Exercice 1: Théorie de Sprague-Grundy

Un **jeu** est un triplet (S, A, s_0) où S est un ensemble fini d'**états**, $A \subseteq S^2$ est un ensemble de **transitions** et $s_0 \in S$ est un **état initial**, tels que le graphe (S, A) est acyclique.

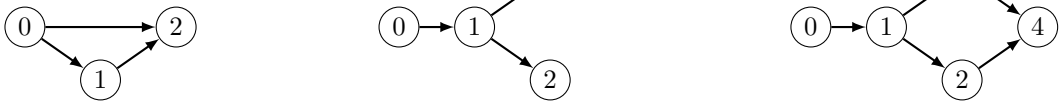
Une **stratégie** est une fonction partielle $\varphi : S \rightharpoonup S$ telle que pour tout état s où elle est définie, $(s, \varphi(s)) \in A$. On peut faire jouer deux stratégies φ_0 et φ_1 l'une contre l'autre en les faisant jouer alternativement. On définit ainsi la séquence d'états $s_1 = \varphi_0(s_0)$, $s_2 = \varphi_1(s_1)$, $s_3 = \varphi_0(s_2)$, ..., $s_k = \varphi_{(k-1) \bmod 2}(s_{k-1})$.

1. Montrer que la séquence (s_k) est finie, c'est-à-dire qu'il existe un entier n_f minimal tel que $\varphi_{n_f \bmod 2}(s_{n_f})$ n'est pas défini.

Pour deux stratégies φ_0 et φ_1 jouées par les joueurs 0 et 1, le joueur perdant est le premier joueur pour lequel la stratégie n'est plus définie. Formellement, si n_f est pair, alors le joueur 0 perd et le joueur 1 gagne. Inversement, si n_f est impair, alors le joueur 1 perd et le joueur 0 gagne.

Une **stratégie gagnante** pour le joueur $i \in \{0, 1\}$ est une stratégie qui garantit que le joueur i gagne quand il joue suivant cette stratégie, quelle que soit la stratégie jouée par l'autre joueur.

2. Parmi les jeux suivants (où l'état initial est le sommet 0), quels sont ceux qui ont une stratégie gagnante pour le joueur 0 ? Pour le joueur 1 ?



3. On définit le jeu suivant : il y a un tas de n jetons entre les deux joueurs. Chacun son tour, un joueur peut prendre $1, 2, 3$ ou 4 jetons. Le joueur qui prend le dernier jeton gagne. Expliquer comment cette description informelle du jeu peut se formaliser avec la notion formelle de jeu définie plus haut. Sous quelle condition sur n existe-t-il une stratégie gagnante pour le joueur 0 ? Pour le joueur 1 ?

Pour un jeu (S, A, s_0), on définit, $s \in S$, sa **valeur de Grundy** $G(s) \in \mathbb{N}$ par :

$$G(s) = \min(\mathbb{N} \setminus \{G(t) \mid (s, t) \in A\})$$

4. Justifier que cette relation définit bien G de manière unique.
5. Donner une condition nécessaire et suffisante sur G pour que le joueur 0 ait une stratégie gagnante. Qu'en est-il du joueur 1 ?
6. Donner le pseudo-code d'un algorithme permettant de déterminer, étant donné un jeu, quel(s) joueur(s) a(ont) une stratégie gagnante. Quelle est sa complexité en temps et en espace ?

Dans le **jeu de Nim**, il y a n tas de $t_0, \ldots, t_{n-1}$ jetons chacun entre les deux joueurs. Chacun son tour, un joueur choisit un tas et en retire autant de jetons qu'il le souhaite. Le joueur qui prend le dernier jeton gagne.

7. Montrer qu'il existe une stratégie gagnante pour le joueur 0 si et seulement si $t_0 \oplus \ldots \oplus t_{n-1} \neq 0$, où $\oplus$ représente le ou-exclusif bit à bit (ou l'addition modulo 2, bit à bit).

Pour $J_1 = (S_1, A_1, s_1)$ et $J_2 = (S_2, A_2, s_2)$, leur **somme** $J_1 + J_2 = (S, A, s_0)$ est définie par :

- $S = S_1 \times S_2$;
- $A = \{((s, t), (s, t')) \mid s \in S_1, (t, t') \in A_2\} \cup \{((s, t), (s', t)) \mid (s, s') \in A_1, t \in S_2\}$;
- $s_0 = (s_1, s_2)$.

8. Soit (s, t) un état de $J_1 + J_2$. Montrer que $G(s, t) = G_1(s) \oplus G_2(t)$, où G, G_1 et G_2 désignent les valeurs de Grundy dans $J_1 + J_2$, J_1 et J_2 respectivement.
9. Quelle est la valeur de Grundy des états du jeu de Nim ? Commenter.

Corrigé

1. Si (s_k) n'est pas finie, alors par le principe des tiroirs, il existe $i < j$ tels que $s_i = s_j$. Cela signifie que le graphe contient un cycle par définition des stratégies. C'est contradictoire avec l'hypothèse. On en déduit par ailleurs que $n_f \leqslant |S|$.
2. On commence par signaler qu'il ne peut pas exister de stratégie gagnante simultanément pour les deux joueurs pour un même graphe.
   - pour le premier graphe, $\varphi_0 : 0 \mapsto 2$ est une stratégie gagnante pour le joueur 0 ;
   - pour le deuxième graphe, $\varphi_1 : 1 \mapsto 2$ est une stratégie gagnante pour le joueur 1 ;
   - pour le troisième graphe, $\varphi_0 : 0 \mapsto 1$, $2 \mapsto 4$, $3 \mapsto 4$ est une stratégie gagnante pour le joueur 0.
3. On peut modéliser ce jeu par $J = (S, A, s_0)$ où :
   - $S = [\![0, n]\!]$;
   - $s_0 = n$;
   - $A = \{(s, s-i) \mid s \in S, i \in [\![1, 4]\!], s - i \geqslant 0\}$.

   Pour déterminer une stratégie gagnante, le mieux est de commencer par étudier de petites valeurs de n :
   - si $n \leqslant 4$, il est clair que 0 a une stratégie gagnante (prendre tous les jetons en un seul coup) ;
   - si $n = 5$, c'est 1 qui a une stratégie gagnante, car quel que soit le coup de 0, on se ramène au cas précédent en inversant les joueurs ;
   - si $n \in [\![6, 9]\!]$, 0 a une stratégie gagnante, en prenant un nombre de jetons pour ramener le total à 5, donc au cas précédent en inversant les joueurs ;
   - ...

   On en déduit (et cela peut se montrer par récurrence) que 1 a une stratégie gagnante si n est un multiple de 5, sinon c'est 0 qui a une stratégie gagnante.
4. Soit $s \in S$. Montrons par récurrence sur k, la longueur maximale d'un chemin dans (S, A) de sommet initial s, que $G(s)$ est défini de manière unique :
   - si $k = 0$, alors s est un puits (il n'a pas de voisin), donc $G(s) = 0$ est bien défini de manière unique ;
   - supposons le résultat établi jusqu'à $k \in \mathbb{N}$ fixé. Soit s ayant une longueur maximale de chemin sortant égale à $k+1$. Alors chaque voisin t de s a une longueur maximale de chemin sortant $\leqslant k$, donc $G(t)$ est défini de manière unique, donc $G(s)$ est défini de manière unique (minimum d'un ensemble non vide).

   On conclut par récurrence finie, car le graphe est acyclique.
5. Le joueur 0 a une stratégie gagnante si et seulement si $G(s_0) \neq 0$. Montrons ce résultat par récurrence sur k, la longueur maximale d'un chemin sortant de s_0 :
   - si $k = 0$, alors s_0 est un puits et $G(s_0) = 0$, et 0 n'a effectivement pas de stratégie gagnante ;
   - supposons le résultat établi jusqu'à $k \in \mathbb{N}$ fixé. Supposons que la longueur maximale d'un chemin sortant de s_0 soit $k+1$ et distinguons :
     * si $G(s_0) = 0$, alors chaque voisin t de s_0 vérifie $G(t) \neq 0$ (sinon $G(s_0)$ ne pourrait pas valoir 0). Par hypothèse de récurrence, il existe une stratégie gagnante pour le premier joueur depuis t. Ainsi, quel que soit le coup de 0, 1 a une stratégie gagnante, donc 0 n'a pas de stratégie gagnante depuis s_0.
     * si $G(s_0) \neq 0$, alors il existe un voisin t de s_0 tel que $G(t) = 0$ (sinon on aurait $G(s_0) = 0$). Par hypothèse de récurrence, il n'existe pas de stratégie gagnante pour le premier joueur depuis t. Ainsi, il existe une stratégie gagnante depuis s_0 pour le joueur 0, vérifiant $\varphi_0(s_0) = t$.

   On conclut par récurrence. De même, on en déduit que 1 a une stratégie gagnante si et seulement si $G(s_0) = 0$.
6. Il s'agit d'implémenter un parcours de graphe depuis s_0. L'algorithme ressemble également à l'algorithme glouton de coloration de graphe (selon l'ordre inverse d'un ordre topologique).

```
Entrée : Jeu J = (S, A, s_0), n = |S|.
Début algorithme
    G ← tableau de taille n contenant des −1.
    Fonction Calcul_G(s)
        Si G[s] = −1 Alors
            m ← 0.
            Pour t voisin de s Faire
                Calcul_G (t)
                m ← m + 1.
            T ← tableau de taille m + 1 contenant des Faux.
            Pour t voisin de s Faire
                T[G[t]] ← Vrai.
            Pour i de 0 à m Faire
                Si T[i] Alors
                    G[s] ← i.
                    Sortir de la boucle.
    Calcul_G (s_0).
    Si G[s_0] ≠ 0 Alors
        Renvoyer 0.
    Sinon
        Renvoyer 1.
```

À noter, la création de T et les deux boucles qui suivent permettent de trouver le plus petit entier naturel absent des nombres de Grundy des voisins en temps linéaire. Cela permet d'avoir une complexité temporelle totale en $\mathcal{O}(|S| + |A|)$ et une complexité spatiale en $\mathcal{O}(|S|)$.

7. Montrons ce résultat par récurrence sur le nombre total k de jetons restant. Appelons **score** d'une configuration la valeur $t_0 \oplus \ldots \oplus t_{n-1}$.
   - si $k = 0$, alors tous les $t_i = 0$, donc le score est nul et 0 n'a pas de stratégie gagnante ;
   - supposons le résultat établi pour $k \in \mathbb{N}$ fixé et soit un jeu de Nim à $k+1$ jetons. Distinguons :
     * si le score σ est nul, alors quel que soit i, si 0 prend $k_i > 0$ jetons dans le tas i, il reste $k + 1 - k_i \leqslant k$ jetons, et en faisant le ou-exclusif entre l'ancien score σ et le nouveau score σ', on obtient $\sigma \oplus \sigma' = t_i \oplus (t_i - k_i) \neq 0$. Comme $\sigma \oplus \sigma' = 0 \oplus \sigma' = \sigma'$, on en déduit que $\sigma' \neq 0$. Par hypothèse de récurrence, il existe une stratégie gagnante pour le joueur suivant ;
     * si le score σ est non nul, soit d la position du bit de poids fort dans σ. Il existe $i \in [\![0, n-1]\!]$ tel que le d-ème bit de t_i est différent de 0. En en déduit que $\sigma \oplus t_i < t_i$. Posons alors $k_i = t_i - \sigma \oplus t_i$. En prenant k_i jetons dans le tas i, alors le nouveau score σ' vaut $\sigma' = \sigma \oplus t_i \oplus (t_i - k_i) = \sigma \oplus t_i \oplus (\sigma \oplus t_i) = 0$. On en déduit par hypothèse de récurrence qu'il n'y a pas de stratégie gagnante pour le joueur suivant.

   On conclut par récurrence.

8. Pour $s \in S_1$, notons $\ell_1(s)$ la longueur maximal d'un chemin partant de s (et de même $\ell_2(t)$ pour $t \in S_2$).

   Montrons l'égalité par récurrence sur $\ell_1(s) + \ell_2(t)$:
   - si $\ell_1(s) + \ell_2(t) = 0$, alors ni s, ni t n'a de voisin, donc $G(s,t) = 0 = G_1(s) \oplus G_2(t)$;
   - supposons le résultat établi jusqu'à $\ell_1(s) + \ell_2(t) = k \in \mathbb{N}$ fixé et soit $(s,t) \in S$ tels que $\ell_1(s) + \ell_2(t) = k + 1$.
     * Soit $x < G_1(s) \oplus G_2(t)$. Montrons qu'il existe (s', t') un voisin de (s,t) tel que $G(s',t') = x$. Posons $y = G_1(s) \oplus G_2(t) \oplus x$. On a donc $y \neq 0$. Soit d la position du bit de poids fort de y. Sachant que $x < G_1(s) \oplus G_2(t)$, le d-ème bit dans $G_1(s) \oplus G_2(t)$ vaut 1 (et 0 dans x). Sans perte de généralité, le d-ème bit de $G_1(s)$ vaut donc 1. On en déduit que $G_1(s) \oplus y < G_1(s)$, donc il existe par définition un voisin s' de s tel que $G_1(s) = G_1(s) \oplus y$. Dès lors, (s', t) est un voisin de (s,t), et par hypothèse de récurrence, $G(s',t) = G_1(s') \oplus G_2(t) = G_1(s) \oplus G_2(t) \oplus y = x$.
     * Soit (s', t') un voisin de (s,t). Montrons que $G(s',t') \neq G_1(s) \oplus G_2(t)$. Sans perte de généralité, $t' = t$ et on a $G_1(s') \neq G_1(s)$. On en déduit que $G(s',t') = G_1(s') \oplus G_2(t') \neq G_1(s) \oplus G_2(t)$.

     On en déduit que $G(s,t) = G_1(s) \oplus G_2(t)$.

   On conclut par récurrence.

9. Par récurrence immédiate, la valeur de Grundy d'un jeu de Nim à un seul tas est son nombre de jetons. De plus, un jeu de Nim à plusieurs tas est la somme des jeux de Nim pour chacun des tas, donc la valeur de Grundy est $t_0 \oplus \ldots \oplus t_{n-1}$, ce qui est cohérent avec les questions 5 et 7.

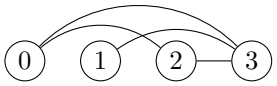
Exercice 2: Graphes de saut

Définition

Un **graphe de saut** de longueur $n \geqslant 1$ est un graphe non-orienté $G = (S, A)$ tel que $S = [\![0, n-1]\!]$ et tel que pour tous $a \leqslant b < c \leqslant d < n$, $\{b, c\} \in A \Rightarrow \{a, d\} \in A$.

Exemple

Voici un graphe de saut de longueur 4.



1. Dessiner tous les graphes de saut de longueur 1, 2 et 3.
2. Proposer un algorithme en pseudo-code qui prend en argument la matrice d'adjacence d'un graphe et détermine si ce graphe est un graphe de saut. Déterminer sa complexité en temps et en espace.

Définition

Un graphe de saut de longueur n est dit **comprimé** si pour tout $i < n-1$, les sommets i et $i+1$ ne sont pas adjacents.

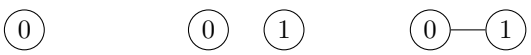
3. Montrer que les graphes de saut comprimés de longueur $n+1$ sont en bijection avec les graphes de saut de longueur n.
4. Soit P_n le nombre de graphes de saut de longueur n. On pose par convention $P_0 = 1$. Déterminer une formule de récurrence pour calculer P_{n+1} en fonction de $P_0, \ldots, P_n$.

Le but des questions suivantes est de considérer s'il existe une permutation des sommets transformant un graphe quelconque en un graphe de saut.

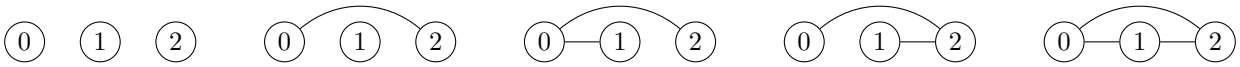
5. Décrire en français un algorithme qui prend en argument un tableau T de longueur n représentant une permutation de 0 à $n-1$ inclus et modifie et renvoie T pour qu'il représente la prochaine permutation suivant un certain ordre total $\prec$ sur les permutations. Si T est la dernière permutation de cet ordre, alors le programme renvoie un tableau vide.
6. Décrire en français un algorithme naïf prenant en argument un entier $n \geqslant 0$ et une matrice d'adjacence M de taille $n \times n$ et décide s'il existe une permutation de $\{0, \ldots, n-1\}$ telle que le graphe résultant est un graphe de saut. Déterminer sa complexité en temps et en espace.
7. Montrer que pour tout graphe $G = (\{0, 1, 2\}, A)$, il existe une permutation de $\{0, 1, 2\}$ telle que le graphe résultant est un graphe de saut.
8. Déterminer un graphe $G = (\{0, 1, 2, 3\}, A)$ dont aucune permutation ne résulte en un graphe de saut.
9. Que dire de l'existence de graphes non connexes tels qu'il existe une permutation des sommets donnant un graphe de saut ?

Corrigé

1. Les graphes de saut de longueur 1 et 2 sont :



Les graphes de saut de longueur 3 sont :



2. On commence par remarquer que la définition est équivalente à : $\forall b < c, \{b, c\} \in E \Rightarrow (b = 0$ ou

$\{b-1,c\} \in E$) et ($c = n-1$ ou $\{b,c+1\} \in E$). Il est clair que la définition initiale implique cette définition. La réciproque se fait par récurrence descendante sur b et ascendante sur c.

Cela permet d'imaginer un algorithme naïf de complexité temporelle quadratique et de complexité spatiale constante :

```
Algorithme : Graphe_de_saut
Entrée : matrice d'adjacence M de taille n fois n
Pour b = 0 à n - 1 Faire
   Pour c = b à n - 1 Faire
      Si M[b][c] = 1
         Si (b > 0 et M[b-1][c] = 0) ou (c < n - 1 et M[b][c+1] = 0)
            Renvoyer Faux
Renvoyer Vrai
```

3. On considère f la fonction qui prend en argument un graphe de saut comprimé $G = (\llbracket 0,n \rrbracket, E)$ de longueur $n+1$ en un graphe de saut $G' = (\llbracket 0, n-1 \rrbracket, E')$ de longueur n de la manière suivante :

$$E' = \{\{a, b-1\}, a < b \text{ et } \{a,b\} \in E\}$$

G étant comprimé, $\{a, b-1\}$ n'est jamais un singleton, donc f est bien définie. Montrons que G' est bien un graphe de saut. Soit $\{b,c\} \in E'$, $b < c$ et (a,d) tels que $a \leqslant b$ et $c \leqslant d$. Alors $\{b, c+1\} \in E$ par définition de E'. G étant un graphe de saut, on en déduit $\{a, d+1\} \in E$, ce qui implique $\{a,d\} \in E'$ et G' est bien un graphe de saut.

En posant g la fonction qui à un graphe de saut $G' = (\llbracket 0, n-1 \rrbracket, E')$ associe $G = (\llbracket 0, n \rrbracket, E)$ en posant :

$$E = \{\{a,b\}, a < b \text{ et } \{a, b-1\} \in E'\}$$

il est clair que $g(G')$ est un graphe de saut comprimé (car $\{a, b-1\} \in E' \Rightarrow a < b-1$) et que g est la réciproque de f.

4. Soit $n \in \mathbb{N}$ et soit $G = (\llbracket 0, n \rrbracket, E)$ un graphe de saut de taille $n+1$. Soit $i \leqslant n$ le plus petit sommet tel que $\{i, i+1\} \in E$, ou $i = n$ si un tel sommet n'existe pas. Dès lors, le sous-graphe de G induit par $\llbracket 0, i \rrbracket$ est un graphe de saut de longueur $i+1$, comprimé par définition de i. Le sous-graphe de G induit par $\llbracket i+1, n \rrbracket$ est un graphe de saut de longueur $n-i$. De plus, toutes les arêtes entre un sommet de $\llbracket 0, i \rrbracket$ et un sommet de $\llbracket i+1, n \rrbracket$ existent car $\{i, i+1\} \in E$ ou le deuxième ensemble est vide. Enfin, par la question précédente, le sous-graphe induit par $\llbracket 0, i \rrbracket$ est en bijection avec un graphe de saut de longueur i (car il est comprimé). On en déduit que le nombre de graphes de saut de longueur $n+1$ est $P_i P_{n-i}$ pour i fixé.

Finalement, la formule de récurrence est $P_{n+1} = \sum_{i=0}^{n} P_i P_{n-i}$.

5. On considère $\prec$ comme l'ordre lexicographique sur les images de $0, \ldots, n-1$. Il est clair que c'est un ordre total. En écrivant les premières permutations pour $n = 4$, essayons de déterminer un algorithme pour passer d'une permutation à la suivante :

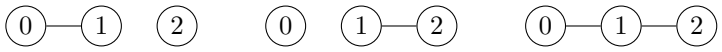
$$0123 \quad 0132 \quad 0213 \quad 0231 \quad 0312 \quad 0321 \quad 1023$$

On remarque que 0 reste en première position pour les 6 premières permutations. Le moment où 0 est remplacé a lieu lorsque les chiffres qui suivent sont par ordre décroissant (sinon il existerait une autre permutation commençant par 0 plus grande selon l'ordre lexicographique). Dès lors, on remplace 0 par le plus petit des chiffres qui suit parmi ceux qui sont plus grands que 0, et on réordonne les chiffres par ordre croissant. On propose alors l'algorithme suivant :

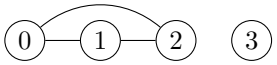
- Poser j le plus grand indice tel que $T[j] < T[j+1]$.
- Si j n'est pas défini, renvoyer le tableau vide.
- Sinon poser $k > j$ le plus grand indice tel que $T[j] < T[k]$ (k est bien défini, car il vaut au pire $j+1$).
- Permuter $T[j]$ et $T[k]$.
- Retourner le sous-tableau $T[j+1 : n]$ (car il était par ordre décroissant) et renvoyer T.

6. On propose alors l'algorithme suivant :
   - Poser T le tableau $[0, 1, \ldots, n-1]$.
   - Tant que T n'est pas le tableau vide, faire les deux instructions suivantes :
   - Si le graphe G auquel on applique la permutation T est un graphe de saut, renvoyer Vrai.
   - Transformer T par l'algorithme précédent.
   - Renvoyer Faux.

   La complexité en temps est $\mathcal{O}(n!n^2)$ et celle en espace est $\mathcal{O}(n)$.
7. Il suffit d'étudier les graphes sur $\{0, 1, 2\}$ qui ne sont pas des graphes de saut, à savoir :



   Dans le premier et le dernier cas, il suffit d'intervertir 1 et 2. Dans le deuxième cas, il suffit d'intervertir 0 et 1.
8. Le seul graphe qui convient est :



   En effet, 3 n'étant lié ni à 0, ni à 1 qui sont liés, le sommet 3 doit apparaître entre les sommets 0 et 1. On raisonne de même pour 1 et 2 et 0 et 2, ce qui est impossible.
9. Si un tel graphe existe, alors ses composantes connexes sauf au plus une doivent être des singleton (sinon chaque sommet d'une C.C. de taille > 1 doit se trouver entre deux sommets reliés d'une autre C.C. de taille > 1 ce qui n'est pas possible). De plus, la C.C. non triviale doit être bipartie : comme il existe une C.C. triviale composée d'un sommet v, pour chaque arête $\{x, y\}$ de la C.C. non triviale, v doit se trouver entre x et y. Cela permet de décomposer la C.C. en deux (les sommets avant et ceux après), et aucune arête ne doit exister entre deux sommets d'un même côté.

Exercice 3: Langages continuables et mots primitifs

On fixe un alphabet Σ de taille au moins 2. Dans le sujet, on considèrera des automates sur Σ qui seront toujours considérés finis, déterministes et complets.

Définition

Soit $u \in \Sigma^* \setminus \{\varepsilon\}$. u est dit **primitif** s'il n'existe pas de mot $v \in \Sigma^*$ et d'entier $p > 1$ tel que $u = v^p$.

Un langage rationnel L est dit **continuable** si et seulement si

$$\forall u \in \Sigma^*, \exists v \in \Sigma^*, uv \in L$$

1. Le mot *abaaabaa* est-il primitif ? Le mot *ababbaabbbababbab* (de longueur 17) est-il primitif ?
2. Proposer un algorithme naïf qui, étant donné un mot, détermine s'il est primitif. Préciser la complexité en temps et en espace.
3. Donner un exemple de langage rationnel infini qui ne contient aucun mot primitif.
4. Donner un exemple de langage rationnel infini qui ne contient que des mots primitifs.
5. Donner un exemple de langage rationnel infini non continuable. Existe-t-il des langages rationnels continuables dont le complémentaire est infini ?
6. Étant donné un automate A, proposer un algorithme qui détermine si $L(A)$ est continuable. Justifier sa correction et préciser la complexité en temps et en espace.
7. Montrer qu'un langage rationnel continuable contient une infinité de mots primitifs.
8. Étant donné un langage rationnel continuable L reconnu par un automate A, donner une borne supérieure de la taille du plus petit mot primitif de L.
9. La réciproque à la question 7 est-elle vraie ?

Corrigé

1. Le premier est non primitif car égal à $(abaa)^2$. Le deuxième est primitif car sa taille est un nombre premier.
2. Pour u de taille n, il s'agit de vérifier, pour chaque préfixe v de taille $\leqslant \frac{n}{2}$, si $u = v^{|u|/|v|}$. Vérifier si u est de cette forme se fait en complexité $\mathcal{O}(n)$, d'où un algorithme de complexité $\mathcal{O}(n^2)$ en temps et $\mathcal{O}(1)$ en espace.
3. Le langage aa^+ convient.
4. Le langage a^*b convient.
5. Le langage a^* convient : le mot b est un contre-exemple. Pour la deuxième partie de la question, Σ^*a convient (car l'alphabet est de taille au moins 2).
6. Dans un premier temps, on supprime tous les états non accessibles, puis on complète l'automate (en rajoutant éventuellement un état puits). Dès lors, il suffit de vérifier que tous les états sont co-accessibles. Ces vérifications se font en temps et espace linéaires (il s'agit d'un parcours de graphe, en partant de l'état initial pour trouver les états accessibles, en partant des états finaux pour trouver les états co-accessibles).

   Pour la preuve, s'il existe un état accessible et non co-accessible q, alors il existe $u \in \Sigma^*$ tel que $\delta^*(q_0, u) = q$ et $\forall v \in \Sigma^*$, $\delta^*(q, v) \notin F$. Cela signifie bien que $L(A)$ est non co-accessible. La réciproque se fait de la même manière.
7. Soit A un automate fini à n états dont $L(A)$ est continuable. On peut supposer que tous les états de A sont accessibles et co-accessibles par la question précédente. Dès lors, $\forall u \in \Sigma^*, \exists v \in \Sigma^*$ tel que $|v| < n$ et $uv \in L(A)$. On pose alors, pour $i \in \mathbb{N}$, $i \geqslant n-1$, $u_i = ab^i$. On pose de plus v_i le plus petit mot tel que $w_i = u_i v_i \in L(A)$. On a $|v_i| < n$, donc w_i est un mot primitif. En effet, si $w_i = x^p$ avec $p \geqslant 2$, alors x (la première occurrence) commence par un a (car u_i commence par un a), mais comme $|u_i| \geqslant n > |v_i|$, on a $|x| < |u_i|$, et donc x (la deuxième occurrence) commence par un b. C'est absurde, donc w_i est un mot primitif. Il existe bien une infinité de tels mots (car $1 + i + n > |w_i| > i$).
8. Avec la construction de la preuve précédente, $2n-1$ semble être une borne supérieur, où n est le nombre d'états de l'automate.
9. La réciproque est fausse, par exemple avec a^*b (tous les mots sont primitifs, mais il n'est pas continuable).