Exercice 1 (Planification):

On note $I = \{1, \dots, n\}$. Un problème de planification est donné par :

- n variables booléennes $\{x_i\}_{i \in I}$;
- m opérations où chaque opération est définie par une condition de la forme $\bigwedge_{i \in I'} x_i = \alpha_i$ avec $I' \subseteq I$ et une mise à jour de la forme $\{x_i \leftarrow \beta_i\}_{i \in I''}$ avec $I'' \subseteq I$.
  Voici un exemple d'opération : Si $x_1 = \mathtt{V} \wedge x_3 = \mathtt{F}$ Alors $x_1 \leftarrow \mathtt{F}; x_2 \leftarrow \mathtt{V}$
- Une configuration initiale s_{init} et une configuration finale s_{fin} où une configuration est une valuation des variables.
- Une opération est applicable à une configuration si la condition de l'opération s'évalue à `V`. Son application consiste à effectuer ses mises à jour pour obtenir la nouvelle configuration. Par exemple l'opération précédente est applicable à la configuration $(\mathtt{V}, \mathtt{F}, \mathtt{F})$ et conduit à la configuration $(\mathtt{F}, \mathtt{V}, \mathtt{F})$.

Le problème consiste à déterminer s'il existe une suite d'applications des opérations (avec éventuellement plusieurs applications d'une même opération) qui conduise de la configuration initiale à la configuration finale.

1. Montrez que le problème de planification est dans PSPACE.

Solution:

Grâce au théorème de Savitch, il suffit de montrer que le problème est dans NPSPACE :

Algorithm 1: Planification

```
1 x ← s_init ;                                  // x est un tableau de taille n
2 c ← 0 ;                                       // il y a 2^n configurations possibles
3 while x ≠ s_fin ∧ c < 2^n do
      // non déterministe :
4     Choisir une opération op parmi les m opérations du problème;
5     Mettre à jour x en simulant op;
6     c ← c + 1;
7 end
8 if c ≤ 2^n then accept;
9 else reject;
```

Cet algorithme fonctionne bien en espace polynomial car il utilise uniquement un tableau x de taille n, à valeurs dans $\{0, 1\}$.

2. Montrez que le problème de planification est PSPACE-difficile. Pour cela, on montrera une réduction LOGSPACE du problème ci-dessous vers le problème de planification :
   Donnée : Une machine $\mathcal{M}$ calculant en espace n^k et un mot w.
   Question : $\mathcal{M}$ accepte w.
   (Pour simplifier, on supposera que $\mathcal{M}$ a pour alphabet $\{0, 1\}$ et n'a qu'une seule bande de travail, qui sert aussi de bande de sortie)

Solution:

À partir de $(\mathcal{M}, w)$, on construit une instance $\mathcal{P}$ du problème de planification de la manière suivante (avec $|w| = n$) :

- n variables $\mathtt{w}_i$ codent l'entrée w. Ces variables ne seront jamais modifiées par les opérations de $\mathcal{P}$.
- n autres variables $\mathtt{pw}_i$ codent la position de la tête de lecture sur la bande d'entrée : si la tête e lecture est sur w_i, alors on s'assurera que $\mathtt{pw}_i = \mathtt{V}$ et $\mathtt{pw}_j = \mathtt{F}$ pour $j \neq i$.

- La bande de travail étant un mot x de taille n^k, pour peut la coder de la même manière avec n^k variables $\mathtt{x}_i$ codant sa valeur, et n^k variables $\mathtt{px}_i$ codant la position de la tête de lecture sur cette bande (les opérations seront telles qu'à tout instant, exactement une variable $\mathtt{px}_i$ est vraie).
- Une variable pour chaque état de $\mathcal{M}$ (là encore, seule la variable correspondant à l'état s est vraie lorsque la configuration courante est en l'état s).
- Les opérations de $\mathcal{P}$ sont exactement celles nécessaires pour simuler une transition de la machine $\mathcal{M}$.
- s_{init} est le codage de la configuration initiale de $\mathcal{M}$ sur w.
- s_{fin} est le codage de la configuration finale acceptante de $\mathcal{M}$.

Cette construction s'effectue bien en LOGSPACE, et on a que $\mathcal{M}$ accepte w si et seulement si $\mathcal{P}$ possède un run valide.

Exercice 2 (Vacuité d'un Automate Déterministe Concurrent):

Un automate déterministe concurrent $\mathcal{A}$ est donné par un ensemble d'automates déterministes $\{\mathcal{A}_i\}_{i\leq n}$ appelés composants. Un état de l'automate déterministe concurrent est un tuple $(s_1, \ldots, s_n)$ composé d'un état par composant. Lorsqu'une lettre a est lue, les automates $\mathcal{A}_i$ qui ont une transition issue de s_i étiquetée par la lettre effectuent simultanément leur transition tandis que les autres conservent leur état. Pour qu'une lettre puisse être lue, au moins un automate doit effectuer une transition. Un mot est reconnu s'il conduit à un tuple d'états terminaux. Le problème de la vacuité consiste à savoir s'il existe au moins un mot accepté par l'automate.

1. Montrez que le problème de la vacuité des automates déterministes concurrents est dans PSPACE.

Solution:

On pourra concevoir un algorithme non déterministe et appliquer le Grâce au théorème de Savitch, il suffit de montrer que le problème est dans NPSPACE :

Algorithm 2: Planification

```
   // s est un tableau de taille n, représentant les états courants
 1 for i = 1 to n do
 2 |  s[i] ← s^i_init;
 3 end
 4 c ← 0 ;     /* Il y a m^n configurations, où m est le nombre maximal
     d'états d'un des automates */
 5 while s ≠ [s_fin] ∧ c < m^n do
   |  // non déterministe :
 6 |  Choisir une lettre a ∈ Σ;
 7 |  Mettre à jour s en simulant chaque automate sur la lettre a;
 8 |  c ← c + 1;
 9 end
10 if c ≤ m^n then accept;
11 else reject;
```

Cet algorithme fonctionne bien en espace polynomial car il utilise uniquement un tableau s de taille n, à valeurs dans Σ.

2. Montrez que le problème de la vacuité est PSPACE-difficile.

Solution:
On effectue une réduction du problème de planification. Pour une instance $\mathcal{P}$ du problème de planification, on construit l'automate déterministe concurrent $\mathcal{A}$ suivant :

- L'alphabet correspond à l'ensemble des opérations de $\mathcal{P}$.
- Pour chaque variable x_i, on construit un automate à 2 états $\{\mathtt{V}, \mathtt{F}\}$, dont l'état initial correspond à la valeur de x_i dans s_{init}, l'état final correspond à la valeur de x_i dans s_{fin}, et les transitions correspondent à la mise à jour de x_i pour chaque opération.

Ainsi, on peut passer de s_{init} à s_{fin} dans $\mathcal{P}$ si et seulement si $\mathcal{A}$ accepte au moins un mot (les mots acceptés par $\mathcal{A}$ sont exactement aux exécutions valides de $\mathcal{P}$).

Exercice 3 (Chemin dans un Graphe Pondéré à Poids Relatifs):
On considère des graphes pondérés de la forme $G = (V, E, p)$ où les arêtes de $E \subseteq V \times V$ sont orientées et portent chacune un poids, un entier *relatifs* donné par $p : E \to \mathbb{Z}$.

On commence par définir ou rappeler quelques notions et notations qui seront utiles dans la suite de l'énoncé et dans vos solutions : Pour une arête $e = (u, v) \in E$, on note ${}^\bullet e$ pour u et $e^\bullet$ pour v. Un *chemin de longueur ℓ dans G* est un mot $\rho = e_1 \cdots e_\ell \in E^+$ composé de $\ell > 0$ arêtes de E et tel que ${e_{i-1}}^\bullet = {}^\bullet e_i$ pour tout $i = 2, \ldots, \ell$. Si $\rho = e_1 \cdots e_\ell$ est un chemin, les notations ${}^\bullet\rho$ et $\rho^\bullet$ désignent ${}^\bullet e_1$ et ${e_\ell}^\bullet$ respectivement. Le poids $p(\rho)$ d'un chemin est la somme $\sum_{i=1}^{\ell} p(e_i)$ des poids de ses arêtes.
Un chemin $\rho = e_1 \ldots e_\ell \in E^+$ est un *cycle* si ${e_l}^\bullet = {}^\bullet e_1$ et le cycle est *élémentaire* si les sommets ${}^\bullet e_1, \ldots, {}^\bullet e_\ell$ sont tous distincts.
Le problème WEIGHTEDPATH a comme input un graphe pondéré G, deux sommets $u, v \in E$, un poids $a \in \mathbb{N}$. Il s'agit de décider s'il existe dans G un chemin allant de u à v et de poids total a.

1. On dit qu'un chemin ρ est *factorisé en cycles* si ρ est écrit sous la forme

$$\rho = \rho_0 \sigma_1^{k_1} \rho_1 \sigma_2^{k_2} \cdots \rho_{r-1} \sigma_r^{k_r} \rho_r$$

telle que les facteurs $\sigma_1, \ldots, \sigma_r$ sont des cycles élémentaires, les entiers $k_1, \ldots, k_r$ sont non nuls et les facteurs $\rho_0, \ldots, \rho_r$ n'ont aucun facteur qui soit un cycle. (La notation w^k avec $k \in \mathbb{N}$ dénote la concaténation de k copies de w, avec $w^0 = \epsilon$ et $w^{k+1} = w^k \cdot w$).
Montrez que tout chemin admet une factorisation en cycles.

Solution:
Par induction sur $|\rho|$.

- Si $\rho = (u, u)$ alors on prend $r = 1$, $\sigma_1 = \rho$, $k_1 = 1$ et $\rho_0 = \rho_1 = \epsilon$ en notant que la définition de factorisation en cycle (de "FC") dit que les ρ_i sont des facteurs de ρ, pas forcément des chemins, et donc peuvent être vides.
- Si $\rho = (u, v)$ avec $u \neq v$ on prend $r = 0$ et $\rho_0 = \rho$.
- Si $\rho = \rho' \cdot (u, v)$ alors on prend une FC de ρ', sous la forme $\rho' = \rho_0 \cdot \prod_{i=1}^{r} (\sigma_i^{k_i} \rho_i)$, qui existe par hyp. ind. On considère alors $\rho_r \cdot e$. Si ce suffixe de ρ ne contient pas de cycle, on obtient une FC de ρ en remplaçant ρ_r par $\rho_r \cdot e$ dans la FC de ρ'. Si $\rho_r \cdot e$ contient un cycle alors, puisque ρ_r n'en contient pas, c'est que $e^\bullet$ coïncide avec un sommet visité par ρ_r. On écrit $\rho_r = \rho'_r \cdot \rho''_r$ tel que ${\rho'_r}^\bullet = e^\bullet$. On en tire une FC de $\rho_r \cdot e$ via $\rho_r \cdot e = \rho'_r (\rho''_r \cdot e)^1 \epsilon$. En remplaçant ρ_r par cette FC dans la FC de ρ' on obtient une FC de ρ.

2. Montrez que si G admet un chemin de poids a allant de s à t alors il existe en particulier un tel chemin avec une factorisation en cycles $\rho_0\sigma_1^{k_1}\rho_1\sigma_2^{k_2}\cdots\rho_{r-1}\sigma_r^{k_r}\rho_r$ telle que les σ_i's aient tous des poids $p(\sigma_i)$ différents.

Solution:
Par induction sur $|\rho|$. Le cas $|\rho| = 1$ est trivial (comme à la question 1).
Si $|\rho| > 1$ on l'écrit $\rho = \rho' \cdot e$ avec $e \in E$: par hypothèse d'induction il existe un chemin ρ_{ind} de même poids que ρ' et admettant une factorisation en cycle de poids distincts (une FCPD) $\rho_{ind} = \rho_0 \cdot \prod_{i=1}^{r}(\sigma_i^{k_i}\rho_i)$. Notons que $\rho_{ind}\cdot e$ est un chemin de s à t de même poids que $\rho' \cdot e = \rho$ et qu'il nous suffit de montrer l'existence d'une FCPD pour $\rho_{ind} \cdot e$.
Si $\rho_r \cdot e$ est sans cycle on obtient une FCPD de $\rho_{ind}\cdot e$ en remplaçant ρ_r par $\rho_r \cdot e$ dans la FCPD de ρ_{ind}. Si $\rho_r \cdot e$ contient un cycle alors comme ρ_r est sans cycle on sait que, comme à la question précédente, $\rho_r = \rho'_r \cdot \sigma_{r+1}$ avec ρ'_r sans cycle. On a alors deux cas :

(a) Si pour tout $i = \{1, \ldots, r\}$ le poids de σ_i est différent du poids de σ_{r+1} alors $\rho_0 \cdot \prod_{i=1}^{r-1}(\sigma_i^{k_i}\rho_i) \cdot \sigma_r^{k_r} \cdot \rho'_r \cdot \sigma_{r+1}^{1} \cdot \epsilon$ est une FCPD de $\rho_{ind} \cdot e$ (on a supposé $r > 0$).

(b) Sinon il existe un $0 < j \leq r$ tel que σ_j et σ_{r+1} soient de même poids. Dans ce cas, et en supposant $j < r$ pour simplifier l'écriture,

$$\rho_0 \cdot \Big(\prod_{i=1}^{j-1}\sigma_i^{k_i}\rho_i\Big) \cdot \sigma_j^{1+k_j} \cdot \rho_j \cdot \Big(\prod_{i=j+1}^{r-1}\sigma_i^{k_i}\rho_i\Big) \cdot \sigma_r^{k_r} \cdot \rho'_r$$

est la FCPD d'un chemin de s à t de même poids que $\rho' \cdot e$.
(Par rapport au chemin ρ, ce chemin prend le cycle σ_j une fois de plus, et ne prend pas le dernier cycle σ_{r+1})

3. Pour $G = (V, E, p)$ avec $p : E \to \mathbb{Z}$, on notera k le nombre $|V|$ de sommets, m le nombre $|E|$ d'arêtes, et $P = \max_{e\in E}|p(e)|$ le plus grand poids (en valeur absolue). Ainsi la donnée du graphe utilise un espace mémoire en $O(k + m\lceil\log_2(P)\rceil)$.
Donnez un polynôme à quatre variables $Q(x_1, x_2, x_3, x_4)$ tel que pour toute instance $\langle G, u, v, a\rangle$, si G a un chemin de poids a reliant u à v alors il existe un tel chemin de longueur bornée par $Q(k, m, P, a)$.

Solution:
Les résultats sur les FCPD restent valides quand les poids sont des relatifs. On sait donc que s'il existe un chemin de s à t de poids a il en existe un admettant une FCPD $\rho_0 \cdot \prod_{i=1}^{r}(\sigma_i^{k_i} \cdot \rho_i)$. On considère un chemin ρ et une factorisation qui minimisent $\sum_{i=1}^{r} k_i$.
Notons qu'un circuit élémentaire σ a au plus k arêtes et donc $-kP \leq p(\sigma) \leq kP$. On notera P' pour kP et on sait donc que $r \leq 2P' + 1$ puisque les cycles sont de poids distincts.
Étape 1. Pour ρ et sa FCPD $\rho_0 \cdot \prod_{i=1}^{r}(\sigma_i^{k_i} \cdot \rho_i)$, soit $I(\rho)$ l'ensemble (éventuellement vide) des indices i tels que σ_i soit de poids (strictement) positif, et $J(\rho)$ celui des indices i tels que $p(\sigma_i) < 0$. Montrons que l'on peut supposer que :

$$(\forall i \in I(\rho), k_i \leq P') \vee (\forall i \in J(\rho), k_i \leq P')$$

En effet, s'il existe un cycle négatif σ_i de poids p^- avec $k_i > P'$ et un cycle σ_j de poids positif p^+ avec $k_j > P'$ alors on peut remplacer k_i par $k_i - p^+$ et k_j par $k_j + p^-$

dans la FCPD sans changer le poids total tout en diminuant $\sum k_i$ qui était supposé minimal.

Étape 2. Supposons grâce à l'étape précédente que $\forall i \in J(\rho), k_i \leq P'$ (le cas $\forall i \in I(\rho), k_i \leq P'$ est similaire). On sait donc que :

$$\sum_{i \in I(\rho)} k_i.p(\sigma_i) = a - \left(\sum_{0 \leq i \leq r} p(\rho_i) \right) - \left(\sum_{i \in J(\rho)} k_i.p(\sigma_i) \right) .$$

Donc, puisque $p(\rho_i) \geq -P'$ tout comme $p(\sigma_i)$, et comme $k_i \leq P'$ quand $i \in J(\rho)$:

$$\sum_{i \in I(\rho)} k_i.p(\sigma_i) \leq a + \left(\sum_{0 \leq i \leq r} P' \right) + \left(\sum_{i \in J(\rho)} P'.P' \right) .$$

D'où

$$\sum_{i \in I(\rho)} k_i.p(\sigma_i) \leq a + (2P' + 1)P' + P'^3 .$$

On en déduit que $k_i \leq a + (2P' + 1)P' + P'^3$ pour tout $i \in I(\rho)$. Cette borne s'applique aussi quand $i \notin I(\rho)$ puisque $k_i \leq P'$ quand $i \in J(\rho)$, et puisque on peut supposer $k_i = 1$ s'il y a un cycle σ_i de poids nul. La longueur du chemin est donc bornée par $r.k.Q'(a, k, P) + (r + 1)k$.

4. Montrer que le problème WEIGHTEDPATH est NP-complet.

Solution:

Le problème est NP–complet. Il est évidemment NP–difficile puisqu'il l'est déjà quand les poids sont tous positifs. Pour montrer qu'il est dans NP, il suffit de donner un algorithme non déterministe en temps polynômial. Comme pour le cas positif, on devine un vecteur v de nombres d'occurrences des arcs $e \in V$ tel que $\sum_{e \in E} v[e] \leq Q(k, m, P, a)$, c.-à-d. qu'il suffit de deviner un vecteur d'une taille bornée par un polynôme fixé de n, la taille de l'instance.

Attention, si k et m sont bornés par n, les valeurs P et a peuvent quant à elles être exponentielles : c'est la taille de leur représentations qui est bornée par n. Donc les valeurs de v ne sont pas bornées polynomialement en n mais la taille d'une représentation de v est en $O(n^2)$ puisque $v[e] \leq Q(k, m, P, a)$ pour chaque $e \in E$.

On vérifie alors que v est bien l'image de Parikh d'un chemin de u à v grâce aux conditions d'Euler. On vérifie aussi que $\sum_{e \in E} v[e].p(e) = a$. Ces vérifications impliquent des calculs arithmétiques simples sur un nombre polynomial de valeurs qui s'écrivent toutes avec un nombre polynomial de chiffres, elles sont donc réalisables en temps polynomial.