sont récursivement énumérables.

Exercice 1.9 Étudier la propriété de fermeture de la famille des langages récursifs pour les opérations $\cup, \cap$ et $\setminus$. Étudier les mêmes propriétés pour la famille des langages r.e.

Exercice 1.10

1. Montrer que tout ensemble fini est récursif.
2. Montrer qu'une partie infinie r.e. E de $\mathbb{N}$ est récursive ssi elle peut être énumérée dans l'ordre croissant.
3. En déduire que tout ensemble r.e. infini admet un sous-ensemble infini récursif.

Exercice 1.11 Soit

$$L = \{n \in \mathbb{N} \mid \text{ le développement décimal du nombre } \pi \text{ contient } n \text{ chiffres 3 consécutifs } \}.$$

Le langage L est-il récursivement énumérable ? récursif ?

2. Décidabilité

2.1 Pour commencer

On considère les langages :

$$L_{\text{halt}} = \{\langle M, w\rangle | M \text{s'arrête sur } w\} \qquad \text{et} \qquad L_{\text{accept}} = \{\langle M, w\rangle | M \text{accepte } w\}$$

On rappelle que L_{halt} est indécidable (cf. cours).

Exercice 2.1

1. Montrer que le langage L_{accept} se réduit à L_{halt}.
2. Montrer que le langage L_{halt} se réduit à L_{accept}. (On fait l'hypothèse que les MT possèdent un unique état acceptant q_{halt} et un unique état rejetant q_{reject}).
3. Que peut-on en déduire concernant L_{halt} et L_{accept} ?

Exercice 2.2 Montrer que le langage $L_\emptyset = \{\langle M\rangle | L(M) = \emptyset\}$ est indécidable en réduisant L_{accept} à $L_\emptyset$.

Exercice 2.3 Montrer que le langage $L_{reach} = \{\langle M, q, w\rangle | M \text{ visite l'état } q \text{ sur } w\}$ est indécidable en réduisant L_{accept} à L_{reach}.

Exercice 2.4 Montrer que le langage

$$L^\infty_{reach} = \{\langle M, q, w\rangle \mid M \text{ visite l'état } q \text{ infiniment souvent sur } w\}$$

est indécidable en réduisant L_{accept} à L^∞_{reach}.

Exercice 2.5 Montrer que L_{halt} est décidable pour la classe des MT dont l'alphabet de ruban est limité au symbole blanc : $\Gamma = \{\square\}$.

2.2 Plus difficile

Exercice 2.6 Montrer que le problème suivant est indécidable.

- Instance : $\langle M, x\rangle$ où M est une machine de Turing et x un mot d'entrée pour M
- Question : Est-ce que la machine M passe par tous ses états non terminaux sur l'entrée x ?

Exercice 2.7 On considère dans cet exercice les machines de Turing à une bande, dont la tête de lecture/écriture se déplace toujours lors de chaque transition. Formellement, on considère les machines de Turing $M = (Q, \Sigma, \delta)$ telles que $\delta : Q \times \Sigma \to Q \times \Sigma \times \{\mathbf{L}, \mathbf{R}\}$.

Montrer que le problème suivant est *décidable* :

- Instance : M machine de Turing, x mot d'entrée pour M.
- Question : La machine M effectue-t-elle un déplacement gauche sur le mot d'entrée x ?

On suppose qu'initialement la bande comporte le mot x précédé du symbole de début $\triangleright$. La tête de lecture/écriture est positionnée sur le symbole de début de bande.

Exercice 2.8 Soient n, k deux entiers tels que $n \geq k \geq 1$. $U_{n,k}$ est l'ensemble des machines de Turing à une bande définies sur l'alphabet $\Sigma = \{0, 1, \square\}$ qui possèdent n états dont k terminaux. $\mathcal{U}_n = \bigcup_{k=1}^{n} U_{n,k}$ est l'ensemble des machines de Turing qui possèdent n états.

1. Calculer le cardinal des ensembles $U_{n,k}$ et $\mathcal{U}_n$.
2. Soit $\mathcal{A}_n \subseteq \mathcal{U}_n$ le sous-ensemble des machines qui s'arrêtent sur le mot d'entrée vide. Soit $T(n)$ le nombre maximum d'opérations que fait une machine de la classe $\mathcal{A}_n$ sur la bande vide. Expliquer pourquoi la fonction $T(n)$ est totale et bien définie.
3. La fonction $T(n)$ n'est pas calculable. Pire : il n'existe pas de fonction f calculable telle que $\forall n, T(n) \leq f(n)$. Démontrer cette propriété.

Exercice 2.9 Montrer que le problème qui suit est *décidable*.

- Instance : une machine de Turing M, un mot d'entrée x et un entier n.
- Question : au cours de l'exécution de M sur l'entrée x, la tête de lecture/écriture sort-elle du segment de la bande de longueur n ?

2.3 Le problème de correspondance de Post

On considère le problème suivant, dit de correspondance de Post [1] (PCP) :

— Instance : un ensemble $I = ((u_1, v_1), \dots (u_n, v_n))$ de couples de mots finis sur un alphabet Σ

— Question : existe-t-il un entier k et une suite $i_1, \dots, i_k$ à valeurs dans $[1, n]$ tels que $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$?

(l'entier n est appelé *longueur* de l'instance)

N.B : On peut toujours choisir comme alphabet Σ l'ensemble des lettres apparaissant dans les mots $u_1, \dots, u_n, v_1, \dots, v_n$, sans changer la réponse à l'instance de PCP.

Exercice 2.10

1. Le problème a-t-il une solution pour l'instance $I = ((a, ab), (b, ab), (aba, ba), (aa, a))$?
2. Le problème a-t-il une solution pour l'instance $I = ((ab, aba), (baa, aa), (aba, baa))$?

Exercice 2.11 (Problèmes de correspondance de Post simples)

1. Montrez que le problème de correspondance de Post devient décidable si pour chaque couple (u_i, v_i), $|u_i| = |v_i|$.
2. Montrez qu'il devient aussi décidable si l'alphabet Σ est réduit à une lettre.

Exercice 2.12 (PCP de largeur bornée) Soit K un entier strictement positif. Le problème de correspondance de Post de largeur bornée par l'entier $K \geq 1$ est défini comme suit :

— Instance : un ensemble $I = ((u_1, v_1), \dots (u_n, v_n))$ de couples de mots tels que $|u_i| \leq K, |v_i| \leq K$.

— Question : existe-t-il un entier k et une suite $i_1, \dots, i_k$ à valeurs dans $[1, n]$ tels que $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$?

On admet que le PCP, en toute généralité est indécidable (voir une preuve détaillée ci-dessous) et on se demande si, pour des *petites* valeurs de K, ce problème devient décidable.

1. Traiter le cas $K = 1$.
2. Traiter le cas $K = 4$.
3. Quels sont les entiers $K \geq 1$ pour lesquels on est sûr que le problème de correspondance de Post borné par $K \geq 1$ est décidable ? indécidable ?

Exercice 2.13 Le but de ce problème est de montrer que le problème PCP est *indécidable*. Nous considérons une suite de 3 problèmes :

— le problème de correspondance de Post (PCP)

— le problème de l'accessibilité pour les systèmes semi-Thuéiens (ACC-ST)

— le problème de l'acceptation pour les machines de Turing (ACC-MT)

Nous donnerons des réductions :

$$\text{ACC} - \text{MT} \leq_m \text{ACC} - \text{ST} \leq_m \text{PCP}$$

Le problème de l'accessibilité pour les systèmes semi-Thuéiens (ACC-ST) [2] est le suivant :

1. Du nom de son auteur, Emil Post : *A variant of a recursively unsolvable problem.* Bulletin of the American Mathematical Society 52, 1946.

2. les systèmes de réécriture de mots sont appelés systèmes semi-Thuéiens, du nom du mathématicien Norvégien, Axel Thue, qui les a étudiés dès 1914, avant que la notion d'algorithme ne soit définie formellement

— Instance :

$$S := \{(\ell_i, r_i) \mid 1 \leq i \leq n\} \text{ où, pour tout } i \in [1,n],\ \ \ell_i, r_i \in \{a,b\}^* \tag{1}$$

et deux mots $u, v \in \{a,b\}^*$.

— Question : $u \to_S^* v$? C'est à dire est-ce-qu'il existe une dérivation allant de u à v dans le système semi-Thuéien S.

Le problème de l'acceptation pour les machines de Turing (ACC-MT)

— Instance : Une machine de Turing à une bande, infinie vers la droite, d'état initial q_- et d'ensemble d'états acceptants Q_+.

— Question : Est-ce que la machine M, partant de $q_- u \square^\omega$ atteint, en un nombre fini d'étapes, une configuration dont l'état appartient à Q_+ ? (c'est à dire est-ce-que M accepte le mot u ?)

Soit $I := ((u_1, v_1), \ldots, (u_n, v_n))$ une instance du Problème de Post. Introduisons une lettre x_i pour chaque entier $i \in [1,n]$ et posons $X := \{x_1, x_2, \ldots, x_n\}$. Considérons les homomorphismes de monoides libres $\varphi : X^* \to \Sigma^*,\ \ \psi : X^* \to \Sigma^*$ tels que :

$$\forall i \in [1,n],\ \ \varphi(x_i) = u_i,\ \ \psi(x_i) = v_i \tag{2}$$

1. Vérifier que l'instance I de (PCP) a la réponse "oui" ssi

$$\exists w \in X^+, \varphi(w) = \psi(w).$$

2. Soit $S(I) = \{w \in X^* \mid \varphi(w) = \psi(w)$ (c'est l'ensemble des solutions au PCP, augmenté du mot vide).
   $S(I)$ est-il clos par produit ? par facteur ? S'agit-il d'un sous-monoide de X^* ?
3. On appelle solution *atomique* de I une solution $w \in X^+$ telle que, pour toute décomposition $w = w_1 \cdot w_2$, avec $w_1, w_2 \in \mathrm{S}(I)$, $w_1 = \varepsilon$ ou $w_2 = \varepsilon$. On note $\mathrm{A}(I)$ l'ensemble des solutions atomiques de I.
   Montrer que $\mathrm{S}(I) = \mathrm{A}(I)^*$.
4. Soit S un système de réecriture sur $\{a,b\}^*$ (de la forme (1) et soient $u, v \in \{a,b\}^*$. On définit $\Phi(S,u,v)$ comme l'instance suivante du PCP :

$$((\#\#x, \#\#xu^x\#), (\#xv^x\#\#, x\#\#), (ax, xa), (bx, xb), (\#x, x\#), (\ell_1^x,\ {}^x r_1), \ldots, (\ell_n^x,\ {}^x r_n)) \tag{3}$$

   où $x, \#$ sont des nouvelles lettres ($x \notin \{a,b\}, \# \notin \{a,b\}, x \neq \#$) et les exposants ont la signification suivante : pour tout mot $w = x_1 x_2 \ldots x_m$

$$w^x = x_1 x x_2 x \ldots x_m x, \quad {}^x w = x x_1 x x_2 \ldots x x_m,\ \ \varepsilon^x = {}^x \varepsilon = \varepsilon.$$

   Montrer que, si $u = u_0 \to_S u_1 \ldots \to_S u_p = v$, alors $\Phi(S,u,v)$ a une solution utilisant p occurences de couples de l'ensemble $\{(\ell_i^x,\ {}^x r_i) \mid 1 \leq i \leq n\}$.
5. On renomme $D, F, L_a, L_b, L_\#, R_1, \ldots, R_n$ les lettres de X (prises dans l'ordre de la définition (3)), et on continue de noter φ, ψ les homomorphismes définis en (2). Ainsi :

$$\varphi(D) = \#\#x,\ \ \psi(D) = \#\#xu^x\#,\ \ \varphi(F) = \#xv^x\#\#,\ \ \psi(F) = x\#\#, \ldots, \varphi(R_n) = \ell_n^x,\ \ \psi(R_n) = {}^x r_n.$$

   Montrer que si $H \in \{D, F, L_a, L_b, L_\#, R_1, \ldots, R_n\}^*$ est une solution atomique, alors H est de la forme

$$H = D \cdot H_0 \cdot F$$

avec $H_0 \in \{L_a, L_b, L_\#, R_1, \ldots, R_n\}^*$ et

$$\varphi(H_0)\#\,{}^x v = u^x \# \psi(H_0)$$

6. Montrer que, pour tous mots $H \in \{L_a, L_b, L_\#, R_1, \ldots, R_n\}^*, u_1 \in \{a,b\}^*, v_1 \in \{a,b\}^*$,

$$\varphi(H)\#\,{}^x v_1 = u_1^x \# \psi(H) \Rightarrow [u_1 \to_S^p v_1 \text{ où } p = |H|_{R_1,\ldots,R_n}].$$

(aide : raisonner par récurrence sur $|H|_{L_\#}$).

7. Montrer que $u \to_S^* v$ si et seulement si $\Phi(S,u,v)$ a une solution.
8. Conclure que Φ est une réduction du problème ACC-ST au problème PCP.
9. Donner une réduction du problème ACC-MT au problème ACC-ST.
10. Montrer que PCP est indécidable.

Exercice 2.14 Le problème de décider si le langage d'une grammaire algébrique contient un palindrome est indécidable. Le démontrer en réduisant PCP à ce problème.

Indication : on pourra considérer la grammaire

$$S \to u_i S \tilde{v_i} | u_i \# \tilde{v_i}$$

Exercice 2.15 On dit qu'une grammaire algébrique est ambiguë si un mot admet deux arbres de dérivation. Montrer que le problème de décider si une grammaire est ambiguë est indécidable.

Indication : on pourra considérer la grammaire

$$\begin{array}{ll} S & \to S_1 | S_2 \\ S_1 & \to u_i S_1 i | u_i i \\ S_2 & \to v_i S_2 i | v_i i \end{array}$$

3. Calculabilité

Exercice 3.1 Prouvez en utilisant le théorème de Rice qu'il est impossible de décider qu'une machine de Turing reconnaît un langage infini.

Exercice 3.2 Comme on l'a vu en cours, on peut identifier un problème de décision à un langage. Si A se réduit à B et que B est un langage régulier, ceci implique-t-il que A est régulier ? Expliquez pourquoi.

Exercice 3.3 Montrez que si A est r.e. et que $\bar{A}$ se réduit à A, alors A est décidable.

Exercice 3.4 On considère le problème de déterminer si une machine de Turing à deux rubans écrit à un moment de son exécution un symbole autre que $\square$ sur son second ruban quand elle est exécutée sur un mot w. Montrer que ce problème est indécidable.

Exercice 3.5 (Problème du castor affairé (*busy beaver*)) On dénote par $\mathcal{T}_1$ l'ensemble des machines de Turing déterministe à une bande sur l'alphabet $\{1, \square, \triangleright\}$. On définit la productivité d'une machine de $\mathcal{T}_1$ par :

— le nombre de caractères 1 contigus qu'elle écrit au début du ruban à partir d'un ruban vide, si elle s'arrête
— 0 si elle ne s'arrête pas

Pour tout entier $n \geq 1$, on note $p(n)$ la productivité maximale des machines de $\mathcal{T}_1$ à n états.

1. Calculer $p(1)$
2. Montrer que la fonction p est croissante
3. Montrer qu'il existe une constante k telle que, pour tout $n \geq 1$, $p(n+k) \geq 2p(n)$.
4. En déduire une minoration de $p(n)$ par une fonction exponentielle de n.
5. Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction totale. On dit qu'une machine T de $\mathcal{T}_1$ *calcule la fonction* f si, pour tout entier $n \in \mathbb{N}$, partant de la configuration

$$q_- \triangleright 1^n \square 1 \square^\omega$$

(i.e. le symbole de début suivi de n symboles 1 puis du mot $\square 1$ suivi d'une infinité de blancs, avec la tête de lecture sur le symbole de début et l'état q_-) la machine a un calcul fini, qui s'arrête dans une configuration

$$q \triangleright 1^{f(n)} \square 1 \square^\omega$$

où $q \in Q$.

Montrer que, si p est calculable par une machine de $\mathcal{T}_1$, alors $n \mapsto p(p(n)) + 1$ est aussi calculable par une machine de $\mathcal{T}_1$,

6. En déduire qu'il existe un entier $\ell \geq 0$ tel que

$$\forall n \in \mathbb{N}, p(n+\ell) \geq p(p(n)) + 1.$$

7. Démontrer que la fonction p n'est pas calculable par une machine de $\mathcal{T}_1$.
8. Montrer que toute fonction calculable est calculable par une machine de $\mathcal{T}_1$.
   Idée : on peut coder les lettres de l'alphabet $X_0 = \{0, 1, \square, \triangleright\}$ par des mots de longueur 4 sur l'alphabet $X_1 = \{1, \square, \triangleright\}$:

$$0 \mapsto 1\square 11, \quad 1 \mapsto 11\square 1, \square \mapsto 1\square\square 1 \quad \triangleright \mapsto \triangleright\square\square\square$$

   puis simuler chaque transition d'une machine $\mathcal{M}_0$ sur X_0 par une séquence de transitions d'une machine $\mathcal{M}_1$ sur X_1.
9. Montrer que p n'est pas une fonction calculable.

4. Complexité

4.1 Logique booléenne

Exercice 4.1

1. Montrer que toute expression booléenne peut être mise en FNC (Forme Normale Conjontive) en donnant une fonction T qui envoie toute formule booléenne F sur une formule booléenne $T(F)$ telle que :

$$F \equiv T(F) \text{ et } T(F) \text{ est en forme normale conjonctive.}$$

2. Appliquer T aux formules :

$$x \to (y \vee z), \;\; x \to (y \wedge z),$$

$$x_1 \leftrightarrow x_0, \;\; \neg(x_1 \leftrightarrow x_0), \;\; x_2 \leftrightarrow (x_1 \leftrightarrow x_0), \;\; \neg(x_2 \leftrightarrow (x_1 \leftrightarrow x_0))$$

   Plus généralement, on construit une suite de formules booléennes $(\Phi_n)_{n\in\mathbb{N}}$ sur l'ensemble de variables $X = \{x_0, x_1, \ldots, x_n, \ldots\}$ en posant $\Phi_0 := x_0, \;\; \bar{\Phi}_0 := \neg\Phi_0$ et, pour tout $n \geq 0$

$$\Phi_{n+1} := x_{n+1} \leftrightarrow \Phi_n, \;\; \bar{\Phi}_{n+1} := \neg\Phi_{n+1}.$$

3. Combien de clauses comportent les FNC que vous obtenez pour $\Phi_n, \bar{\Phi}_n$? La transformation T est-elle de complexité polynomiale ?

4. On dit que deux formules F, F' sont *équisatisfiables* ssi (F est satisfiable $\Leftrightarrow$ F' est satisfiable).

   Une expression booléenne est en *3-Forme Normale Conjonctive* (3-FNC), si elle est en FNC dont chaque clause contient exactement 3 littéraux. Soit F une formule booléenne sur l'ensemble de variables X. On note $\mathrm{SF}(F)$ l'ensemble des sous-formules de F. On introduit, pour chaque sous-formule G de F une nouvelle variable Q_G et une formule Ψ_G par :

$$\begin{array}{lcll} \Psi_G := & x & \text{si } F = x \\ \Psi_G := & \neg x & \text{si } F = \neg x \\ \Psi_G := & Q_{F_1} \vee Q_{F_2} & \text{si } F = F_1 \vee F_2 \\ \Psi_G := & Q_{F_1} \wedge Q_{F_2} & \text{si } F = F_1 \wedge F_2 \end{array}$$

   et finalement

$$\Psi := Q_F \wedge \bigwedge_{G\in\mathrm{SF}(F)} T(Q_G \to \Psi_G).$$

   Montrer que F et Ψ sont équisatisfiables.

5. Que peut-on déduire des résultats précédents ?

Exercice 4.2 Montrer que le langage des expressions booléennes qui ne sont pas des tautologies est NP-complet.

Exercice 4.3 Le problème 3-SAT est NP-complet. Que peut-on dire de 2-SAT, sur la satisfaisabilité des expressions booléennes dont les clauses contiennent 2 littéraux ?

Indication : considérer un graphe orienté dont les sommets sont les variables et leurs négations. Pour chaque clause $x \vee y$, il y a un arc orienté de $\neg x$ vers y et un de $\neg y$ vers x. À quelle condition doit satisfaire le graphe pour que l'expression soit satisfaisable ?

4.2 Graphes

Exercice 4.4 Le problème du *voyageur de commerce*, TS (*Travelling Salesman*), est le suivant :

— Instance : un ensemble de n villes, une matrice D à n lignes et n colonnes, où $D_{ij} \in \mathbb{N}$ est la distance de la ville i à la ville j, et une constante $c \in \mathbb{N}$.

— Question : existe-t-il un circuit fermé passant par toutes les villes de longueur inférieure ou égale à c ?

Montrer que le problème du circuit hamiltonien se réduit polynomialement au problème TS.

Exercice 4.5 Soit $G_{p,q} = (V_{p,q}, E_{p,q})$ la grille rectangulaire de largeur p et longueur q :

$$V_{p,q} = \{(x,y) \mid 1 \leq x \leq q, 1 \leq y \leq p\}, \quad E_{p,q} = \{\{(x,y),(z,t)\} \mid (x,y) \neq (z,t) \text{ et } |x-z|+|y-t| = 1\}.$$

1. $G_{4,4}$ a-t-elle un cycle Hamiltonien ? $G_{3,3}$ a-t-elle un cycle Hamiltonien ?
2. $G_{4,4}$ a-t-elle un chemin Hamiltonien "diagonal" i.e. allant de de $(1,1)$ jusqu'à $(4,4)$? $G_{3,3}$ a-t-elle un chemin Hamiltonien "diagonal" i.e. allant de $(1,1)$ jusqu'à $(3,3)$) ?
3. Conjecturer des Conditions Nécessaires et Suffisantes sur les entiers (p,q) pour que :
   — $G_{p,q}$ ait un cycle Hamiltonien ;
   — $G_{p,q}$ ait un chemin Hamiltonien diagonal.
   Prouver vos conjectures ou (les modifier puis les prouver ou (les modifier puis les prouver ...)).

Exercice 4.6 Soit G un graphe, considéré suivant les cas orienté ou non. Démontrer que les 4 versions hamiltoniennes suivantes sont polynomialement équivalentes (i.e. chacune se réduit polynomialement à l'autre) :

1. Étant donné G non orienté, admet-il un cycle hamiltonien ?
2. Étant donné G non orienté, admet-il une chaîne hamiltonienne ?
3. Étant donné G orienté, admet-il un circuit hamiltonien ?
4. Étant donné G orienté, admet-il un chemin hamiltonien ?

Exercice 4.7 Formuler un problème de décision pour la recherche d'un plus long chemin élémentaire (ne passant pas plus d'une fois par un sommet). Ensuite réduire polynomialement une des versions du problème hamiltonien à celui-ci.

NB : Dans les exercices suivants, précisez les instances des problèmes si nécessaire et, pour démontrer la NP-compétude d'un problème (ou un langage), prouver qu'il appartient à la classe NP et qu'un problème reconnu d'être NP-complet est *polynomialement* réductible à celui-ci.

Exercice 4.8 Le problème CLIQUE est le suivant :

— Instance : Un graphe non orienté $G = (V, E)$ et un entier k.

— Question : Admet-il un sous-graphe *complet* à k sommets (appelé *k-clique*) ?

Montrer que CLIQUE est NP-complet. (Indication : réduire polynomialement le problème SAT à celui-ci.)

Exercice 4.9 Le problème STABLE est le suivant :

— Instance : Un graphe non orienté $G = (V, E)$ et un entier k.

— Question : Admet-il un *stable* à k sommets (un sous-ensemble de V' de V est un *stable*, si toute arête de E a au plus une extrémité dans V'.) ?

Montrer que STABLE est NP-complet. (Indication : réduire polynomialement CLIQUE à celui-ci).

Exercice 4.10 Le problème SOMMETS-COUVRANTS est le suivant :
- Instance : Un graphe non orienté $G = (V, E)$ et un entier k.
- Question : Admet-il une *couverture* à k sommets (un sous-ensemble de V' de V est une *couverture*, si toute arête de E a au moins une extrémité dans V') ?

Montrer que SOMMETS-COUVRANTS est NP-complet. (Indication : réduire polynomialement CLIQUE à celui-ci).

Exercice 4.11 On peut associer à chacun des 3 problèmes de *décision* qui précèdent un problème d'*optimisation*. Écrire les problèmes d'optimisation associés.

Exercice 4.12 Le problème ENSEMBLES-COUVRANTS est le suivant :
- Instance : Une classe $\mathcal{C} = \{E_1, \ldots, E_m\}$ de m ensembles finis, contenant au total n éléments, et un entier $k \leq m$.
- Question : Existe-t-il une sous-classe $\mathcal{C}'$ de $\mathcal{C}$ à k éléments telle que $|\cup_{E \in \mathcal{C}'} E| = n$?

Montrer que ENSEMBLES-COUVRANTS est NP-complet.

Exercice 4.13 Soit l'instance suivante :
- Instance : Deux graphes non orientés G et H.

On considère deux problèmes, appelés ISOMORPHISME-DE-GRAPHES et SOUS-GRAPHE, correspondant à deux questions suivantes respectivement :
- Question 1 : G et H sont-ils isomorphes ?
- Question 2 : Existe-t-il un sous-graphe de G isomorphe à H ?

1. Montrer que les problèmes CLIQUE et ISOMORPHISME-DE-GRAPHES sont chacun polynomialement réductibles au problème SOUS-GRAPHE.
2. Que peut-on en déduire ?

   Remarque : le problème ISOMORPHISME-DE-GRAPHES n'est pas connu pour être NP-complet, sans être connu pour ne pas l'être !

Exercice 4.14 Le problème k-COLORATION se définit comme suit :
- Instance : Un graphe non orientés G.
- Question : Disposant de k couleurs distinctes, peut-on colorier les sommets de G de façon à ce que deux sommets voisins reçoivent deux couleurs distinctes ?

1. Montrer que 3-SAT se réduit polynomialement au problème 3-COLORATION. En déduire que, pour $k \geq 3$, k-COLORATION est NP-complet.
2. Que peut-on dire de 2-COLORATION ?

Exercice 4.15 Le problème TRIANGLE-MONOCHROMATIQUE est le suivant :
- Instance : Un graphe non orienté G.
- Question : Existe-t-il une coloration des *arêtes* de G en deux couleurs, telle qu'il n'y ait pas de triangle *monochromatique* (de côtés de même couleur) ?

1. Colorier les arêtes du graphe K_5 (graphe complet à 5 sommets) en deux couleurs sans triangle monochromatique.
2. Montrer que toute coloration de K_6 (graphe complet à 6 sommets) en deux couleurs admet au moins un triangle monochromatique.

3. Réduire polynomialement le problème TRIANGLE-MONOCHROMATIQUE au problème SAT.

   (Attention : Cela n'implique pas que TRIANGLE-MONOCHROMATIQUE est NP-complet, mais on démontre qu'il l'est)

4.3 Equations/inéquations en nombres entiers

Exercice 4.16 On considère les deux problèmes suivants :
appelés SUBSET-SUM et SAC-A-DOS (*Knapsack*) respectivement.
SUBSET-SUM :
— Instance : Un multi-ensemble $E = \{a_1, \ldots, a_n\}$ de n entiers et un entier cible a.
— Question : Existe-t-il un sous multi-ensemble de E de somme exactement égale à a ?
SAC-A-DOS :
— Instance : n objets de poids $p_1, \ldots, p_n$ et de valeurs $v_1, \ldots, v_n$, un poids total p à ne pas dépasser et une valeur totale v à réaliser.
— Question : Un randonneur peut-il réaliser la valeur v, tout en respectant la contrainte du poids total ?

On démontre que SUBSET-SUM est NP-complet. Montrer que ce problème se réduit à SAC-A-DOS et en déduire que le problème SAC-A-DOS est NP-complet.

Exercice 4.17 Le problème PROGRAMMATION-ENTIÈRE (programmation en nombres entiers) est le suivant :
— Instance : des matrices $A \in \mathbb{M}_{m,n}(\mathbb{Z})$, $\overrightarrow{b} \in \mathbb{M}_{m,1}(\mathbb{Z})$
— Question : Existe-t-il un vecteur d'entiers naturels $\overrightarrow{x} \in \mathbb{M}_{n,1}(\mathbb{N})$ tel que

$$A \cdot \overrightarrow{x} \geq \overrightarrow{b}$$

(où l'ordre $\geq$ est l'ordre coordonnée par coordonnée sur les vecteurs).
La taille de la donnée est ici $\|(A, \overrightarrow{b})\| := \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}| + \sum_{i=1}^{m} |b_i|$.
Montrer que PROGRAMMATION-ENTIÈRE est NP-difficile.
(NB : Le problème PROGRAMMATION-ENTIÈRE est aussi dans NP).