Exercice 1 (Grammaires contextuelles) : On admet le théorème suivant :

Théorème : lemme de l'étoile pour les langages algébriques

Soit L un langage algébrique (ou hors-contexte). Alors il existe $n \in \mathbb{N}$ tel que pour tout mot $u \in L$ vérifiant $|u| \geq n$, on peut écrire $u = vwxyz$ avec :

1. $|wxy| \leq n$;
2. $|wy| > 0$;
3. $\forall k \in \mathbb{N}, vw^k xy^k z \in L$.

On pose $L_0 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ et $L_1 = \{uu \mid u \in \{a,b\}^*\}$.

1. Montrer que L_0 n'est pas algébrique. L_1 est-il algébrique ?

Solution :

Supposons que L_0 soit algébrique et soit n la longueur de pompage donnée par le lemme de pompage algébrique. On pose $u = a^n b^n c^n \in L$. D'après le lemme, u admet une décomposition $u = vwxyz$ vérifiant les trois conditions. Mais si u vérifie les deux premières conditions, alors wy est un mot non vide qui est soit dans a^*b^*, soit dans b^*c^*. Dans les deux cas, vxz ne contient pas autant de a que de b que de c, ce qui contredit la troisième condition. On conclut par l'absurde que L n'est pas algébrique.

Pour L_1, on pose $u = a^n b^n a^n b^n$. On distingue :

- si $wy \in a^* \mid b^*$, alors comme $|wxy| \leq n$, w et w sont contenus dans la même des quatre portions du mot. Dans vw^2xy^2z, il y a un déséquilibre entre deux sections correspondant à la même lettre, et le mot ne peut donc pas s'écrire comme $u'u'$.
- si wxy est à cheval sur deux portions, alors on a le même type de déséquilibre dans vxz.

Définition

Une **grammaire** (non restreinte) est un quadruplet $G = (V, T, P, S)$ où V est un alphabet dit de **variables**, T est un alphabet disjoint de V dit de **terminaux**, $S \in V$ est le **symbole initial** et P est un ensemble de **règles de production** de la forme $\alpha \to \beta$, avec $\alpha \in (V \cup T)^+$ et $\beta \in (V \cup T)^*$.

Pour $u, v \in (V \cup T)^*$, on dit que u se **dérive immédiatement** en v, noté $u \Rightarrow v$, s'il existe $\alpha \to \beta \in P$, $\gamma, \delta \in (V \cup T)^*$ tels que $u = \gamma\alpha\delta$ et $v = \gamma\beta\delta$. On note $\Rightarrow^*$ la clôture réflexive et transitive de $\Rightarrow$. Le **langage engendré par** G, noté $\mathcal{L}(G)$, est $\mathcal{L}(G) = \{u \in T^* \mid S \Rightarrow^* u\}$.

Une grammaire est dite **monotone** si toute règle est de la forme $S \to \varepsilon$ ou $\alpha \to \beta$, $|\alpha| \leq |\beta|$, $\beta \in (V \cup T \setminus \{S\})^+$. Un langage est dit **monotone** s'il est engendré par une grammaire monotone.

On notera par des lettres capitales les éléments de V et par des minuscules les éléments de T.

2. Déterminer le langage engendré par $S \to aSBa \mid aba$, $aB \to Ba$, $bB \to bb$. Ce langage est-il monotone ?

Solution :

Le langage engendré est $\{a^n b^n a^n \mid n \in \mathbb{N}^*\}$. L'idée est que si $S \Rightarrow^* \alpha$ et α contient un S, alors $\alpha = a^n S \beta$, avec $\beta \in \{a, B\}^*$, $|\beta|_a = |\beta|_B = n$. Lors de la dérivation immédiate qui fait disparaître S, on obtient un mot de la forme $a^{n+1}aba\beta$. Dès lors, les variables B ne peuvent disparaître qu'au "contact" d'un b, donc il y a inversion de tous les couples aB en Ba pour faire revenir les B "au centre" et les transformer en b. La grammaire donnée n'est pas monotone, car S apparaît du côté droit d'une règle, mais on peut facilement la transformer sans changer le langage, en supprimant cette règle et en rajoutant $S \to aS'Ba$ et $S' \to aS'Ba \mid aba$. La nouvelle grammaire est monotone.

3. Montrer que L_0 est monotone. En étudiant la grammaire donnée par $S \to S'F$, $S' \to aS'A \mid bS'B \mid F$, $F \to a$, $xX \to Xx$ (pour $x \in \{a, b\}$, $X \in \{A, B\}$), montrer que L_1 est monotone.

Solution :
On propose pour L_0 :

$$\begin{aligned} S &\to aS'Bc \mid \varepsilon \\ S' &\to aS'Bc \mid abc \\ cB &\to Bc \\ bB &\to bb \end{aligned}$$

C'est la même idée que pour le langage précédent.
Pour L_1, la grammaire donnée par l'énoncé engendre le langage $L_a = \{uaua \mid u \in \{a,b\}^*\}$. Il suffit alors de remarquer que $L_1 = L_a \cup L_b \cup \{\varepsilon\}$ et que les langages monotones sont stables par union (il suffit de faire l'union des alphabets, des règles, d'ajouter un nouveau symbole de départ, et une règle $S \to S_1 \mid S_2$).

4. Montrer que tout langage algébrique est monotone.

Solution :
Soit L un langage engendré par une grammaire algébrique $G = (V, T, P, S)$. On commence par créer un nouveau symbole de départ S' et à rajouter une règle $S' \to S$, pour faire disparaitre le symbole de départ du côté droit de chaque règle. Dès lors, toutes les règles sont monotones, sauf celles de la forme $X \to \varepsilon$. On appelle variable annulable une variable telle que $X \Rightarrow^* \varepsilon$. On peut calculer l'ensemble des variables annulables récursivement (X est annulable si $X \to \varepsilon$ ou $X \to X_1 \dots X_n$ et tous les X_i sont annulables). Dès lors, pour X annulable et $Y \to \alpha X \beta$ une règle telle que $\alpha\beta \neq \varepsilon$, on ajoute la règle $Y \to \alpha\beta$, puis on supprime $X \to \varepsilon$ (si elle existe). La grammaire obtenue est bien monotone.

Définition

Une grammaire $G = (V, T, P, S)$ est dite **contextuelle** si toute règle est de la forme $S \to \varepsilon$ ou $\alpha X \gamma \to \alpha\beta\gamma$, $X \in V$, $\alpha, \gamma \in (V \cup T)^*$, $\beta \in (V \cup T \setminus \{S\})^+$.

Un langage est dit **contextuel** s'il est engendré par une grammaire contextuelle.

5. Montrer que L_0 est contextuel.
On cherchera un moyen de remplacer une règle $XY \to YX$ par plusieurs règles contextuelles.

Solution :
On repart de la grammaire précédente : on remplace c dans les règles par C, on ajoute une règle $C \to c$, et il suffit alors simplement de remplacer la règle $CB \to BC$. On peut la supprimer et rajouter $CB \to CX$, $CX \to YX$, $YX \to YC$, $YC \to BC$. Ces règles sont bien contextuelles.

Définition

Une grammaire monotone est en **forme normale de Kuroda** si ses règles sont de l'une des formes suivantes : $S \to \varepsilon$, $X \to a$, $X \to Y$, $X \to YZ$, $XY \to ZU$.

6. Montrer que tout langage monotone est engendré par une grammaire en forme normale de Kuroda.

Solution :
On commence par remplacer chaque terminal a du côté droit d'une règle par X_a, et on rajoute des règles $X_a \to a$ pour $a \in T$. Après cette transformation, les règles restantes sont de la forme $X_1 \dots X_n \to Y_1 \dots Y_m$, avec $n \leq m$. On distingue :

- si $m \leq 2$, la règle est autorisée par la forme normale de Kuroda ;
- si $n = 1$ et $m > 2$, on remplace $X_1 \to Y_1 \dots Y_m$ par $X_1 \to Y_1 Z_1$, $Z_1 \to Y_2 Z_2$, …, $Z_{m-2} \to Y_{n-1} Y_n$;

- si $n \geq 2$ et $m > 2$, on remplace $X_1 \dots X_n \to Y_1 \dots Y_m$ par $X_1X_2 \to Y_1Z_1$, $Z_1X_3 \to Y_2Z_2$, …, $Z_{n-2}X_n \to Y_{n-1}Z_{n-1}$, $Z_{n-1} \to Y_n \dots Y_m$, et on remplace cette dernière règle comme à l'étape précédente.

La grammaire obtenue est bien en forme normale de Kuroda.

7. En déduire que les langages monotones sont exactement les langages contextuels.

Solution :

On remarque qu'une grammaire contextuelle est monotone. Réciproquement, si un langage est monotone, par la question précédente, il est engendré par une grammaire en forme normale de Kuroda. Les seules règles non contextuelles sont de la forme $XY \to ZU$. On peut les remplacer par $XY \to XU'$, $XU' \to Z'U'$, $Z'U' \to Z'U$, $Z'U \to ZU$. Ces règles sont bien contextuelles.

8. On admet que pour $G = (V, T, P, S)$ une grammaire non restreinte et $u \in T^*$, le problème qui consiste à déterminer si $u \in \mathcal{L}(G)$ est indécidable. Qu'en est-il si on impose à G d'être monotone ?

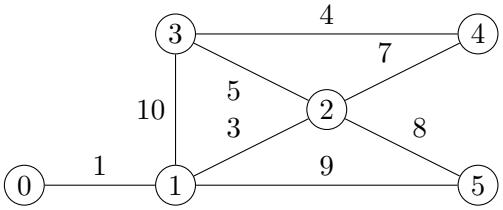
Solution :

Il suffit de remarquer que dans une dérivation de S en u, tous les mots intermédiaires sont de taille $\leq$ à celle de u. Par ailleurs, pour $\alpha, \beta \in (V \cup T)^*$, on peut vérifier en temps polynomial si $\alpha \Rightarrow \beta$. On peut alors tester si $u \in \mathcal{L}(G)$ en créant un graphe orienté dont les sommets sont les mots de taille $\leq |u|$, tel qu'il existe une arête entre α et β si $\alpha \Rightarrow \beta$. Il suffit de tester l'existence d'un chemin de S à u dans ce graphe. Le problème est donc décidable.

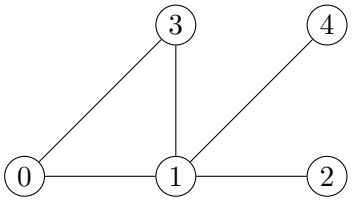
Exercice 2 (Problème de l'arbre de Steiner) : On définit le problème d'optimisation suivant :

Arbre de Steiner	
Entrée :	un graphe pondéré non orienté connexe $G = (S, A, \omega)$, un sous-ensemble $X \subset S$ de sommets dits **terminaux**.
Sortie :	un arbre $T = (S', A')$ tel que $X \subset S' \subset S$.
Optimisation :	minimiser le poids de l'arbre $\omega(T)$.

On considère les graphes suivants :



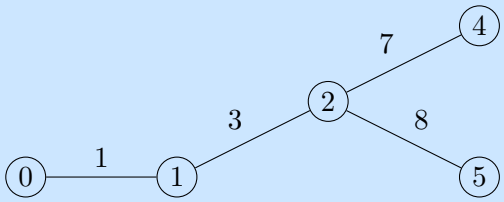
Graphe G_0



Graphe G_1

1. Résoudre le problème Arbre de Steiner pour le graphe G_0 et $X = \{0, 4, 5\}$.

Solution :
On obtient l'arbre :



2. Déterminer un algorithme résolvant le problème Arbre de Steiner. Déterminer sa complexité temporelle et spatiale.

Solution :
On propose l'algorithme suivant :

- Pour chaque sous-ensemble $S' \subset S$:
  - vérifier que $X \subset S'$ (ou passer à l'itération suivante) ;
  - vérifier que $G' = (S', \mathcal{P}_2(S') \cap A)$ est connexe (ou passer à l'itération suivante) ;
  - calculer un arbre couvrant de poids minimal de G' ;
- Renvoyer l'arbre de poids minimal parmi tous ceux calculés.

Pour $|S| = n$, il y a 2^n sous-ensembles S'. La vérification que $X \subset S'$ peut se faire en $\mathcal{O}(n)$, la vérification de connexité en $\mathcal{O}(n+|A|)$, le calcul d'un arbre couvrant de poids minimal en $\mathcal{O}(|A| \log n)$. On obtient donc une complexité temporelle totale en $\mathcal{O}(n^2 \log n 2^n)$ (car $|A| = \mathcal{O}(n^2)$). La complexité spatiale est en $\mathcal{O}(|S| + |A|)$: un sous-ensemble a une taille linéaire (avec un tableau de booléens par exemple), et il faut construire les sous-graphes.

On s'intéresse maintenant à une version décisionnelle du problème de l'arbre de Steiner :

Arbre de Steiner (déci)	
Entrée :	un graphe non orienté connexe $G = (S, A)$, $X \subset S$ et $k \in \mathbb{N}$.
Question :	existe-t-il un arbre $T = (S', A')$ de G tel que $X \subset S' \subset S$, $A' \subset A$ et $\lvert A' \rvert \leq k$?

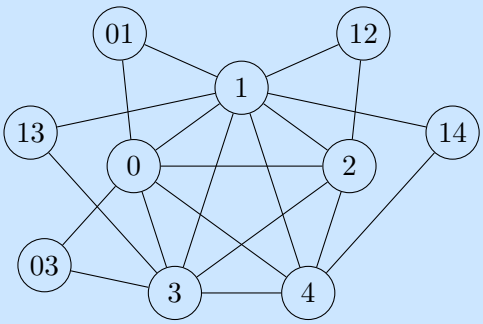
On admet que le problème suivant Couverture par sommets est NP-complet :

Couverture par sommets	
Entrée :	un graphe non orienté $G = (S, A)$ et $k \in \mathbb{N}$.
Question :	existe-t-il une couverture des arêtes de G par les sommets de cardinal $\leq k$, c'est-à-dire un ensemble $C \subset S$ tel que $\lvert C\rvert \leq k$ et $\forall a \in A$, $a \cap C \neq \emptyset$?

Pour un graphe non orienté $G = (S, A)$, on appelle **graphe d'incidence** de G le graphe $G_I = (S_I, A_I)$ tel que $S_I = S \cup A$ et $A_I = \mathcal{P}_2(S) \cup \{\{s, a\} \mid s \in S, a \in A, s \in a\}$.

3. Quel est le graphe d'incidence de G_1 ?

Solution :
On obtient le graphe :



4. Montrer que pour un graphe $G = (S, A)$, G possède une couverture par sommets de taille k si et seulement si G_I possède un arbre de Steiner à $\lvert A\rvert + k - 1$ arêtes ayant A comme sommets terminaux.

Solution :
On procède par double implication.

- Soit C une couverture par sommets de G de taille k. Alors pour tout $a = \{s, t\} \in A$, $s \in C$ ou $t \in C$. On en déduit que dans G_I, chaque élément de A est adjacent à un élément de C. Sachant que tous les éléments de S sont adjacents dans G_I, le sous-graphe induit par $C \cup A$ est connexe dans G_I et d'ordre $\lvert A\rvert + k$. Il existe donc un arbre couvrant de ce sous-graphe. Cet arbre est bien un arbre de Steiner couvrant A et avec $\lvert A\rvert + k - 1$ arêtes.
- Soit $T = (S', A')$ un arbre de Steiner couvrant A dans G_I, avec $\lvert A'\rvert = \lvert A\rvert + k - 1$. Comme c'est un arbre, il possède $\lvert A\rvert + k$ sommets, soit k sommets appartenant à S en particulier. Posons $C = S' \setminus A$. Alors pour $a \in A$, a possède un voisin x dans T. Comme les éléments de A ne sont pas adjacents dans G_I, on en déduit que $x \in C$. Par définition du graphe d'incidence, cela signifie que $x \in a$. Ainsi, C est bien une couverture par les sommets de G de cardinal k.

5. En déduire la NP-complétude de Arbre de Steiner (déci).

Solution :
La construction de G_I peut bien se faire en temps polynomial en la taille de G. On en déduit que Couverture par sommets $\leq_{\mathsf{P}}$ `Arbre de Steiner (déci)`.
Par ailleurs, Arbre de Steiner (déci) $\in$ NP. En effet, un certificat est un tel arbre de Steiner, et on peut vérifier en temps polynomial que c'est un arbre, qu'il couvre X, et qu'il contient moins de k arêtes. On en déduit que Arbre de Steiner est NP-complet.

Pour $G = (S, A, \omega)$ un graphe pondéré non orienté connexe, ω à valeurs positives, et $X \subset S$, on considère l'algorithme suivant :

- construire $K = (X, \mathcal{P}_2(X), d)$ un graphe pondéré complet, où d représente la distance dans G ;
- construire $B = (X, A_B, d)$ un arbre couvrant de poids minimal de K ;
- construire $H = (S', A_H, \omega)$ à partir de B, en remplaçant chaque arête $\{s, t\}$ dans B par un chemin de poids minimal dans G (quitte à rajouter les sommets intermédiaires) ;
- construire $T = (S', A', \omega)$ un arbre couvrant de poids minimal de H.

6. Montrer que l'algorithme précédent renvoie un arbre de Steiner couvrant X (pas nécessairement de poids minimal) et déterminer sa complexité temporelle.

Solution :

On remarque que $X \subset S'$ (car on ne fait que rajouter des sommets à B en construisant H). De plus, H est connexe, car B l'est, et que chaque étape de construction de H ne perd pas la connexité (car G était connexe). Enfin, T est bien un arbre. On en déduit que T est bien un arbre de Steiner couvrant X.

Pour la complexité temporelle, pour $|S| = n$, $|X| = m$, $|A| = p$:

- la construction de K se fait en $\mathcal{O}(n^3)$ ou $\mathcal{O}(mp \log n)$ (il faut calculer toutes les distances entre les sommets de X, avec l'algorithme de Floyd-Warshall ou Dijkstra par exemple) ;
- le calcul de B se fait en $\mathcal{O}(m^2 \log m)$ (avec l'algorithme de Kruskal par exemple) ;
- la construction de H se fait en $\mathcal{O}(m^2 n)$: on peut réutiliser les chemins construits à l'étape 1 (il y a de l'ordre de m^2 chemins, de longueur de l'ordre de n) ;
- la construction de T se fait en $\mathcal{O}(p \log n)$ (à nouveau avec Kruskal).

La complexité totale est en $\mathcal{O}(n^3)$.

7. Montrer que l'algorithme précédent est une 2-approximation de Arbre de Steiner.

Solution :

On commence par remarquer que $\omega(T) \leq d(B)$, car on a remplacé les arêtes de B par des chemins de même poids, puis supprimé des arêtes. Dès lors, soit T^* un arbre de Steiner couvrant X de poids minimal. Un parcours en profondeur cyclique de T^* passant deux fois par chaque arête a un poids égal à $2f(T^*)$. Par ailleurs, la somme des poids des arêtes reliant deux sommets terminaux consécutifs dans ce parcours est supérieure ou égale à la distance entre ces sommets. On en déduit que $2f(T^*)$ est supérieur ou égal au poids d'un cycle hamiltonien dans K parcourant les sommets terminaux dans le même ordre. De plus, en supprimant une arête de ce cycle hamiltonien, on obtient un arbre couvrant de K. On en déduit $2f(T^*) \geq d(B) \geq \omega(T)$. L'algorithme est bien une 2-approximation.

Exercice 3 (Décidabilité de l'arithmétique de Presburger) : On considère dans cet exercice la logique de Presburger, c'est-à-dire la logique du premier ordre sur le langage $\{0, 1, +, =\}$. Formellement, un **terme** est défini par induction comme une constante (0 ou 1), une variable (x, y, x_1, x_2, ...) ou une addition ($+$) entre deux termes. Une **formule** est définie par induction comme une égalité ($=$) entre deux termes, une négation ($\neg$), conjonction ($\wedge$), disjonction ($\vee$), implication ($\rightarrow$) de formules, ou une quantification existentielle ($\exists$) ou universelle ($\forall$) d'une formule.

Par exemple, $\varphi_0 \ : \ \forall x \, \exists y \, (x = 0 \vee x = y + 1)$ et $\varphi_1 \ : \ \forall x \, \forall y \, \forall z \, (x + y) + z = x + (y + z)$ sont des formules en logique de Presburger.

Pour réaliser des démonstrations en arithmétique de Presburger, on considère les règles données en annexes, correspondant aux règles de la logique classique auxquelles on ajoute des axiomes et le raisonnement par récurrence.

1. Montrer que les règles suivantes (où t, u et v désignent des termes) sont dérivables à partir des règles $(=_i)$ et $(=_e)$:

$$\frac{\Gamma \vdash u = v}{\Gamma \vdash t[x := u] = t[x := v]} \ (=_c) \qquad \frac{\Gamma \vdash t = u}{\Gamma \vdash u = t} \ (=_s) \qquad \frac{\Gamma \vdash t = u \quad \Gamma \vdash u = v}{\Gamma \vdash t = v} \ (=_t)$$

Solution :
On pose φ la formule $t[x := u] = t$. On obtient alors l'arbre de preuve suivant :

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash t[x := u] = t[x := u]} \ (=_i)}{\Gamma \vdash \varphi[x := u]} \qquad \Gamma \vdash u = v}{\dfrac{\Gamma \vdash \varphi[x := v]}{\Gamma \vdash t[x := u] = t[x := v]}} \ (=_e)$$

Pour la deuxième règle, on pose φ la formule $x = t$. On a :

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash t = t} \ (=_i)}{\Gamma \vdash \varphi[x := t]} \qquad \Gamma \vdash t = u}{\dfrac{\Gamma \vdash \varphi[x := u]}{\Gamma \vdash u = t}} \ (=_e)$$

Enfin, pour la troisième règle, c'est une application directe de $=_e$: on pose φ la formule $t = x$. On a alors :

$$\dfrac{\dfrac{\Gamma \vdash t = u}{\Gamma \vdash \varphi[x := u]} \quad \Gamma \vdash u = v}{\dfrac{\Gamma \vdash \varphi[x := v]}{\Gamma \vdash t = v}} \ (=_e)$$

2. Montrer que la règle (R_2) est dérivable à partir des règles de la logique classique et des autres règles de l'arithmétique de Presburger.

Solution :

$$\dfrac{\dfrac{\dfrac{}{\vdash 0 = 0} \ (=_i)}{\vdash 0 = 0 \vee \exists y \, (0 = y + 1)} \ (\vee_i) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash x + 1 = x + 1} \ (=_i)}{\vdash \exists y \, (x + 1 = y + 1)} \ (\exists_i)}{\vdash x + 1 = 0 \vee \exists y \, (x + 1 = y + 1)} \ (\vee_i)}{x = 0 \vee \exists y \, (x = y + 1) \vdash x + 1 = 0 \vee \exists y \, (x + 1 = y + 1)} \ (Aff)}{\vdash (x = 0 \vee \exists y \, (x = y + 1)) \rightarrow (x + 1 = 0 \vee \exists y \, (x + 1 = y + 1))} \ (\rightarrow_i)}{\vdash \forall x \, (x = 0 \vee \exists y \, (x = y + 1))} \ (rec)$$

On a l'arbre de preuve ci-dessus, ce qui montre bien que la règle (R_2) est dérivable à partir des autres.

3. Montrer que la formule φ_1 est démontrable.

Solution :

On va faire une preuve par récurrence. On commence par l'initialisation :

$$\dfrac{\dfrac{}{\vdash \forall z\,(z+0=z)}\,(R_4)}{\vdash (x+y)+0=x+y}\,(\forall_e) \qquad \dfrac{\dfrac{\dfrac{}{\vdash \forall z\,(z+0=z)}\,(R_4)}{\vdash y+0=y}\,(\forall_e)}{\vdash (x+z)[z:=y+0]=(x+z)[z:=y]}\,(=_c) \quad \dfrac{\vdash x+(y+0)=x+y}{\vdash x+y=x+(y+0)}\,(=_s)}{\vdash (x+y)+0=x+(y+0)}\,(=_t)$$

Puis l'hérédité, en notant φ la formule $(x+y)+z=x+(y+z)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \forall y\,\forall z\,(y+z)+1=y+(z+1)}\,(R_5)}{\vdash (y+z)+1=y+(z+1)}\,(\forall_e)\times 2}{\vdash x+((y+z)+1)=x+(y+(z+1))}\,(=_c) \quad \dfrac{\dfrac{\dfrac{}{\vdash \forall x\,\forall y\,(x+y)+1=x+(y+1)}\,(R_5)}{\vdash \forall y\,(x+y)+1=x+(y+1)}\,(\forall_e)}{\vdash (x+(y+z))+1=x+((y+z)+1)}\,(\forall_e)}{\vdash (x+(y+z))+1=x+(y+(z+1))}\,(=_t)$$
$$\dfrac{\vdash (x+(y+z))+1=x+(y+(z+1))}{\varphi \vdash (x+(y+z))+1=x+(y+(z+1))}\,(Aff)$$

puis :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \forall x\,\forall z\,(x+z)+1=x+(z+1)}\,(R_5)}{\vdash \forall z\,((x+y)+z)+1=(x+y)+(z+1)}\,(\forall_e)}{\vdash ((x+y)+z)+1=(x+y)+(z+1)}\,(\forall_e)}{\varphi \vdash ((x+y)+z)+1=(x+y)+(z+1)}\,(Aff)}{\varphi \vdash (x+y)+(z+1)=((x+y)+z)+1}\,(=_s) \quad \dfrac{\dfrac{}{\varphi \vdash \varphi}\,(Ax)}{\varphi \vdash ((x+y)+z)+1=(x+(y+z))+1}\,(=_c)}{\varphi \vdash (x+y)+(z+1)=(x+(y+z))+1}\,(=_t)$$

et en regroupant les deux preuves précédentes :

$$\dfrac{\dfrac{\dfrac{\varphi \vdash (x+(y+z))+1=x+(y+(z+1)) \quad \varphi \vdash (x+y)+(z+1)=(x+(y+z))+1}{\varphi \vdash (x+y)+(z+1)=x+(y+(z+1))}\,(=_t)}{\vdash \varphi \to (x+y)+(z+1)=x+(y+(z+1))}\,(\to_i)}{\vdash \forall z\,((x+y)+z=x+(y+z)\to (x+y)+(z+1)=x+(y+(z+1)))}\,(\forall_i)$$

On conclut alors :

$$\dfrac{\vdash (x+y)+0=x+(y+0) \quad \vdash \forall z\,(x+y)+z=x+(y+z)\to (x+y)+(z+1)=x+(y+(z+1))}{\dfrac{\vdash \forall z\,(x+y)+z=x+(y+z)}{\vdash \varphi_1}\,(\forall_i)\times 2}\,(rec)$$

Pour la suite, on cherche à démontrer que l'arithmétique de Presburger est décidable, c'est-à-dire qu'il existe un algorithme qui, étant donnée une formule en logique de Presburger, détermine si elle est démontrable ou non. On introduit pour cela une sémantique : on dit qu'une formule φ est **vraie dans** $\mathbb{N}$, et on note $\mathbb{N} \vDash \varphi$, si la formule est vraie en considérant les variables quantifiées comme appartenant à $\mathbb{N}$. Pour une formule non close $\varphi(x_1, \ldots, x_n)$ et des entiers $a_1, \ldots, a_n \in \mathbb{N}^n$, on dit que φ est **vraie dans** $\mathbb{N}$ **aux valeurs** $a_1, \ldots, a_n$, et on note $\mathbb{N} \vDash \varphi(a_1, \ldots, a_n)$, si elle est vraie en substituant chaque x_i par l'entier a_i. On admet que la logique de Presburger est **complète** pour cette sémantique, c'est-à-dire que si $\mathbb{N} \vDash \varphi$, alors φ est démontrable.

À un n-uplet d'entiers $(a_1, \ldots, a_n) \in \mathbb{N}^n$, on associe un mot de Σ^*, $\Sigma = \{0,1\}^n$, noté également

$(a_1, \dots, a_n)$ par abus de notation, défini comme le mot $(a_1^{(0)}, \dots, a_n^{(0)})(a_1^{(1)}, \dots, a_n^{(1)}) \dots (a_1^{(m)}, \dots, a_n^{(m)})$, où $(a_i^{(0)} a_i^{(1)} \dots a_i^{(m)})_2$ correspond à la décomposition binaire de a_i, en commençant par les bits de poids faible. On suppose qu'on rajoute autant de 0 que nécessaire pour que toutes les écritures binaires soient de même taille. Par exemple, au triplet $(2, 7, 1)$, on associe le mot $(0, 1, 1)(1, 1, 0)(0, 1, 0)$.

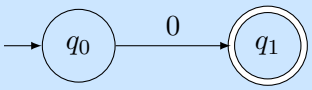
Pour $x_1, \dots, x_n$ des variables, on appelle **formule élémentaire** une formule de la forme $x_i = 0$, $x_i = 1$, $x_i = x_j$ ou $x_i + x_j = x_k$. On dit qu'une formule $\varphi(x_1, \dots, x_n)$ est **rationnelle** si le langage sur Σ $L(\varphi) = \{(a_1, \dots, a_n) \in \Sigma^* \mid \mathbb{N} \vDash \varphi(a_1, \dots, a_n)\}$ est rationnel.

4. Montrer que les formules élémentaires sont rationnelles.

Solution :

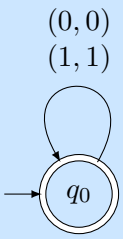
On commence par des cas particuliers :

- $x_1 = 0$, pour $n = 1$, est rationnelle, car $L(x_1 = 0)$ est reconnu par :

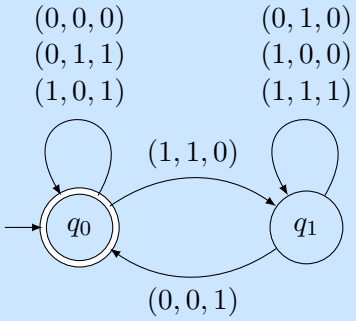


  De même, $x_1 = 1$ est rationnelle.

- $x_1 = x_2$, pour $n = 2$, est rationnelle, car $L(x_1 = x_2)$ est reconnu par :



- $x_1 + x_2 = x_3$, pour $n = 3$, est rationnelle, car $L(x_1 + x_2 = x_3)$ est reconnu par :



  L'état q_0 correspond à l'état sans retenue, et l'état q_1 est l'état qui permet de garder en mémoire la propagation de retenue.

On étend naturellement ces deux solutions à toutes les formules élémentaires en gardant ces transitions pour les variables apparaissant dans la formule et en rajoutant dans chaque uplet toutes les valeurs possibles pour les autres composantes.

5. Montrer qu'une formule obtenue par conjonctions, disjonctions, négations et implications de formules élémentaires est rationnelle. En déduire que toute formule sans quantificateur est rationnelle.

Solution :

On peut montrer que $L(\varphi \wedge \psi) = L(\varphi) \cap L(\psi)$, $L(\varphi \vee \psi) = L(\varphi) \cup L(\psi)$, $L(\neg\varphi) = \overline{L(\varphi)}$ et $L(\varphi \to \psi) = L(\neg\varphi \vee \psi)$. Les langages rationnels étant clos par union, intersection et complémentaire, on obtient le résultat voulu.

Par ailleurs, une formule de la forme $t_1 + t_2 + \dots + t_p = t_{p+1} + \dots + t_q$, où les t_i sont des variables ou des constantes, peut s'écrire comme une conjonction de formules élémentaires :

- pour chaque t_i qui est une constante, on la remplace par une variable y_i et on rajoute une clause $y_i = t_i$;

- en rajoutant des variables intermédiaires, on peut réduire le nombre d'additions de chaque côté de l'égalité. Par exemple, $y_1 + y_2 + y_3 = y_4 + y_5$ est sémantiquement équivalente à $y_1 + z_2 = z_4 \wedge y_2 + y_3 = z_2 \wedge y_4 + y_5 = z_4$.

En combinant ce résultat avec le précédente, on en déduit que toute formule sans quantificateur est rationnelle.

6. On suppose $\varphi(x_1, \dots, x_n)$ rationnelle. Montrer que $\exists x_1\, \varphi(x_1, \dots, x_n)$ est rationnelle.

Solution :

Supposons $\varphi(x_1, \dots, x_n)$ rationnelle et soit $A = (Q, \Sigma = \{0,1\}^n, \delta, q_0, F)$ un AFD reconnaissant $L(\varphi)$. On pose B l'automate non déterministe défini par $B = (Q, \Sigma' = \{0,1\}^{n-1}, \Delta, \{q_0\}, F')$ tel que :

- pour un état $q \in Q$ et une lettre $(b_1, \dots, b_{n-1}) \in \Sigma'$, on pose :

$$\Delta(q, (b_1, \dots, b_{n-1})) = \{\delta(q, (b_1, \dots, b_{n-1}, 0)), \delta(q, (b_1, \dots, b_{n-1}, 1)\}$$

  Autrement dit on supprime la dernière composante des lettres dans les transitions de A ;
- $F' = \{q \in Q \mid \exists p \in F \mid \exists u \in (\{0\}^{n-1} \times \{0,1\})^* \mid \delta^*(q,u) = p\}$, autrement dit les états finaux dans B sont ceux qui permettent d'atteindre un état final de A en lisant un mot dont les lettres ne contiennent que des 0 sur les $n-1$ premières composantes (pour gérer le cas où x_n correspond à l'entier a_n avec l'écriture binaire la plus grande).

Montrons que B reconnait $L(\exists x_n\, \varphi)$:

- supposons $(a_1, \dots, a_{n-1}) \in L(\exists x_n\, \varphi)$. Alors il existe $a_n \in \mathbb{N}$ tel que $\mathbb{N} \vDash \varphi(a_1, \dots, a_n)$. On en déduit que $(a_1, \dots, a_n) \in L(\varphi)$. Distinguons deux cas :
  - si $a_n \neq \max\{a_1, \dots, a_n\}$, alors $\delta^*(q_0, (a_1, \dots, a_n)) \in F \cap \Delta^*(q_0, (a_1, \dots, a_{n-1})) \subset F' \cap \Delta^*(q_0, (a_1, \dots, a_{n-1}))$ donc $(a_1, \dots, a_{n-1}) \in L(B)$;
  - sinon, $\delta^*(q_0, (a_1, \dots, a'_n)) \in F' \cap \Delta^*(q_0, (a_1, \dots, a_{n-1}))$, où a'_n correspond à l'entier dont l'écriture binaire est celle de a_n, tronquée au nombre de bits significatifs le plus grand parmi $a_1, \dots, a_{n-1}$. À nouveau, on a $(a_1, \dots, a_{n-1}) \in L(B)$.
- la réciproque se fait sur la même idée : si $(a_1, \dots, a_{n-1}) \in L(B)$, alors il existe un chemin acceptant dans B. On peut lui associer un chemin acceptant dans A dont les $n-1$ premières composantes des lettres lues correspondent à $(a_1, \dots, a_{n-1})$. On en déduit qu'il existe $a_n \in \mathbb{N}$ tel que $\mathbb{N} \vDash \varphi(a_1, \dots, a_n)$, et donc que $\mathbb{N} \vDash \exists x_n\, \varphi(a_1, \dots, a_{n-1}, x_n)$.

7. En déduire que toute formule de la logique de Presburger est rationnelle, puis que la logique de Presburger est décidable.

Solution :

On traite le cas $\forall x_n\, \varphi(x_1, \dots, x_n)$ en remarquant qu'une telle formule est sémantiquement équivalente à $\neg(\exists x_n\, \neg\varphi(x_1, \dots, x_n))$. De plus, la conjonction, disjonction, négation et implication de formules non élémentaires se traite de la même manière que les formules élémentaires. On conclut par induction sur les formules que toute formule de la logique de Presburger est rationnelle.

On obtient alors un automate sur l'alphabet $\Sigma = \{0,1\}^0$ (c'est-à-dire dont les lettres sont des 0-uplets). Il suffit alors de remarquer que $\mathbb{N} \vDash \varphi \Leftrightarrow L(\varphi) \neq \emptyset$. Or, étant donné un automate fini (déterministe ou non), on peut tester si le langage reconnu est vide par un parcours de graphe : il faut vérifier s'il existe un chemin depuis un état initial vers un état final. Tout cela donne bien un algorithme permettant de tester si une formule est vraie dans $\mathbb{N}$. Comme on a admis la complétude de la logique de Presburger pour cette sémantique, on en déduit que la logique de Presburger est décidable.

Annexe : règles d'inférence de la logique de Presburger

$$\frac{}{\Gamma, \varphi \vdash \varphi}\ (Ax) \qquad \frac{}{\Gamma \vdash \top}\ (\top) \qquad \frac{\Gamma \vdash \varphi}{\Gamma, \psi \vdash \varphi}\ (Aff)$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\ (\to_i) \qquad \frac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\ (\to_e)$$

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}\ (\wedge_i) \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi}\ (\wedge_e^g) \qquad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}\ (\wedge_e^d)$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}\ (\vee_i^g) \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}\ (\vee_i^d) \qquad \frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \theta \quad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta}\ (\vee_e)$$

$$\frac{\Gamma, \varphi \vdash \bot}{\Gamma \vdash \neg\varphi}\ (\neg_i) \qquad \frac{\Gamma \vdash \neg\varphi \quad \Gamma \vdash \varphi}{\Gamma \vdash \bot}\ (\neg_e)$$

$$\frac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \varphi}\ (Abs)$$

$$\frac{\Gamma \vdash \varphi \quad x \notin FV(\Gamma)}{\Gamma \vdash \forall x\ \varphi}\ (\forall_i) \qquad \frac{\Gamma \vdash \forall x\ \varphi \quad FV(t) \cap BV(\varphi) = \emptyset}{\Gamma \vdash \varphi[x := t]}\ (\forall_e)$$

$$\frac{\Gamma \vdash \varphi[x := t]}{\Gamma \vdash \exists x\ \varphi}\ (\exists_i) \qquad \frac{\Gamma \vdash \exists x\ \varphi \quad \Gamma, \varphi[x := y] \vdash \psi \quad y \notin FV(\Gamma \cup \{\varphi, \psi\})}{\Gamma \vdash \psi}\ (\exists_e)$$

$$\frac{}{\Gamma \vdash t = t}\ (=_i) \qquad \frac{\Gamma \vdash \varphi[x := t] \quad \Gamma \vdash t = u}{\Gamma \vdash \varphi[x := u]}\ (=_e)$$

On ajoute les axiomes suivants :

- 0 n'est pas un successeur : $\dfrac{}{\Gamma \vdash \forall x\, \neg(0 = x + 1)}\ (R_1)$
- tout entier non nul est un successeur : $\dfrac{}{\Gamma \vdash \forall x\, (x = 0 \vee \exists y\, x = y + 1)}\ (R_2)$
- l'incrémentation est inversible : $\dfrac{}{\Gamma \vdash \forall x\, \forall y\, (x + 1 = y + 1) \to (x = y)}\ (R_3)$
- 0 est l'élément neutre pour l'addition : $\dfrac{}{\Gamma \vdash \forall x\, (x + 0 = x)}\ (R_4)$
- l'incrémentation est associative : $\dfrac{}{\Gamma \vdash \forall x\, \forall y\, (x + y) + 1 = x + (y + 1)}\ (R_5)$

Enfin, on ajoute une infinité dénombrable de règles exprimant le principe de récurrence : pour toute formule $\varphi(x, y_1, \ldots, y_n)$ ayant pour variables libres x, y_1, …, y_n :

$$\frac{\Gamma \vdash \varphi(0, y_1, \ldots, y_n) \quad \Gamma \vdash \forall x\, (\varphi(x, y_1, \ldots, y_n) \to \varphi(x + 1, y_1, \ldots, y_n))}{\Gamma \vdash \forall x\, \varphi(x, y_1, \ldots, y_n)}\ (rec)$$

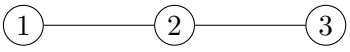
Exercice 4 (Séquence d'élimination) :

Définition

Soit $G = (S, A)$ un graphe non-orienté. On appelle **séquence** une suite finie non-vide σ dont la longueur sera notée $|\sigma| > 0$. On appelle **chemin** dans G une séquence $c = v_1 \dots v_{|c|}$ d'éléments de S telle que pour $1 \leq i < |c|$, on a $\{v_i, v_{i+1}\} \in A$. On dit qu'un chemin dans G est **simple** s'il n'existe pas $1 \leq i < j \leq n$ tels que $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$. Un **cycle** dans G est un chemin simple $c = v_1 \dots v_n$ de longueur $n > 1$ tel que $v_1 = v_n$.

On dit que deux séquences $u_1 \dots u_n$ et $v_1 \dots v_n$ de même longueur se **rencontrent** s'il existe $i \in [\![1, n]\!]$ tel que $u_i = v_i$. Une **séquence d'élimination** pour G est une séquence $\sigma = v_1 \dots v_{|\sigma|}$ d'éléments de S telle que, pour tout chemin c dans G tel que $|c| = |\sigma|$, les séquences σ et c se rencontrent.

1. Construire une séquence d'élimination pour le graphe suivant :



Solution :

$2, 2$ est une séquence d'élimination : soit c un chemin de longueur 1, $c = u_0 u_1$. On distingue les cas :

- si $u_0 = 2$, alors les séquences se rencontrent ;
- sinon, $u_0 = 1$ ou 3, mais alors $u_1 = 2$.

2. Montrer que si G a un cycle, alors G n'admet pas de séquence d'élimination.

Solution :

Supposons qu'il existe une séquence d'élimination $\sigma = v_1 \dots v_n$. Construisons alors un chemin qui ne rencontre jamais σ. Soit C un cycle de longueur $k \geq 3$ dans le graphe. On pose u_1 un sommet de C différent de v_1 (il en existe un). Dès lors, on construit u_{i+1} en choisissant l'un des voisins de u_i dans C, qui est différent de v_{i+1}. Comme u_i a toujours deux voisins dans le cycle, cette construction est possible. On conclut par l'absurde.

3. Pour $k \in \mathbb{N}^*$, on note L_k le graphe (S, A) tel que $S = [\![1, k]\!]$ et $A = \{\{i, i+1\} | 1 \leq i < k\}$.
Construire une séquence d'élimination pour L_k.

Solution :

Si $c = v_1 \dots v_n$ est un chemin dans le graphe, remarquons que v_i et v_j ont la même parité si et seulement si i et j ont la même parité.
De plus, si $n = k$ et v_1 est impair, alors la séquence $\sigma = 1, 2, \dots, k$ rencontre c. En effet, si on suppose que que les deux séquences ne se rencontrent pas, il existe $i \in [\![1, k]\!]$ tel que $i < v_i$ et $i_1 > v_{i+1}$. C'est impossible, car si $i < v_i$, alors $v_i \geq i + 1$ (car v_i a la même parité que i). Dès lors, on propose la séquence d'élimination suivante (l'idée étant de faire deux parcours, en s'assurant que l'un a la même parité que le chemin à rencontrer) :

$$\sigma = 1, 2, \dots, k-1, k, k, k-1, k-2, \dots, 2, 1$$

Définition

Une **séquence d'élimination partielle** pour G et pour un sous-ensemble $X \subset S$ est une séquence $\sigma = v_1 \dots v_{|\sigma|}$ d'éléments de S telle que, pour tout chemin c dans G tel que $|c| = |\sigma|$, si le dernier élément de c est dans X, alors σ et c se rencontrent.

On note $\phi_E : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$ la fonction définie par :

$$\phi_E(X) := S \setminus \{v | \exists v' \in S \setminus X, \{v, v'\} \in A\}$$

4. Montrer que pour $X \subset S$, σ une séquence d'élimination partielle pour G et X et $v \in S$, alors la séquence σ' obtenue en concaténant σ et v est une séquence d'élimination partielle pour G et $\phi_E(X) \cup \{v\}$.

Solution :
Informellement, $\phi_E(X)$ est l'ensemble des sommets qui n'ont que des voisins dans X. Dès lors, avec les hypothèses de l'énoncé, soit $c = v_1 \dots v_n v_{n+1}$ un chemin dont le dernier élément est dans $\phi_E(X) \cup \{v\}$.

- Si $v_{n+1} = v$, alors c et σ' se rencontrent bien (en $i = n+1$).
- Sinon, $v_{n+1} \in \phi_E(X)$ et donc, par définition, tout voisin de v_{n+1} est dans X. On en déduit que $v_n \in X$, et par définition de σ que σ rencontre $v_1 \dots v_n$ et donc que σ' rencontre c.

5. En déduire un algorithme pour décider si un graphe admet une séquence d'élimination, et si oui, la calculer. Discuter de l'efficacité de l'algorithme et de la longueur de la séquence obtenue.
Indication : On construira un graphe orienté sur $\mathcal{P}(S)$.

Solution :
On ramène à un problème d'accessibilité dans un graphe. On construit le graphe orienté $G' = (S', A')$ tel que :

- $S' = \mathcal{P}(S)$
- Pour $A \in S'$ et $v \in S$, $(A, \phi_E(A) \cup \{v\}) \in A'$. On peut considérer que l'arête est étiquetée par v.

S'il existe un chemin de $\emptyset$ à S dans G', alors, d'après la question précédente et par une récurrence, il existe une séquence d'élimination pour G. De plus, le chemin permet de construire la séquence d'élimination.
Réciproquement, s'il existe une séquence d'élimination pour G, $\sigma = v_1 \dots v_n$. Par construction du graphe, il existe un unique chemin étiqueté par σ passant par les sommets $V_1, \dots, V_n$. Montrons que ce chemin termine en S en montrant que si $u \notin V_i$, alors il existe un chemin c se terminant par u tel que c ne rencontre pas $\sigma_i = v_1 \dots v_i$.

- Le résultat est trivialement vrai pour V_1 (qui ne contient qu'un seul élément).
- Supposons le résultat vrai pour $i < n$. On sait que $V_{i+1} = \phi_E(V_i) \cup \{v_i\}$. Supposons $u \notin V_{i+1}$. On sait donc que $u \neq v_i$ et $u \notin \phi_E(V_i)$. On en déduit qu'il existe $u' \notin V_i$ tel que u et u' sont voisins. Dès lors, il existe un chemin c terminant par u' qui ne rencontre pas σ_i et donc cu ne rencontre pas $\sigma_{i+1} = \sigma_i v_i$.

Dès lors, comme $\sigma_n = \sigma$ est une séquence d'élimination, on en déduit que $V_n = S$.
La complexité d'un tel algorithme est exponentielle :

- La construction du graphe G' (dont le nombre de sommets est $2^{|S|}$) est exponentielle et nécessite le calcul des arêtes (on calcule $\phi_E(A)$ pour chaque ensemble).
- La recherche du chemin se fait en temps $O(|S'| + |A'|)$ ou $O(|S'|^2)$ selon la structure (parcours en largeur par exemple). À nouveau, la complexité est exponentielle.
- La longueur du chemin est potentiellement exponentielle également.

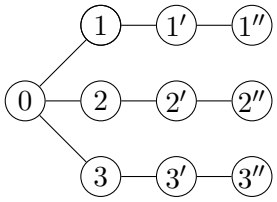
6. Pour tout $k \in \mathbb{N}^*$, on note G_k le graphe suivant :

$$(\{0\} \cup \{1, \dots, k\} \cup \{1', \dots, k'\}, \{\{0, i\} | 1 \leq i \leq k\} \cup \{\{i, i'\} | 1 \leq i \leq k\})$$

Construire une séquence d'élimination pour G_k.

Solution :
Soit $\sigma' = (1, 0, 2, 0, \dots k)$ et $\sigma = \sigma'\sigma'$. Alors $\sigma = (s_1, \dots, s_{4k-2})$ est une séquence d'élimination pour G_k.
Appelons sommet impair un sommet $i \in [\![1, k]\!]$ et pair un des autres. Comme à la question 3, un chemin change toujours de parité à chaque étape. Si le premier sommet est un sommet impair, alors il y aura une rencontre pendant le premier passage de σ'. De même pour le deuxième passage si le premier sommet est pair.

7. Montrer que le graphe suivant n'admet pas de séquence d'élimination :



Solution :

Soit $\sigma = v_1 \dots v_n$ une séquence d'éléments de S, ne commençant pas par 0. On définit un chemin c de la façon suivante :

- $u_1 = 0$
- Pour $i \in [\![1, n-1]\!]$:
  - si $u_i = j''$, alors $u_{i+1} = j'$ (seul choix possible) ;
  - si $u_i = j'$, alors $u_{i+1} = j$ sauf si $v_{i+1} = j$, auquel cas $u_{i+1} = j''$;
  - si $u_i = j$, alors $u_{i+1} = 0$ sauf si $v_{i+1} = 0$, auquel cas $u_{i+1} = j'$;
  - si $u_i = 0$, alors $u_{i+1} = j$ où j est choisi de la façon suivante : $j \neq v_{i+1}$ et on pose, pour $k \in \{1, 2, 3\} \setminus \{v_{i+1}\}$: $i < \ell_k < n$ la plus petite position après i dont la parité est $\neq i$ et qui satisfait $v_{\ell_k} = k$ et $v_{\ell_k+1} = k'$. Si l'un des deux ℓ_k n'est pas défini, on pose $j = k$. Sinon, on pose $j = k$ tel que ℓ_k est maximal.

On définit ici bien un chemin, et il ne rencontre pas σ. Le seul cas de rencontre possible serait lorsque $u_i = 0$, mais on choisit le bon sommet pour éviter une éventuelle collision par la suite.

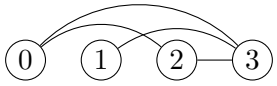
Exercice 5 (Graphes de saut) :

Définition

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

Exemple

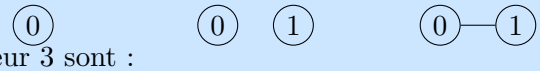
Voici un graphe de saut de longueur 4.



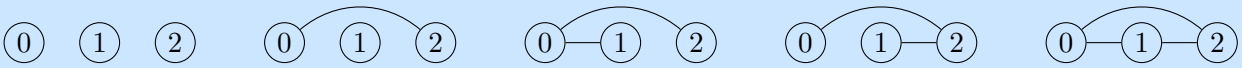
1. Dessiner tous les graphes de saut de longueur 1, 2 et 3.

Solution :

Les graphes de saut de longueur 1 et 2 sont :



Les graphes de saut de longueur 3 sont :



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.

Solution :

On commence par remarquer que la définition est équivalente à : $\forall b < c, \{b, c\} \in A \Rightarrow (b = 0$ ou $\{b-1, c\} \in A)$ et $(c = n-1$ ou $\{b, c+1\} \in A)$. 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

Données : une matrice d'adjacence M de taille $n \times n$
Résultat : M est-elle la matrice d'adjacence d'un graphe de saut ?
pour $b = 0$ *à* $n-1$ **faire**
  pour $c = b$ *à* $n-1$ **faire**
    si $M[b][c] = 1$ **alors**
      si $(b > 0 \wedge M[b-1][c] = 0) \vee (c < n-1 \wedge M[b][c+1] = 0)$ **alors**
        retourner *Faux* ;
retourner *Vrai* ;

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.

Solution :

On considère f la fonction qui prend en argument un graphe de saut comprimé $G = ([\![0, n]\!], A)$ de longueur $n+1$ en un graphe de saut $G' = ([\![0, n-1]\!], A')$ de longueur n de la manière suivante :

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

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 $\{b, c\} \in A'$, $b < c$ et (a, d) tels que $a \leq b$ et $c \leq d$. Alors $\{b, c+1\} \in A$ par définition de A'. G étant un graphe de saut, on en déduit $\{a, d+1\} \in A$, ce qui implique $\{a, d\} \in A'$ et G' est bien un graphe de saut.

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

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

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

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 $P_0, \ldots, P_n$.

Solution :

Soit $n \in \mathbb{N}$ et soit $G = (\llbracket 0, n \rrbracket, A)$ un graphe de saut de taille $n+1$. Soit $i \leq n$ le plus petit sommet tel que $\{i, i+1\} \in A$, 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 $\{i, i+1\} \in A$ 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 : $P_{n+1} = \sum_{i=0}^{n} P_i P_{n-i}$.

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.

Solution :

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 0132 0213 0231 0312 0321 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. Décrire en français un algorithme naïf prenant en argument un entier $n \geq 0$ et une matrice d'adjacence

M de taille $n \times n$ et décide s'il existe une permutation de $\{0, \dots, n-1\}$ telle que le graphe résultant est un graphe de saut. Déterminer sa complexité en temps et en espace.

Solution :

On propose alors l'algorithme suivant :

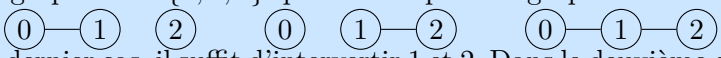
- Poser T le tableau $[0, 1, \dots, n-1]$.
- Tant que T n'est pas le tableau vide, faire les deux instructions suivantes :
  - Si le graphe G auquel on applique la permutation T est un graphe de saut, renvoyer Vrai.
  - Transformer T par l'algorithme précédent.
- Renvoyer Faux.

La complexité en temps est $\mathcal{O}(n!n^2)$ et celle en espace est $\mathcal{O}(n)$.

7. 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.

Solution :

Il suffit d'étudier les graphes sur $\{0, 1, 2\}$ qui ne sont pas des graphes de saut, à savoir :

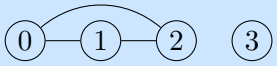


Dans le premier et le dernier cas, il suffit d'intervertir 1 et 2. Dans le deuxième cas, il suffit d'intervertir 0 et 1.

8. Déterminer un graphe $G = (\{0, 1, 2, 3\}, A)$ dont aucune permutation ne résulte en un graphe de saut.

Solution :

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. Que dire de l'existence de graphes non connexes tels qu'il existe une permutation des sommets donnant un graphe de saut ?

Solution :

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.

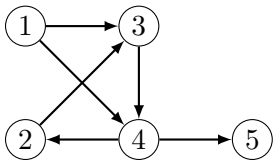
Exercice 6 (Familles closes de graphes) :

Définition

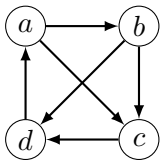
Un **graphe** $G = (S, A)$ est la donnée d'un ensemble fini non vide de sommets S et d'un ensemble d'arêtes $A \subset S \times S$. On imposera toujours que $S \subset \mathbb{N}$ et on autorise les **boucles**, c'est-à-dire les arêtes de la forme (v, v) allant d'un sommet $v \in S$ vers lui-même.

Un **homomorphisme** d'un graphe $G = (S, A)$ vers un graphe $G' = (S', A')$ est une fonction $\phi : V \to S'$ telle que, pour tout $(x, y) \in A$, $(\phi(x), \phi(y)) \in A'$.

1. Pour G et G' représentés ci-dessous, décrire un homomorphisme de G vers G'.
Y a-t-il un homomorphisme de G' vers G ?



G



G'

Solution :
Plusieurs solutions sont possibles, comme par exemple $\phi(1, 2, 3, 4, 5) = (b, a, c, d, a)$.
Réciproquement, il n'existe pas d'homomorphisme : tous les sommets de G' ont une arête entrante et une arête sortante, donc ne pourraient être envoyés que vers $2, 3, 4$ (qui sont dans la même situation). Si $\phi(a) = 2$, alors $\phi(b) = 3$ et $\phi(c) = 4$ mais alors l'arête (a, c) ne peut pas être représentée (et de même si $\phi(a) = 3$ ou 4).

2. Écrire le pseudocode d'un algorithme naïf qui détermine, étant donnée la matrice d'adjacence de deux graphes G et G', s'il existe un homomorphisme de G vers G' et le calcule le cas échéant. Déterminer sa complexité en temps et en espace.

Solution :
On peut agir par force brute en générant toutes les fonctions de S dans S' et en vérifiant si l'une d'entre elle est un homomorphisme. La vérification se fait de la manière suivante :

Algorithme : Vérification
Données : matrices carrées M et M' de tailles n et m, tableau Φ de taille n
Résultat : Φ est-il un homomorphisme de M vers M' ?
pour $i = 1$ *à* n **faire**
  pour $j = 1$ *à* n **faire**
    si $M[i][j] = 1 \wedge M'[\Phi[i]][\Phi[j]] = 0$ **alors**
      retourner *Faux* ;
retourner *Vrai* ;

Pour générer toutes les fonctions de S dans S', on peut penser à un algorithme d'incrémentation :

Algorithme : Incrémentation
Données : tableau Φ de taille n, entier m
Résultat : Modifie Φ par effets de bords pour obtenir le prochain tableau à tester
$i \leftarrow 1$;
tant que $i < n + 1 \wedge Phi[i] = m$ **faire**
  $\Phi[i] \leftarrow 1$;
  $i \leftarrow i + 1$;
si $i < n + 1$ **alors**
  $\Phi[i] \leftarrow \Phi[i] + 1$;

Finalement, la vérification d'existence d'homomorphisme se fait de la manière suivante :

Algorithme : Homomorphisme

Données : matrices carrées M et M' de tailles n et m
Résultat : Existe-t-il un homomorphisme de M vers M' ?
$\Phi \leftarrow [1, \dots, 1]$ (taille n) ;
tant que *Vrai* **faire**
  si *Vérification*(M, M', Φ) **alors**
    retourner Φ ;
  si $\Phi = [m, \dots, m]$ **alors**
    retourner *Faux* ;
  Incrémentation(Φ) ;

Il est clair que la vérification se fait en temps $\mathcal{O}(n^2)$, et on peut montrer que l'incrémentation est en $\mathcal{O}(1)$ amortie (mais est trivialement en $\mathcal{O}(n)$). Comme il existe m^n fonctions de $[\![1, n]\!]$ dans $[\![1, m]\!]$, la complexité temporelle totale est $\mathcal{O}(n^2 m^n)$.
La complexité spatiale ne concerne que la création de Φ, donc est en $\mathcal{O}(n)$.

3. Soit G et G' deux graphes orientés et soit $G_1, \dots, G_n$ et $G'_1, \dots, G'_m$ les composantes connexes (c'est-à-dire dans les représentations désorientées des graphes) de G et G' respectivement. Supposons que l'ont ait déterminé, pour chaque $(i, j) \in [\![1, n]\!] \times [\![1, m]\!]$ s'il existe un homomorphisme de G_i vers G'_j. Peut-on déterminer s'il existe un homomorphisme de G vers G' ?

Solution :
On commence par montrer que l'image d'un graphe connexe par un homomorphisme est connexe. Soit G un graphe connexe et G' son image par ϕ un homomorphisme. Soient $x, y \in \phi(S)$ et $u, v \in S$ tels que $\phi(u) = x$ et $\phi(v) = y$. G étant connexe, il existe un chemin non orienté entre u et v. Il est clair que l'image de ce chemin est un chemin non orienté entre x et y dans G'.
Cela suggère, dans le cas général, qu'il existe un homomorphisme de G vers G' si et seulement si pour chaque C.C. de G, il existe un homomorphisme vers une C.C. de G'. Le sens direct découle de la propriété précédente (si V_i est l'ensemble de sommets d'une C.C. de G, alors $\phi(V_i)$ est inclus dans une C.C. de G'). Réciproquement, il suffit de définir $\phi(v)$ comme étant égal à $\phi_{i,j}(v)$, $\phi_{i,j}(v)$ étant un homomorphisme de la C.C. V_i contenant v vers une C.C. S'_j de G'.

Définition

Une **famille** de graphes est un ensemble de graphes (possiblement infini). Une famille $\mathcal{F}$ est dite **close** si pour tout graphe $G \in \mathcal{F}$, si G a un homomorphisme vers un graphe G', alors $G' \in \mathcal{F}$.

4. On pose $\mathcal{F}_C$ la famille des graphes possédant un cycle. Montrer que $\mathcal{F}_C$ est close.

Solution :
Soit $v_1, \dots, v_n = v_1$ un cycle de G et ϕ un homomorphisme de G vers G'. Alors $\phi(v_1), \dots, \phi(v_n)$ est un cycle de G', donc $G' \in \mathcal{F}_C$.

5. Déterminer un graphe $G_\perp$ tel que la famille de tous les graphes est l'unique famille close contenant $G_\perp$.

Solution :
On définit $G_\perp = (\{0\}, \emptyset)$. Soit $G = (S, A)$ un graphe et $v \in S$. On définit une fonction ϕ de $G_\perp$ vers G par $\phi(0) = v$. Il est clair que ϕ est un homomorphisme, donc si $G_\perp \in \mathcal{F}$ une famille close, alors $G \in \mathcal{F}$. De la même manière, tout graphe sans arête convient.

6. Justifier qu'une famille close non vide est nécessairement infinie.

Solution :
Soit $G \in \mathcal{F}$ une famille close. En rajoutant un nombre arbitraire de sommets à G pour former G', il existe clairement un homomorphisme de G vers G'. On en déduit que $\mathcal{F}$ est infinie.

Définition

Un graphe $G = (S, A)$ est dit **minimal** pour une famille close $\mathcal{F}$ si tout graphe obtenu à partir de G en supprimant une arête n'est pas dans $\mathcal{F}$.

7. Montrer que si $\mathcal{F} \neq \emptyset$ alors $\mathcal{F}$ admet un graphe minimal.

Solution :
Soit $G \in \mathcal{F}$. On peut supprimer des arêtes de G tant qu'il en existe une qui le laisse dans $\mathcal{F}$. Le graphe ainsi obtenu est bien minimal (éventuellement sans arête auquel cas $\mathcal{F}$ contient tous les graphes).

Définition

Une famille close $\mathcal{F}$ est dite **finiment engendrée** s'il existe une famille finie $\mathcal{F}'$ telle que $\mathcal{F}$ est exactement l'ensemble des graphes G tel qu'il existe un homomorphisme d'un graphe de $\mathcal{F}'$ vers G.

8. Donner un exemple de famille close finiment engendrée et de famille close qui ne l'est pas.

Solution :
La famille contenant tous les graphes est finiment engendrée (par un unique graphe sans arête comme précédemment). Montrons que $\mathcal{F}_C$ n'est pas finiment engendrée. Par l'absurde, supposons $\mathcal{F}$ une famille finie de cardinal qui engendre $\mathcal{F}_C$. Soit n l'ordre maximal d'un graphe de $\mathcal{F}$ et soit C_{n+1} le cycle à $n+1$ sommets. Supposons que $G = (S, A) \in \mathcal{F}$ est tel qu'il existe un homomorphisme ϕ de G vers C_{n+1}. Par hypothèse, $|\phi(S)| \leq |S| \leq n$. Considérons G' le sous-graphe de G induit par $\phi(S)$. Comme $\phi(S) < n+1$, il est clair que G' n'est pas cyclique. De plus, il est clair que ϕ est un homomorphisme de G vers G'. Enfin, comme $\mathcal{F}$ engendre $\mathcal{F}_C$, cela implique que $G' \in \mathcal{F}_C$. On conclut par l'absurde.

9. Montrer que si $\mathcal{F}$ est une famille close qui n'est pas finiment engendrée, alors $\forall n \in \mathbb{N}$, il existe un graphe minimal pour $\mathcal{F}$ ayant au moins n arêtes.

Solution :
On montre la contraposée. Soit $\mathcal{F}$ une famille close telle qu'il existe $n \in \mathbb{N}$ tel que tout graphe minimal de $\mathcal{F}$ a au plus n arêtes. Soit $\mathcal{F}_{\min}$ l'ensemble des graphes minimaux pour $\mathcal{F}$. On peut restreindre $\mathcal{F}_{\min}$ en supprimant les sommets isolés (sauf éventuellement le dernier) et en renommant les sommets de 1 à $2n$ (un graphe à n arêtes sans sommet isolé a au plus $2n$ sommets), de telle sorte qu'on considère que $\mathcal{F}_{\min}$ est fini. Montrons que cette famille engendre $\mathcal{F}$. Soit G tel qu'il existe un homomorphisme de $\mathcal{F}_{\min}$ vers G. $\mathcal{F}$ étant close et $\mathcal{F}_{\min} \subset \mathcal{F}$, cela implique que $G \in \mathcal{F}$.

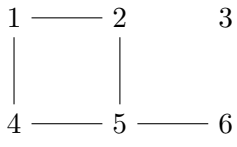
Réciproquement, soit $G \in \mathcal{F}$. Montrons qu'il existe un homomorphisme d'un graphe de $\mathcal{F}_{\min}$ vers G. Comme à une question précédente, on peut retirer des arêtes de G jusqu'à obtenir un graphe minimal (donc dans $\mathcal{F}_{\min}$, à homomorphisme près), qui aura bien un homomorphisme vers G.

Exercice 7 (Acyclicité dans les hypergraphes) :

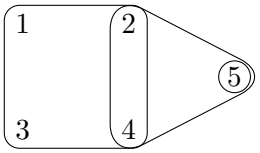
Définition

Un **graphe non orienté** est un couple $G = (S, A)$ où $A \subset \mathcal{P}_2(S)$ est l'ensemble des arêtes. On définit un **hypergraphe** comme un couple $H = (S, M)$ où $M \subset \mathcal{P}(S) \setminus \{\emptyset\}$ est un ensemble d'**hyperarêtes**, c'est-à-dire de sous-ensembles de S de cardinal au moins 1. Ainsi, un graphe peut être vu comme un hypergraphe dont toutes les hyperarêtes sont de cardinal 2. Pour $X \subset S$, on définit également le **projeté** sur X par $H[X] = (X, \{m \cap X | m \in M\})$.

1. On donne le graphe G_0 et l'hypergraphe H_0 ci-dessous. Dessiner $G_0[\{2,3,4,5\}]$ et $H_0[\{1,2,5\}]$.



G_0



H_0

Solution :

Question de prise en main. Il ne faut pas hésiter et dessiner ce qui est intuitif.

Définition

On définit le **réduit** $R(H)$ d'un hypergraphe H comme étant l'hypergraphe H dans lequel on a supprimé les hyperarêtes incluses dans une autre hyperarête. Par exemple, dans le réduit de H_0, on a supprimé le singleton $\{5\}$ comme hyperarête.

On dit qu'un hypergraphe $H = (S, M)$ a un **cycle induit** s'il existe une séquence $(v_1, \dots, v_n)$ d'éléments distincts dans S telle que $n \geq 3$ et si $X = \{v_1, \dots, v_n\}$ et $R(H[X]) = (X, M_X)$, alors $M_X = \{\{v_i, v_{i+1}\} | i \in [\![1, n-1]\!]\} \cup \{v_n, v_1\}$.

2. Déterminer si G_0 et H_0 possèdent des cycles induits.

Solution :

G_0 possède un cycle induit, en prenant $(1, 2, 5, 4)$ comme séquence et $X = \{1, 2, 4, 5\}$. H_0 ne possède pas de cycle induit, car on ne peut pas choisir dans la séquence trois sommets de la même multi-arête. Choisir ≥ 3 sommets pour former un cycle induit revient donc (sans perte de généralité) à choisir $X = (1, 2, 5)$ qui ne forme pas de cycle dans son réduit projeté.

3. Montrer qu'un graphe G possède un cycle induit si et seulement s'il possède un cycle au sens usuel.

Solution :

Le sens direct est évident, car si un réduit projeté possède des multi-arêtes, le graphe possède les mêmes arêtes. Pour le sens réciproque, il faut par contre considérer le plus petit cycle usuel dans le graphe. Cela nous assure qu'il n'existe pas d'arête qui "traverse" le cycle.

4. Exhiber un hypergraphe $H = (S, M \cup \{m\})$ qui ne possède pas de cycle induit, mais tel que (S, M) possède un cycle induit. Commenter.

Solution :

L'hypergraphe $H = \{\{1,2,3\}, \{\{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}\}$ convient en supprimant la multi-arête $\{1,2,3\}$. Cela change par rapport aux graphes usuels, dans lequel supprimer des arêtes ne peut pas créer des cycles.

Définition

Pour un hypergraphe $H = (S, M)$, on définit le **graphe d'incidence** de H par $I(H) = (S \cup M, E_I)$ où $E_I = \bigcup_{m \in M} \bigcup_{v \in m} \{\{v, m\}\}$. On dit qu'un hypergraphe est **Berge-cyclique** si son graphe d'incidence $I(H)$ contient un cycle (au sens usuel).

5. Montrer qu'un graphe G contient un cycle (au sens usuel) si et seulement s'il est Berge-cyclique.
Est-il possible qu'un hypergraphe possède un cycle induit sans être Berge-cyclique ?
Est-il possible qu'un hypergraphe soit Berge-cyclique sans posséder de cycle induit ?

Solution :

Pour la première question, il suffit (pour les deux sens) de voir que la construction du graphe d'incidence consiste à rajouter un sommet intermédiaire sur chaque arête. La cyclicité est donc conservée.

Un hypergraphe H qui possède un cycle induit est toujours Berge-cyclique, car si dans un réduit projeté il existe une multi-arête $\{v_1, v_2\}$, alors cela signifie que dans H, v_1 et v_2 sont dans une même multi-arête m. Il existe donc un chemin $v_1 - m - v_2$ dans le graphe d'incidence. On construit de cette façon un cycle dans le graphe d'incidence.

Pour la dernière question, le contre-exemple de la question précédente convient (car il est Berge-cyclique).

Définition

Dans un hypergraphe $H = (S, M)$, une **feuille** $v \in S$ est un sommet tel qu'il existe une hyperarêtes m_v qui contient toutes les hyperarêtes contenant v. Pour une feuille v, on note $H[-v] = H[S \setminus \{v\}]$.

6. Soit H un hypergraphe et v une feuille de H. Montrer que H possède un cycle induit si et seulement si $H[-v]$ possède un cycle induit.

Solution :

Il s'agit ici de montrer qu'une feuille ne peut pas faire partie d'un cycle induit. En effet, si dans un cycle induit on voit les multi-arêtes $v_1 - v - v_2$, v étant une feuille, il existe nécessairement une multi-arête contenant v, v_1 et v_2, donc on ne peut pas choisir v, v_1 et v_2 pour construire le cycle induit.

7. Montrer qu'il existe un hypergraphe H Berge-cyclique qui possède une feuille v tel que $H[-v]$ ne soit pas Berge-cyclique.

Solution :

Le contre-exemple de la question 4 convient, car en enlevant un sommet, on ne peut plus construire de cycle.

8. Montrer que si un hypergraphe est non vide et non Berge-cyclique, alors il possède une feuille.

Solution :

Il s'agit ici de raisonner par contraposée : si un hypergraphe ne possède pas de feuille, alors on choisit un sommet quelconque v_0 et on peut le relier, dans le graphe d'incidence, à un sommet v_1 de la même multi-arête. Dès lors, il existe un autre sommet $v_2 \notin \{v_0, v_1\}$ tel que v_1 et v_2 sont dans la même multi-arête (sinon, v_1 serait une feuille). On procède ainsi jusqu'à retomber sur un sommet déjà visité pour créer un cycle.

Exercice 8 (Largeur de bande) :

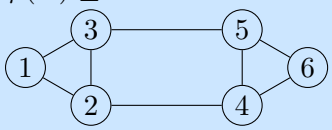
Définition

Soit $G = (S, A)$ un graphe non orienté connexe. Pour une fonction dite **d'étiquetage** $f : S \longrightarrow \mathbb{N}$ injective, on définit la quantité : $\varphi(f, G) = \max\{|f(u) - f(v)|, \{u, v\} \in A\}$.

La **largeur de bande** de G, notée $\varphi(G)$ est le plus petit élément de l'ensemble des $\varphi(f, G)$ lorsque G parcours l'ensemble des fonctions d'étiquetage.

1. Calculer la largeur de bande du graphe suivant :

Solution :
Il existe un étiquetage montrant que $\varphi(G) \leq 2$:



De plus, soit f une fonction d'étiquetage. Il existe un sommet u de degré 3, donc au moins l'un de ses voisins v vérifie $|f(u) - f(v)| \geq 2$.
Finalement, on en déduit $\varphi(G) = 2$.

2. Montrer qu'il suffit, dans la définition, de se limiter aux fonctions bijectives $f : V \longrightarrow [\![1, |S|]\!]$.

Solution :
Soit f une fonction d'étiquetage d'image $f(S)$. Soit g l'unique fonction strictement croissante de $f(S)$ dans $[\![1, |S|]\!]$. Alors pour tout sommet u, v, $|f(u) - f(v)| \geq |\tilde{f}(u) - \tilde{f}(v)|$.

Pour la suite, les fonctions d'étiquetage seront considérées à valeurs dans $[\![1, |S|]\!]$.

3. Calculer la largeur de bande d'un cycle à $n \geq 3$ sommets.

Solution :
Dans un premier temps, remarquons que pour un étiquetage f, on ne peut pas avoir $\varphi(G, f) = 1$. En effet, cela imposerait aux d'étiquettes 1 et n d'être voisins. Dès lors :

- Si n est pair, on utilise l'étiquetage : $1, 3, 5, \ldots, n-1, n, n-2, \ldots, 4, 2$ qui vérifie $\varphi(G, f) = 2$.
- Si n est impair, on utilise l'étiquetage : $1, 3, 5, \ldots, n-2, n, n-1, \ldots, 4, 2$ qui vérifie $\varphi(G, f) = 2$.

On en déduit que $\varphi(G) = 2$.

4. Montrer que $\deg(G)$, le degré maximal d'un sommet de G, est inférieur ou égal à $2\varphi(G)$.

Solution :
Soit f une fonction d'étiquetage telle que $\varphi(G) = \varphi(G, f)$ et soit v un sommet quelconque. On sait qu'il existe au plus $2\varphi(G)$ entiers $k \neq f(v)$ tels que $|f(v) - k| \leq \varphi(G)$. On en déduit donc, par injectivité qu'il existe au plus $2\varphi(G)$ voisins de v.

5. On considère un coloriage des sommets de G tel que chaque arête de G relie deux sommets de couleurs distinctes. Montrer que le nombre minimal de couleurs utilisées pour un tel coloriage est majoré par $\varphi(G) + 1$.

Solution :
Montrons qu'il existe un coloriage à $\varphi(G) + 1$ couleurs. Soit f une fonction d'étiquetage telle que $\varphi(G) = \varphi(G, f)$. Associons à chaque sommet v la couleur $f(v) \mod (\varphi(G) + 1)$. Deux sommets voisins u et v sont de la même couleur si et seulement si $\varphi(G) + 1$ divise $|f(u) - f(v)|$. Or par définition de $\varphi(f, G)$ cette quantité est comprise entre 1 et $\varphi(G)$. On en déduit le résultat.

6. Montrer que $\frac{|S|-1}{\delta(G)} \leq \varphi(G) \leq |S| - \delta(G)$ où $\delta(G)$ désigne le diamètre du graphe, c'est-à-dire la plus grande distance entre deux sommets.

Solution :

Soit f une fonction d'étiquetage de G, v_1 et v_p les sommets tels que $f(v_0) = 1$ et $f(v_p) = |S|$. Soit $v_0, v_1, \ldots, v_p$ les sommets d'un chemin simple reliant v_0 à v_p. Alors on a :

$$|S| - 1 = |f(v_0) - f(v_p)| \leq \sum_{k=1}^{p} |f(v_{k-1} - f(v_k)| \leq p\varphi(G) \leq \delta(G)\varphi(G)$$

On a alors la première inégalité.

Soit $D = v_0 \ldots v_\delta$ un diamètre (simple) de G. Soit f la fonction d'étiquetage des sommets dans l'ordre d'un parcours en largeur depuis v_0. Posons $S_k = \{v \in S | d(v_0, v) = k\}$ pour k allant de 0 à $\delta(G)$. Alors deux sommets voisins u et v sont :

- Soit dans le même S_k, et dans ce cas, $|f(u) - f(v)| \leq |S_k| - 1 = |S_k| - |S_k \cap D| \leq |S| - \delta(G)$.
- Soit dans deux ensembles consécutifs S_k et S_{k+1}, et alors :

$$\begin{aligned} |f(u) - f(v)| &\leq |S_k| + |S_{k+1}| - 1 \\ &\leq |S_k| + |S_{k+1}| - 1 + \sum_{i=0}^{k-1}(|S_i| - 1) + \sum_{i=k+1}^{\delta(G)}(|S_i| - 1) \\ &= |S| - 1 - k - (\delta(G) - (k+2) + 1) \\ &= |S| - \delta(G) \end{aligned}$$

On a bien la majoration attendue.

Exercice 9 (Nombre prescrit de chemins) :

Définition

Un **graphe orienté** est une paire $G = (S, A)$ où $A \subset S \times S$ est l'ensemble des arêtes.

Un **chemin** de $u \in S$ à $v \in S$ est une séquence $u_1, \dots, u_n$ de sommets telle que $u_1 = u$, $u_n = v$ et pour tout $i \in [\![1, n-1]\!]$, $(u_i, u_{i+1}) \in A$.

Un **graphe pointé** est un triplet (G, s, t) où $G = (S, A)$ est un graphe orienté et $s, t \in S$.

On dit qu'un graphe pointé **réalise** un entier $n \in \mathbb{N}$ s'il existe exactement n chemins de s à t dans G.

1. Construire un graphe pointé qui réalise 3.

Solution :
De nombreuses possibilités, on peut se contenter de la version $n = 3$ de la question suivante.

2. Pour tout $n \in \mathbb{N}$, proposer une construction naïve d'un graphe pointé qui réalise n et détailler son nombre de sommets.

Solution :
On considère le graphe à $n + 2$ sommets :

$$S = \{s, t\} \cup \{u_i \mid i \in [\![1, n]\!]\}$$
$$A = \{(s, u_i) \mid i \in [\![1, n]\!]\} \cup \{(u_i, t) \mid i \in [\![1, n]\!]\}$$

3. Quels sommets peuvent être ignorés dans l'étude d'un graphe pointé réalisant un entier $n \in \mathbb{N}^*$? Quelles hypothèses peut-on alors faire sur les arêtes ?

Solution :
En utilisant le vocabulaire des automates, on peut considérer le graphe "émondé", c'est-à-dire où on a supprimé les sommets non accessibles (depuis s) ou non co-accessibles (depuis t). En effet, ces sommets ne peuvent pas appartenir à un chemin de s à t. De plus, on peut considérer que le graphe ne possède pas de cycle, car un chemin doit être constitué de sommets distincts.

4. Pour $k \in \mathbb{N}$, donner une preuve constructive de l'existence d'un graphe pointé à $k + 2$ sommets réalisant 2^k.

Solution :
Considérons le graphe $G = (S, A)$ où $S = \{u_i \mid i \in [\![0, k+1]\!]\}$ et $A = \{(u_i, u_j) \mid 0 \leq i < j \leq k+1\}$ et montrons par récurrence décroissante que (G, u_i, u_{k+1}) réalise 2^{k-i} pour $i \in [\![0, k]\!]$.

- si $i = k$: il n'existe qu'un seul chemin de u_k à u_{k+1}, à savoir l'arête (u_k, u_{k+1}) ;
- supposons le résultat vrai pour $i \in [\![1, k]\!]$ et montrons qu'il reste vrai pour $i - 1$: un chemin de u_{i-1} à u_{k+1} est constitué d'une arête de u_{i-1} à u_j où $j \in [\![i, k+1]\!]$, suivi d'un chemin de u_j à u_{k+1}. En utilisant l'hypothèse de récurrence, le nombre total de tels chemin est :

$$\sum_{j=i}^{k} 2^{k-j} + 1 = 2^{k-i+1}$$

le $+1$ dans l'expression ci-dessus étant le chemin qui est l'arête (u_{i-1}, u_{k+1}).

On conclut par récurrence.

5. Pour $n \in \mathbb{N}^*$, construire un graphe pointé à $\lceil \log_2 n \rceil + 2$ sommets réalisant n.

Solution :
Si n est une puissance de 2, on peut utiliser le graphe précédent. Si n n'est pas une puissance de 2, l'idée est de repartir du graphe précédent pour $k = \lfloor \log_2 n \rfloor$, de rajouter un sommet et des arêtes. En

notant $G = (S, A)$ le graphe précédent avec les mêmes notations, on pose $S' = \{s\} \cup S$ et en notant $n = \sum_{i=0}^{k} b_i 2^i$ sa décomposition binaire, on pose $A' = \{(s, u_{k-i}) \mid b_i = 1\} \cup A$. En effet, le nombre de chemins de s à u_{k+1} est alors :

$$\sum_{i=0}^{k} b_i 2^{k-(k-i)} = n$$

Avec le sommet rajouté, ce graphe a $k + 3 = \lceil \log_2 n \rceil + 2$ sommets.

6. Cette construction est-elle optimale ?

Solution :

Pour $n = 1$, ce graphe n'est pas optimal, car il existe un graphe pointé à un seul sommet qui réalise 1. Pour $n > 1$, montrons que ce graphe est optimal. Montrons qu'un graphe à $k + 2$ sommets a au plus 2^k chemins d'un sommet u à un sommet v. Ce résultat est vrai pour $k = 0$. Supposons le vrai pour $k \in \mathbb{N}$ et soit $G = (S, A)$ un graphe à $k + 3$ sommets. Posons $S = \{u_i \mid i \in [\![0, k + 2]\!]\}$, les sommets étant numérotés par tri topologique, c'est-à-dire tel qu'il n'existe pas de chemin de u_i à u_j si $i > j$ (possible car le graphe est supposé acyclique). Alors u_i peut atteindre $k + 2 - i$ sommets au plus par un chemin, et en sommant en utilisant l'hypothèse de récurrence, on obtient bien le résultat escompté.

Dès lors, un graphe réalisant $n > 0$ doit avoir au moins $\lceil \log_2 n \rceil + 2$ sommets, ce qui montre l'optimalité de la borne.

Définition

Pour la suite de l'exercice, on appelle **automate** un automate fini déterministe $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ sur l'alphabet $\Sigma = \{a, b\}$ où $|F| = 1$.

On dit qu'un automate **réalise** un entier $n \in \mathbb{N}$ si le langage reconnu par $\mathcal{A}$ contient exactement n mots.

7. Montrer que si un automate $\mathcal{A}$ réalise un entier $n \in \mathbb{N}$ avec un nombre minimal d'états, on peut supposer que l'automate est standard, que l'état final n'a pas de transitions sortantes et que tout autre état a exactement 2 transitions sortantes.

Solution :

En considérant l'automate comme émondé, s'il existe une transition sortante depuis l'état final, alors elle est vers un état co-accessible. Cela signifie qu'il existe un cycle dans l'automate et donc une infinité de mots reconnus. Par le même raisonnement, il n'existe aucune transition vers l'état initial, l'automate est donc standard.

L'automate étant déterministe, chaque état a au plus 2 transitions sortantes (l'alphabet est de cardinal 2). Chaque état non final étant co-accessible, il admet au moins une transition sortante.

Supposons qu'un état non final n'ait qu'une seul transition sortante, et montrons qu'il existe un automate réalisant n avec un état de moins :

- si cet état est l'état initial, on le supprime et on considère comme nouvel état initial l'état d'arrivée de cette transition ;
- si cet état q n'est pas l'état initial, notons q' l'arrivée de la transition sortante. On remplace alors chaque $\delta(p, \alpha) = q$ par $\delta(p, \alpha) = q'$.

L'automate ainsi construit vérifie toujours les conditions, avec un état de moins.

8. Montrer que pour tout $n > 0$, il existe un automate qui réalise n dont le nombre d'états est $\lfloor \log_2 n \rfloor$ plus un état par bit égal à 1 dans l'écriture en base 2 de n.

Solution :

On considère la construction suivante :

- si $n = 1$, on considère l'automate à 1 état reconnaissant $\{\varepsilon\}$;
- si n est pair, on construit $\mathcal{A}$ l'automate qui réalise $\frac{n}{2}$, on rajoute un nouvel état initial ayant deux transitions étiquetées par a et b vers l'état initial de $\mathcal{A}$;
- si n est impair, on construit $\mathcal{A}$ l'automate qui réalise $\frac{n-1}{2}$, on rajoute un nouvel état initial et un nouvel état intermédiaire, l'état initial ayant une transition étiquetée par a vers l'état final de $\mathcal{A}$ et une transition étiquetée par b vers l'état intermédiaire, et l'état intermédiaire ayant deux transitions vers l'état initial de $\mathcal{A}$.

Cette construction convient bien pour la réalisation de tout entier n, avec le bon nombre d'états.

Exercice 10 (Dégénérescence de graphe) :

Définition

Un **graphe non orienté** est un couple $G = (S, A)$ où $A \subset \mathcal{P}_2(S)$ est l'ensemble des arêtes.

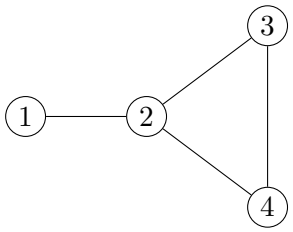
Un **sous-graphe** H de G est un graphe (S', A') où $S' \subset S$ et $A' \subset A$.

Pour $k \in \mathbb{N}$, on dit qu'un graphe non-vide G est **k-dégénéré** si tout sous-graphe non vide H de G contient un sommet de degré au plus k dans H.

La **dégénérescence** de G est le plus petit $k \in \mathbb{N}$ tel que G est k-dégénéré.

Enfin, un sous-graphe (S', A') est dit **induit** si de plus $A' = \mathcal{P}(S') \cap A$.

1. Montrer que le graphe G_0 suivant est 2-dégénéré. Quelle est sa dégénérescence ?



Solution :
Dans ce graphe, le seul sommet de degré 3 est le sommet 2. Tout sous-graphe contenant un des trois autres sommets contient bien un sommet de degré au plus 2. Un sous-graphe qui ne contient aucun des trois est soit vide, soit réduit au sommet 2. Dans un tel sous-graphe, le sommet 2 est de degré 0. La dégénérescence de G_0 est 2, car le sous-graphe induit à $S' = \{2, 3, 4\}$ ne contient aucun sommet de degré 1.

2. On note Δ_G le degré maximal d'un sommet d'un graphe G. Montrer que tout graphe G est Δ_G-dégénéré. Est-il vrai que tout graphe G est de dégénérescence Δ_G ?

Solution :
La première affirmation est triviale (le degré d'un sommet dans un sous-graphe est nécessairement inférieur ou égal au degré de ce sommet dans G). La réciproque est fausse, comme le montre le graphe G_0.

3. Montrer que dans la définition d'un graphe k-dégénéré, il suffit de considérer les sous-graphes induits.

Solution :
Il est clair que si un graphe est k-dégénéré pour la définition initiale, il l'est en ne considérant que les sous-graphes induits. Réciproquement, il suffit de remarquer que le degré d'un sommet dans un sous-graphe est toujours $\leq$ au degré du même sommet dans le sous-graphe induit associé aux mêmes sommets.

4. Déterminer un algorithme naïf pour calculer la dégénérescence d'un graphe. Préciser la complexité.

Solution :
La dégénérescence est exactement :

$$\max_{(S',A') \text{ sous-graphe}} \left(\min_{v \in S'} (\deg(v)) \right)$$

Il suffit alors de faire ce calcul (on peut ne considérer que les sous-graphes induits).
Pour l'écriture du code, on peut envisager d'énumérer tous les sous-ensembles de S par backtracking en utilisant un tableau de booléens (je mets un sommet à `true`, je lance un appel récursif sur la suite, je mets le sommet à `false` et je relance un appel sur la suite). Une fois un sous-ensemble énuméré, on peut calculer le degré minimal en comptant, pour chaque sommet, le nombre de sommets de sa liste d'adjacence qui sont à `true`. La calcul du degré minimal se fait en $\mathcal{O}(|S| + |A|)$ et il y a $2^{|S|}$ sous-graphes possibles.

5. Caractériser les graphes de dégénérescence 1.

Solution :
Ces graphes sont exactement les graphes sans cycle contenant au moins une arête (cela se prouve raisonnablement facilement).

Définition
Un graphe G est dit **régulier** si tous ses sommets ont le même degré.

6. Montrer que tout graphe régulier G est de dégénérescence Δ_G. La réciproque est-elle vraie ? Et si le graphe est connexe ?

Solution :
La dégénérescence d'un graphe est au plus Δ_G (question 2). De plus, si le graphe est régulier, le sous-graphe G lui-même ne contient aucun sommet de degré $< \Delta_G$. La dégénérescence est donc bien Δ_G.
La réciproque est fausse, par exemple en considérant le graphe G_0 dans lequel on supprime l'arête $\{1, 2\}$.
La réciproque est en revanche vraie si on suppose le graphe connexe. En effet, supposons G de dégénérescence Δ_G et soit $H = (S', A')$ un sous-graphe induit non vide, minimal pour l'inclusion, dont le degré minimal est Δ_G. Alors ce sous-graphe est régulier (car Δ_G est aussi le degré maximal), et il est égal à G (car sinon, il existerait une arête absente d'un sommet u de S' à un sommet de $S \setminus S'$, et u serait de degré strictement inférieur à Δ_G dans H).

Définition
Pour $k \in \mathbb{N}$, un k-**arrangement** d'un graphe G est un ordre total sur ses sommets $v_1 < \cdots < v_n$ tel que, pour $i \in [\![1, n]\!]$, on ait $|\{v_j \in S | j < i \text{ et } \{v_i, v_j\} \in A\}| \leq k$.

7. Montrer qu'un graphe a un k-arrangement si et seulement s'il est k-dégénéré.

Solution :
Soit un k-arrangement $v_1 < \cdots < v_n$ de G. Montrons que G est k-dégénéré. Soit H un sous-graphe non vide de G et v le maximum de H pour l'ordre $<$. Alors ses voisins dans H sont $< v$, donc il y en a au plus k, donc H contient bien un sommet de degré $\leq k$.
Réciproquement, montrons par induction sur $|S|$ que si G est k-dégénéré, alors il a un k-arrangement.

- le résultat est vrai si $|S| = 1$, car G est k-dégénéré et admet un k-arrangement pour tout $k \in \mathbb{N}$;
- supposons le résultat vrai pour $|S| \leq n \in \mathbb{N}^*$ et sois G un graphe à $n + 1$ sommets qui est k-dégénéré. G admet donc un sommet v_{n+1} de degré $\leq k$. Soit G' le sous-graphe induit à $S \setminus \{v_{n+1}\}$. Alors G' est toujours k-dégénéré et a n sommet. Soit alors un k-arrangement $v_1 < v_2 \cdots < v_n$ de G' et considérons $v_n < v_{n+1}$. C'est bien un k-arrangement de G.

8. Proposer un algorithme efficace pour calculer la dégénérescence.

Solution :
L'algorithme consiste à construire un k-arrangement. On effectue cela en choisissant le sommet de degré le plus faible, en le mettant à la fin de la liste, et en recommençant en enlevant ce sommet.
Pour ce faire, on commence par construire une liste des degrés des sommets (linéaire). Pour trouver le sommet de degré minimal se fait alors en temps linéaire, de même que la mise à jour des degrés des autres sommets après sa suppression. L'ordre $v_1 < \cdots < v_n$ ainsi produit peut se faire en temps quadratique en n. Il suffit alors de trouver la valeur maximale du degré de v_i dans le sous-graphe induit par $\{v_1, \ldots, v_i\}$, ce qui se fait à nouveau en temps quadratique.
Montrons alors que cet ordre total est optimal. Pour cela, soit un ordre total quelconque $u_1 < \cdots < u_n$ qui réalise la meilleure valeur possible pour k et montrons qu'on peut le modifier pour qu'il soit de la

forme donnée par l'algorithme. Sachant qu'on peut éventuellement éliminer des suffixes qui coïncident, soit $u_1 < \cdots < u_p$ le plus grand préfixe, c'est-à-dire tel que $u_p \neq v_p$ mais $u_i = v_i$ pour $i > p$. Posons $u_q = v_p$. Dans l'ordre $u_1 < \cdots < u_{q-1} < u_{q+1} < \cdots < u_p < u_q$, la valeur de k réalisée n'est pas pire : en effet, le degré de u_q dans le sous-graphe induit par $\{u_1, \ldots, u_p\}$ est $\leq$ à celui de u_p (par définition de l'algorithme) ; de plus, pour chaque autre sommet, il y a un sommet de moins dans la liste des sommets strictement inférieurs. À nouveau, la valeur de k ne peut pas augmenter. On peut réitérer le processus pour retomber sur $v_1 < \cdots < v_n$.

Exercice 11 (Maille d'un graphe) :

Définition

Soit $G = (V, E)$ un graphe non orienté d'ordre ≥ 3. Un **cycle** est un chemin fermé passant au plus une fois par chaque sommet.

On définit, si elle existe, la **maille** de G comme la plus petite longueur d'un cycle de G.

1. Montrer que si G contient au moins un cycle, la maille est bien définie.

Solution :

L'ensemble des longueurs de cycles est minoré par 0 et non vide. Il admet donc un plus petit élément.

2. Soit d le diamètre du graphe (distance maximale entre deux sommets). Montrer que la maille, si elle existe, est majorée par $2d + 1$.

Solution :

Soit un cycle de taille égale à la maille de G. Si la maille est $\geq 2d + 2$, alors il existe x et y deux sommets à distance au moins $d + 1$ dans le cycle. Par définition du diamètre, il existe un chemin de taille $\leq d$ entre x et y. En concaténant ce chemin avec le bon morceau du cycle, on obtient un cycle de taille $\leq 2d + 1$.

Définition

Un graphe $G = (V, E)$ est dit **biparti** si et seulement s'il existe une partition $V = V_1 \sqcup V_2$ telle que toute arête de E relie un sommet de V_1 à un sommet de V_2 (il n'existe pas d'arête entre deux sommets de V_1, ni entre deux sommets de V_2).

3. Montrer que G est biparti si et seulement s'il ne contient pas de cycle de longueur impaire. Qu'en déduire sur la maille d'un tel graphe ?

Solution :

Supposons G biparti. Soit un cycle $v_0 \dots v_n$ un cycle de longueur n, avec $v_n = v_0 \in V_1$. On obtient rapidement que pour i pair, $v_i \in V_1$ et pour i impair, $v_i \in V_2$. On en déduit que n est pair.

Réciproquement, il suffit de montrer que chaque composante connexe est bipartie. On peut donc supposer que G est connexe. Soit alors v_0 un sommet quelconque. On pose V_1 l'ensemble des sommets à distance paire de v_0 et V_2 l'ensemble des sommets à distance impaire de v_0. Ces deux ensembles sont bien disjoints.

Supposons alors qu'il existe $u, v \in V_1$ tels que $\{u, v\} \in E$. Comme $u \in V_1$, il existe un plus court chemin $v_0 u_1 \dots u_{2k+1} = u$ et de même, un plus court chemin $v_0 v_1 \dots v_{2\ell+1} = v$. Soit i le plus grand entier tel que $u_j \neq v_j$ pour $j > i$. Alors $u_i u_{i+1} \dots u_{2k+1} v_{2\ell+1} \dots v_{i+1} v_i$ est un cycle de longueur impaire. On raisonne de même pour V_2 et on conclut par l'absurde.

On en déduit que la maille d'un graphe biparti est un entier pair ≥ 4.

4. Montrer qu'un graphe à n sommets qui contient au plus un cycle contient au plus n arêtes.

Solution :

Par récurrence, si G n'admet pas de cycle, il admet au plus $n - 1$ arêtes.

Si G admet un unique cycle de longueur k, soit G' obtenu à partir de G en enlevant une arête du cycle. Alors G' est sans cycle et comporte n sommets. Il admet donc au plus $n - 1$ arêtes et donc G a au plus n arêtes.

5. Montrer que la maille d'un graphe à n sommets et au moins $n + 1$ arêtes est majorée par $\left\lfloor \frac{2(n+1)}{3} \right\rfloor$.

Solution :

Par la question précédente, on sait qu'il existe au moins deux cycles distincts C_1 et C_2. Notons k le nombre d'arêtes communes aux deux cycles, n_1 le nombre d'arêtes de C_1 hors de C_2 et n_2 le nombre d'arêtes de C_2 hors de C_1.

- Si $k = 0$, alors la maille m vérifie $2m \leq n_1 + n_2 \leq n + 1$ et donc on a :

$$m \leq \frac{n+1}{2} \leq \frac{2(n+1)}{3}$$

- Sinon, on a $m \leq n_1 + k$ et $m \leq n_2 + k$. On peut alors construire un troisième cycle avec des portions de C_1 et C_2 qui ne sont pas communes. Il est de longueur au plus $n_1 + n_2$. On a donc $m \leq n_1 + n_2$. En sommant les trois inégalités, on obtient :

$$3m \leq 2(n_1 + n_2 + k) \leq 2(n + 1)$$

6. Soit G un graphe tel que le plus petit degré δ d'un sommet et la maille m soient supérieurs ou égaux à 3. Montrer que si m est pair, le nombre de sommets est alors minoré par $\frac{2}{\delta - 2}\left((\delta - 1)^{\frac{m}{2}} - 1\right)$.

Solution :
Soit u et v deux sommets voisins dans un cycle et soit w un sommet à distance $k \leq \frac{m}{2} - 1$ de la paire $\{u, v\}$. Il existe alors un unique chemin n'utilisant pas l'arête $\{u, v\}$ joignant u ou v à w de longueur au plus $\frac{m}{2} - 1$ (sinon, la maille serait strictement inférieure à m). Par ailleurs, z admet au moins $\delta - 1$ voisins à distance $k + 1$, car δ est le degré minimum. Enfin, ces voisins à distance $k + 1$ pour tous les z à distance k sont distincts (sinon, cela créerait un cycle de taille $< m$). On en déduit qu'il existe au moins $2(\delta - 1)^d$ sommets à distance k de $\{x, y\}$.
Finalement, on en déduit que le nombre de sommets est minoré par $\sum_{k=1}^{m/2-1} 2(\delta - 1)^d$ qui permet bien de retrouver le résultat.

Exercice 12 (Théorème de Turán) :

1. Déterminer une relation entre le nombre d'arêtes d'un graphe et les degrés de ses sommets.

Solution :
On montre facilement que $\sum_{v \in S} \deg(v) = 2|A|$.

Définition

Un **triangle** dans un graphe $G = (S, A)$ est un ensemble de trois sommets reliés.

Un graphe $G = (S, A)$ est dit **biparti** si et seulement s'il existe une partition $S = S_1 \sqcup S_2$ telle que toute arête de A relie un sommet de S_1 à un sommet de S_2 (il n'existe pas d'arête entre deux sommets de S_1, ni entre deux sommets de S_2). Si, de plus, chaque sommet de S_1 est relié à chaque sommet de S_2, le graphe est dit **biparti complet**.

On étend de manière similaire la définition pour un graphe p-**parti**.

2. Montrer que pour tout $n \in \mathbb{N}$, il existe un graphe sans triangle à n sommets et $\left\lfloor \frac{n^2}{4} \right\rfloor$ arêtes.

Solution :
Soit $n \in \mathbb{N}$ et S_1, S_2 deux ensembles disjoints de cardinaux respectifs $\left\lfloor \frac{n}{2} \right\rfloor$ et $\left\lceil \frac{n}{2} \right\rceil$. Soit G le graphe biparti complet associé. Ce graphe possède bien n sommets et $m = |S_1||S_2|$ arêtes. Si n est pair, la condition est bien vérifiée, sinon $|S_1||S_2| = \frac{n^2-1}{4} = \left\lfloor \frac{n^2}{4} \right\rfloor$.

On associe un poids $p(x) \in [0, 1]$ à chaque sommet x d'un graphe G, de telle sorte que $\sum_{x \in S} p(x) = 1$. Le poids d'une arête $\{x, y\}$ est alors $p(x)p(y)$. Le poids $p(G)$ du graphe G est la somme des poids de ses arêtes.

3. Calculer $p(G)$ lorsque tous les sommets ont le même poids.

Solution :
Soit n le nombre de sommet. Le poids d'un sommet est alors $\frac{1}{n}$. Le poids d'une arête est donc $\frac{1}{n^2}$. Le poids de G est donc $\frac{|A|}{n^2}$.

4. Montrer que si G est sans triangle, alors $p(G) \leq \frac{1}{4}$.

Solution :
Soit G sans triangle fixé. Par compacité et continuité de la fonction poids, le poids maximum est atteint pour une certaine pondération p^*.
Supposons qu'il existe deux sommets u, v de poids non nul, et non reliés par une arête. Soit alors f la fonction qui à $t \in [-p^*(u), p^*(v)]$ associe le poids $p_t(G)$ où $p_t : x \mapsto \begin{cases} p^*(u) + t & \text{si } x = u \\ p^*(v) - t & \text{si } x = v \\ p^*(x) & \text{sinon} \end{cases}$.
f est une fonction affine, car il n'y a pas d'arête entre u et v. De plus, f atteint un maximum en $0 \in [-p^*(u), p^*(v)]$ par hypothèse. Elle est donc de pente nulle. Enfin, la pondération associée à $t = p^*(v)$ est une pondération de poids maximal, qui contient un sommet de poids nul de plus que p^*.
En itérant ce procédé, on en conclut qu'on peut choisir p^* de telle sorte que l'ensemble des sommets de poids non nul est une clique. G étant sans triangle, cette clique est soit de taille 1 ($p^*(G) = 0$), soit de taille 2 ($p^*(G) = ab \leq \left(\frac{a+b}{2}\right)^2 = \frac{1}{4}$ par l'inégalité arithmético-géométrique).

5. En déduire qu'un graphe à n sommets sans triangle possède au plus $\left\lfloor \frac{n^2}{4} \right\rfloor$ arêtes.

Solution :
On utilise les deux questions précédentes.

6. Montrer qu'un graphe sans triangle à n sommets et $\left\lfloor \frac{n^2}{4} \right\rfloor$ arêtes est nécessairement un graphe biparti complet dont les cardinaux des deux parties diffèrent d'au plus 1.

Solution :
Par récurrence forte sur n.

- Le résultat est trivial pour $n = 1$. Pour $n = 2$, on obtient un graphe avec une seule arête qui vérifie les conditions.
- Supposons le résultat vrai pour tout entier $< n$, où $n \in \mathbb{N}$, $n \geq 3$. Soit G un graphe vérifiant les conditions de l'énoncé. Si $\{u, v\} \in A$, G étant sans triangle, il y a au plus $n-2$ arêtes incidentes à $\{u, v\}$ (chacun des $n-2$ autres sommets ne peut pas être simultanément voisins de u et v). Notons alors G' le graphe obtenu par la suppression de u, v et des arêtes correspondantes. Ce graphe est sans triangle et possède $n-2$ sommets. On a :

$$|A'| \geq \left\lfloor \frac{n^2}{4} \right\rfloor - 1 - (n-2) = \left\lfloor \frac{(n-2)^2}{4} \right\rfloor$$

  D'après la majoration de la question précédente, il en possède donc exactement ce nombre et il y avait donc $n-2$ arêtes incidentes à $\{u, v\}$ dans G. Par hypothèse de récurrence, G' est un graphe biparti complet dont les cardinaux des deux parties diffèrent d'au plus 1. L'ensemble des arêtes incidentes à u (resp. v) touche au plus une des parties de G'. Il suffit d'ajouter u et v à la bonne partie pour conclure.

7. Montrer qu'un graphe p-parti ne contient pas de clique (sous-graphe complet) de taille $p+1$.

Solution :
Soit $S = S_1 \sqcup S_2 \sqcup \cdots \sqcup S_p$ une partition du graphe G p-parti. Si on considère $p+1$ sommets de S, par le principe des tiroirs, deux d'entre eux sont contenus dans le même S_i, donc ne peuvent pas être reliés par une arête.

8. Montrer qu'un graphe à n sommets ne contenant pas de clique de taille $p+1$ admet au plus $\left\lfloor \frac{(p-1)n^2}{2p} \right\rfloor$ arêtes.

Solution :
En reprenant les notations de la question 4, on sait que l'ensemble des sommets de poids non nul pour p^* est une clique. Notons k la taille de cette clique, et $a_1, \ldots, a_k$ les poids des sommets de la clique (de somme 1).

On a $2 \sum_{1 \leq i < j \leq k} a_i a_j = \left(\sum_{1 \leq i \leq k} a_i \right)^2 - \sum_{1 \leq i \leq k} a_i^2 = 1 - \sum_{1 \leq i \leq k} a_i^2$.

De plus, par Cauchy-Schwarz, $\sum_{1 \leq i \leq k} a_i^2 = \frac{1}{k} \sum_{1 \leq i \leq k} a_i^2 \sum_{1 \leq i \leq k} 1 \geq \frac{1}{k} \left(\sum_{1 \leq i \leq k} a_i \right)^2 = \frac{1}{k}$.

On obtient donc avec les deux expressions $p^*(G) \leq \frac{p-1}{2p}$, puis par la question 3 : $|A| \leq \left\lfloor \frac{(p-1)n^2}{2p} \right\rfloor$.

9. Caractériser les graphes atteignant cette borne.

Solution :
Montrons que les graphes sans clique de taille $p+1$ atteignant cette borne sont exactement les graphes p-partis complets équilibrés (c'est-à-dire dont les cardinaux des parties diffèrent d'au plus 1). On traite par récurrence forte.

- Pour $n \leq p$, le résultat est immédiat.
- Soit $n > p$ tel que la propriété est vraie pour tout $k < n$, et soit G un graphe de taille n vérifiant les conditions de la questions précédente, atteignant la borne du nombre d'arêtes. Il est clair que G possède une clique de taille p (sinon on pourrait lui ajouter une arête sans créer de clique de taille $p+1$.

  Soit X une telle clique. Comme précédemment, chaque sommet de $S \setminus X$ est adjacent à au plus $p-1$ sommets de X, donc le nombre d'arêtes non comprises dans $S \setminus X$ est inférieur ou égal à : $\frac{p(p-1)}{2} + (n-p)(p-1) = \frac{p-1}{2p}(2pn - p^2)$.

  Ainsi, dans G' où on a supprimé X, le nombre d'arêtes est supérieur ou égal à (en simplifiant) $\left\lfloor \frac{(p-1)(n-p)^2}{2p} \right\rfloor$. Comme précédemment, c'est le nombre exact d'arêtes, donc par hypothèse de récurrence, il existe G' un graphe p-parti équilibré. Soit k le nombre de parties non vides de la partition $S = S_1 \sqcup S_2 \sqcup \cdots \sqcup S_p$ (qui seront $S_1, \ldots, S_k$), et soit $v_i \in S_i$ pour $i \in [\![1, k]\!]$. Chaque sommet y_i est voisin de tous les sommets de X, sauf un, noté $\sigma(y_i)$. On remarque que σ est injective, car si $\sigma(y_i) = \sigma(y_j)$, alors l'ensemble $\{y_i, y_j\} \cup X \setminus \{\sigma(y_i)\}$ est une clique de G de cardinal au plus p, soit $y_i = y_j$.

  En ajoutant chaque sommet $\sigma(y_i)$ à S_i, on crée une partition de G vérifiant les conditions(en complétant par des $S_{k+1}, \ldots, S_p$ pour les sommets restant de X, si $k < p$). En effet, $\sigma(y_i)$ n'est voisin d'aucun sommet de S_i.

Réciproquement, soit G un graphe p-parti équilibré à $n > p$ sommets. On peut retirer une clique de taille p de sorte à faire disparaitre au plus $\frac{p(p-1)}{2} + (n-p)(p-1)$ arêtes et se ramener à un graphe p-parti équilibré à $n-p$ sommets. On conclut par récurrence.

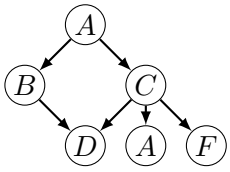
Exercice 13 (Étiquetage d'accessibilité) :

Définition

Soit $G = (S, A)$ un graphe orienté. Un chemin dans G est une suite finie $v_1 \dots v_n$ d'éléments de S telle que $\forall 1 \leq i < n$, on a $(v_i, v_{i+1}) \in A$. Pour $u, v \in S$, on écrit $u \rightsquigarrow v$ quand il existe un chemin de u à v.

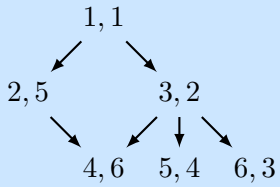
Un **étiquetage** de G est une fonction $f : V \longrightarrow \mathbb{N}^2$. Étant donnés $t_1 = (p_1, q_1)$ et $t_2 = (p_2, q_2)$ des éléments de $\mathbb{N}^2$, on écrit $t_1 \leq t_2$ lorsque $p_1 \leq p_2$ et $q_1 \leq q_2$. Un étiquetage f est appelé **étiquetage d'accessibilité** s'il satisfait la propriété : $\forall u \neq v \in S, u \rightsquigarrow v \Leftrightarrow f(u) \leq f(v)$.

1. Construire un étiquetage d'accessibilité pour le graphe suivant :



Solution :

Une solution possible est :



2. Rappeler la définition de composante connexes et composante fortement connexe (CFC) d'un graphe orienté G et la reformuler en utilisant $\rightsquigarrow$.

Solution :

Les composantes connexes sont les classes d'équivalence de la clôture symétrique, réflexive et transitive de $\rightsquigarrow$.

Les CFC sont les classes d'équivalence de la relation d'équivalence définie par $u \rightsquigarrow v \wedge v \rightsquigarrow u$.

Définition

Soit $G = (S, A)$ un graphe orienté. On défini le **graphe des CFC** de G comme le graphe orienté $G_C = (S_C, A_C)$ où S_C est l'ensemble des CFC de G et où A_C est défini par :

$$A_C := \{(K_1, K_2) \in S_C \times S_C \mid (K_1 \neq K_2) \wedge (\exists v_1 \in K_1, v_2 \in K_2 \text{ tels que } v_1 \rightsquigarrow v_2)\}$$

3. Expliquer pourquoi il n'existe pas de cycle dans G_C.
Montrer que G admet un étiquetage d'accessibilité si et seulement si G_C admet un étiquetage d'accessibilité.

Solution :

Si on suppose qu'il existe un cycle dans G_C, par exemple $K_1, \dots, K_n$, alors pour tous i, j, pour tous $u_i, u_j \in K_i \times K_j$, $u_i \rightsquigarrow u_j$ dans G, et donc tous les éléments des K_i sont dans la même CFC.

Supposons que G admet un étiquetage d'accessibilité, noté f. Soit $\phi : S_C \to S$ qui à une CFC associe un représentant de la CFC. Soit $f_C = f \circ \phi$. On observe facilement que f_C est un étiquetage d'accessibilité.

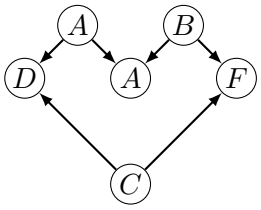
Réciproquement, soit f_C un étiquetage d'accessibilité de G_C. On pose $f : v \mapsto f_C(K_v)$ où K_v est la CFC de v. À nouveau, on vérifie facilement les hypothèses.

4. Montrer que G admet un étiquetage d'accessibilité si et seulement si chacune de ses composantes connexes admet un étiquetage d'accessibilité.

Solution :
Si G admet un étiquetage d'accessibilité, il est clair que c'est le cas pour chacune de ses CC (en considérant la restriction).
Supposons que chaque CC de G possède un étiquetage d'accessibilité. Procédons par récurrence sur le nombre k de CC.

- $k = 1$ est trivial (c'est G lui-même).
- Supposons le résultat vrai pour tout graphe avec strictement moins de $k \in \mathbb{N}$ CC, $k \geq 2$. Partitionnons G en G_1 et G_2 en regroupant des CC dans l'une ou l'autre des deux parties. Chacun des G_1, G_2 possèdent moins de k CC, donc par HR, il existe $f_1 = (p_1, q_1)$ et $f_2 = (p_2, q_2)$ des étiquetages d'accessibilité pour ces deux sous-graphes. Choisissons M plus grand que tous les entiers apparaissant dans les images de f_1 et f_2, et posons $f : v \mapsto \begin{cases} (p_1(v) + M, q_1(v)) & \text{si } v \in S_1 \\ (p_2(v), q_2(v) + M) & \text{sinon} \end{cases}$.
  Alors f est bien un étiquetage d'accessibilité pour G.

5. Montrer que le graphe suivant d'admet pas d'étiquetage d'accessibilité :



Solution :
On dit que $u, v \in S$ sont incomparables s'il n'existe pas de chemin de l'un à l'autre. Dans ce graphe, les couples incomparables sont (A, B), (A, C), (A, F), (B, C), (B, D), (C, A), (D, A), (D, F), (A, F).
Procédons par l'absurde et supposons $f = (p, q)$ un étiquetage d'accessibilité. Pour deux sommets incomparables, on a forcément l'une des situations :

- $p(u) < p(v)$ et $q(u) > q(v)$, qu'on notera $u \lhd v$;
- $u \lhd v$.

Notons alors A, B, C par u_1, u_2 et u_3, et D, A, F par v_1, v_2, v_3. On a que (u_i, v_i) est incomparable. Deux cas sont alors possibles : soit $u_i \lhd v_i$ pour deux valeurs différentes de $i \in \{1, 2, 3\}$, soit $v_i \lhd u_i$ pour deux valeurs différentes de $i \in \{1, 2, 3\}$. Dans le premier cas, soient $i \neq j$ tels que $u_i \lhd v_i$ et $u_j \lhd v_j$. Comme v_i et v_j sont incomparables, on a soit $v_i \lhd v_j$, soit $v_j \lhd v_i$. Dans le premier sous-cas, on a donc $q(v_i) > q(v_j)$ et par transitivité $q(u_i) > q(v_j)$ mais ceci contredit le fait que $f(u_i) \leq f(v_j)$ puisque $u_i \rightsquigarrow v_j$. Tous les autres cas se traitent de manière similaire.

Définition
On dit que deux sommets $u \neq v$ d'un graphe acyclique G sont **comparables** si $u \rightsquigarrow v$ ou $v \rightsquigarrow u$.
On dit qu'un graphe orienté $G' = (S, A')$ est **conjugué** à $G = (S, A)$ s'il est acyclique et si, pour tous $u, v \in S$, si $u \neq v$, alors il y a exactement un graphe parmi G et G' dans lequel u et v sont comparables.

6. Montrer que si G a un conjugué G', alors G admet un étiquetage d'accessibilité. Expliquer comment le calculer à partir de G'.
   Indication : On montrera que $(S, A \cup A')$ est acyclique.

Solution :
Montrons que $(S, A \cup A')$ est acyclique. Supposons par l'absurde qu'il possède un cycle. Comme G et G' sont acycliques, ce cycle doit passer par des arêtes de G et de G'. En considérant le chemin franchi par le cycle dans G et dans G', on peut écrire le cycle comme $v_1 \dots v_n$ où $n \geq 2$ est impair, et si i est impair, alors $v_i \rightsquigarrow v_{i+1}$ dans G et si i est pair, $v_i \rightsquigarrow v_{i+1}$ dans G' (ces cas sont exclusifs par définition du conjugué). Choisissons le cycle qui minimise n.

Tout d'abord, v_1 et v_2 sont comparables dans G et si $v_3 = v_1$, alors v_1 et v_2 sont comparables dans G', ce qui est absurde. On en déduit que $v_1 \neq v_3$ et que $n \geq 5$. Ainsi, soit v_1 et v_3 sont comparables dans G, soit dans G'.

- Dans le premier cas, on a forcément $v_1 \rightsquigarrow v_3$ dans G (sinon, on aurait $v_3 \rightsquigarrow v_3$ dans G), mais comme $v_3 \rightsquigarrow v_4$ dans G, alors $v_1 \rightsquigarrow v_4$ dans G. Ainsi, $v_1 v_4 \dots v_n$ est un cycle a moins d'alternances qu'initialement (absurde).
- Dans le second cas, on montre de même que $v_{n-1} v_3 \dots v_n$ est un cycle avec moins d'alternances.

Montrons à présent que si G a un conjugué G', alors G admet un étiquetage d'accessibilité. Notons G'_R le graphe obtenu à partir de G' en retournant les arêtes. Par symétrie, ce graphe est toujours acyclique et il est clair que c'est également un conjugué de G. Ainsi, les graphes orientés $G'' = (S, A \cup A')$ et $G''_R = (S, A \cup A'_R)$ sont tous deux acycliques.
Soit alors $p : V \to \mathbb{N}$ une fonction injective telle que si $u \rightsquigarrow v$ dans G'', alors $p(u) < p(v)$ (existe car le graphe est acyclique). On construire de même q pour G''_R et on pose $f = (p, q)$.
On montre alors que f est bien un étiquetage d'accessibilité. Si $u \rightsquigarrow v$ dans G, alors dans G'' et G''_R également, et donc on a bien $f(u) \leq f(v)$. Sinon, $u \rightsquigarrow v$ dans G' et $v \rightsquigarrow u$ dans G'_R (ou l'inverse) et donc on a ni $f(u) \leq f(v)$, ni $f(v) \leq f(u)$.

7. Montrer que, réciproquement, si G est acyclique et admet un étiquetage d'accessibilité, alors il admet un conjugué.

Solution :
Soit $f = (p, q)$ un étiquetage d'accessibilité de G. Soit $G' = (S, A')$ tel que pour $u \neq v$, $(u, v) \in A'$ si et seulement si u et v sont incomparables dans G et $p(u) < p(v)$. Montrons que G' est bien un conjugué de G, c'est-à-dire qu'il est acyclique et une paire est comparables dans G' si et seulement si elle est incomparable dans G.
Il est clair que G' est acyclique (car $p(u) < p(v)$ pour toute arête de G). L'implication retour découle de la définition de A'. Il ne reste que l'implication directe :
Soient $u \rightsquigarrow v$ deux sommets comparables dans G'. Soit $u = w_1 \dots w_n = v$ un tel chemin. Par définition de G', on a forcément $q(w_i) > q(w_{i+1}$ (car w_i et w_{i+1} sont incomparables dans G) par transitivité, $q(u) > q(v)$, et donc u et v sont incomparables dans G (car $p(u) < p(v)$).

Exercice 14 (Liste-coloration de graphes) :

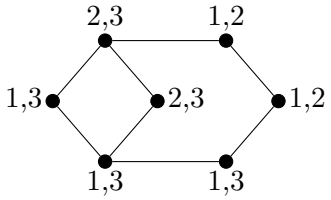
Définition

Soit $G = (S, A)$ un graphe non-orienté simple. Une **coloration propre** de G est une fonction $\phi : S \to \mathbb{N}$ telle que si $\{u, v\} \in A$, alors $\phi(u) \neq \phi(v)$ (les images seront appelées **couleurs**). G est dit k-**coloriable** s'il existe une coloration propre ϕ telle que $|\phi(S)| = k$.

Pour chaque sommet $v \in S$ et $L(v) \subset \mathbb{N}$ (appelée **liste de couleurs**), G est dit **liste-coloriable** (pour L) s'il existe une coloration propre ϕ de G telle que $\forall v \in S, \phi(v) \subset L(v)$.

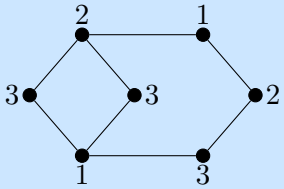
G est dit k-**choisissable** si pour toute famille de listes $(L(v))_{v \in S}$, toutes de longueur au moins k, alors (G, L) est liste-coloriable.

1. Déterminer si l'instance suivante est liste-coloriable.



Solution :

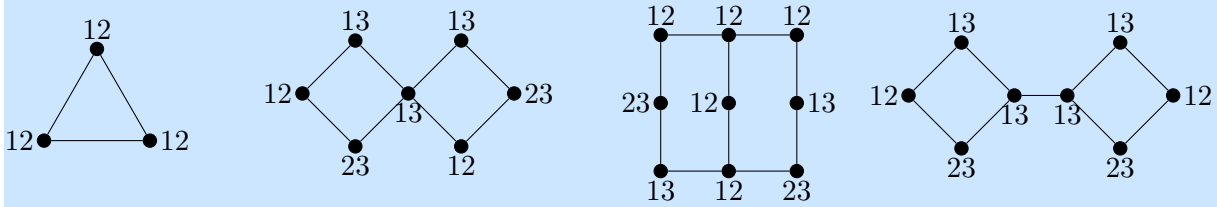
Oui, par exemple :



2. Montrer que les graphes suivants ne sont pas 2-choisissables.

Solution :

On exhibe des contre-exemples :



Définition

Un graphe $G = (S, A)$ est dit k-**dégénéré** s'il est vide ou s'il existe un sommet $v \in S$ de degré au plus k tel que G privé de v et des arêtes reliées à v est k-dégénéré.

3. Montrer que si G est k-dégénéré, alors G est $k+1$-choisissable.

Solution :

On montre par induction sur la structure de graphe k-dégénéré :

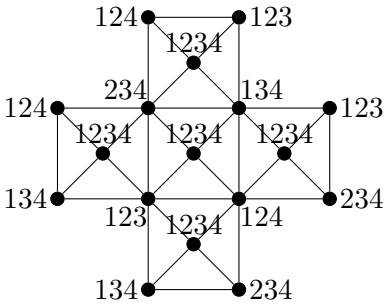
- Si le graphe est vide, c'est direct.
- Soit $n \in \mathbb{N}^*$ fixé. Supposons la propriété vraie pour tout graphe possédant strictement moins de n sommets. Soit G un graphe à n sommets qui est k-dégénéré. Par définition, il existe un

sommet $v \in S$ de degré $\leq k$ tel que $G' = G \setminus v$ est k-dégénéré, donc $k+1$-choisissable par hypothèse. Ainsi, quelle que soit la famille de listes L' toutes au moins de longueur $k+1$, (G', L') sera liste-coloriable. Ainsi, pour n'importe quelle liste $L(v)$ de taille $k+1$, $(G, L' \cup L(v))$ sera liste-coloriable, car le degré de v étant k, il reste toujours un choix pour colorier v après avoir choisi les couleurs de tous les autres sommets.

Définition

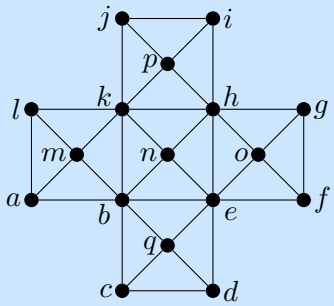
Un graphe est dit **planaire** si on peut le dessiner dans le plan sans que deux arêtes ne se croisent, sauf éventuellement en une extrémité commune. Une **face** d'un graphe planaire dessiné est une composante connexe de $\mathsf{R}^2 \setminus \mathrm{im}(G)$.

4. Le graphe ci-dessous est planaire. On a représenté la liste sans syntaxe pour éviter de surcharger le dessin. Montrer qu'il n'est pas liste-coloriable.



Solution :

Notons les sommets de la façon suivante :



Le problème étant symétrique, supposons qu'on colorie le sommet b par 1. Supposons alors qu'on numérote le sommet e par 2. Cela implique que les sommets c, d et q doivent être coloriés par 3 ou 4. C'est absurde. On en déduit que nécessairement, le sommet e est numéroté 4. En raisonnant de même pour les sommets h et k, on en déduit qu'ils doivent être coloriés par 3 et 2 respectivement. Mais alors, on ne peut pas colorier le sommet n.

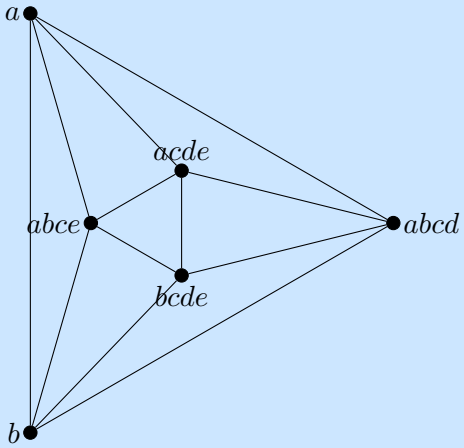
On raisonne de même si le sommet b est colorié par 2 ou 3.

5. Montrer qu'il existe des graphes planaires 3-coloriables qui ne sont pas 4-choisissables.

Solution :

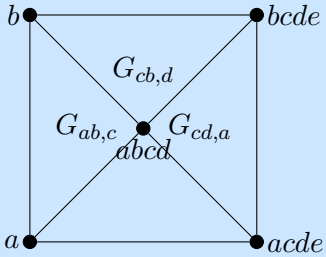
Le contre-exemple possédant beaucoup de sommets, nous allons le construire sous-graphe par sous-graphe.

Commençons par un gadget, que nous nommerons $G_{ab,c}$: il permet, avec des listes de taille ≥ 4, d'imposer la couleur c à un sommet, si deux autres sommets ont les couleurs a et b respectivement. Supposons disposer de 5 couleurs pour les listes, notées (dans le désordre) a, b, c, d et e. Le gadget est le suivant :



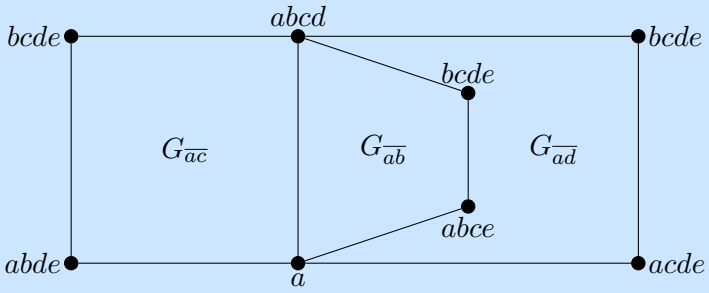
En effet, si on choisit la lettre d au troisième point du grand triangle, alors les 3 points du petit triangle doivent nécessairement choisir leurs couleurs parmi c ou e, ce qui est absurde.
Notons qu'il y a liberté du sommet d parmi les 2 disponibles. Notons de plus que ce gadget est 3-coloriable. Il possède 6 sommets.

Avec ce gadget, on peut alors construire un sous-graphe qui impose de ne pas choisir la couleur b en un point si on a choisi la couleur a en un autre point. Le sous-graphe, qu'on notera $G_{\overline{ab}}$ est le suivant :



En effet, si on choisit la lettre b en haut à gauche, alors on doit nécessairement choisir c au centre, puis d en haut à droite, puis a en bas à droite, ce qui est impossible si on a choisi a en bas à gauche.
Notons de plus que ce sous-graphe est 3-coloriable. Il possède 14 sommets.

Avec ce sous-graphe, on peut alors empêcher le choix de la couleur a en un sommet, en le répétant 3 fois :



En effet, on utilise 3 fois le sous-graphe précédent, pour chaque autre couleur que a sur le sommet en haut au centre. À nouveau, il n'y a pas de problème pour 3-colorier.
Ce morceau possède 38 sommets.

On termine en répétant ce morceau 4 fois, une fois pour chaque couleur du sommet central, à qui on attribuera la couleur 1234. On obtient un graphe à 149 sommets. Il est bien 3-coloriable, planaire, mais cette instance n'est pas liste-coloriable.

6. Montrer que pour $k \in \mathbb{N}$, il existe un graphe biparti non k-choisissable.

Solution :
Pour un nombre n fixé, on pose $K_{n,n}$ le graphe biparti complet à $2n$ sommets (chacune des deux parties possède n sommets, et chaque sommet d'une partie est relié à chaque sommet de l'autre partie). Dès lors, pour k fixé, posons $n = \binom{2k}{k}$ et $G = K_{n,n}$. Construisons alors une instance de famille de listes telle que (G, L) n'est pas liste-coloriable :
À chacune des deux parties, on attribue les $\binom{2k}{k}$ listes à k éléments parmi $[\![1, 2k]\!]$. Le graphe étant biparti complet, il n'est pas difficile de voir que si une couleur est utilisée d'un côté, alors elle ne peut pas être utilisée de l'autre. De plus, si on n'utilise que k couleurs pour un côté, il existe un sommet dont la liste est constituée des k autres couleurs, donc qui ne peut pas être colorié. Ainsi, on a besoin d'au moins $k+1$ couleurs par côté, ce qui est absurde, car on possède $2k$ couleurs au total. Cette instance n'est pas liste-coloriable.

Exercice 15 (Ensembles inévitables) :

Définition

Dans tout le sujet, on fixe $\Sigma = \{a, b\}$ un alphabet. Pour $u \in \Sigma^*$, on écrit $u = u_1 \dots u_n$ où $n = |u|$ est la **longueur** de u. On dit qu'un mot $u \in \Sigma^*$ **évite** un mot $v \in \Sigma^*$ si v n'apparait **pas** comme facteur de u. On dit que u **évite** un ensemble de mots $V \subset \Sigma^*$ s'il évite chaque mot de V.

On dit qu'un ensemble $V \subset \Sigma^*$ est **inévitable** s'il n'existe qu'un nombre fini de mots $u \in \Sigma^*$ qui évitent V. Sinon, il est dit **évitable**.

1. Déterminer un mot de longueur au moins 12 qui évite $V = \{aaaa, aaab, aba, baaa, bab, bbbb\}$.

Solution :
Le mot $aabbaabbaabb$ convient.

2. Donner le pseudo-code d'un algorithme naïf qui détermine, étant donné un mot $u \in \Sigma^*$ et un ensemble fini $V \subset \Sigma^*$, si u évite V.
Déterminer sa complexité en temps et en espace quand tous les mots de V font la même longueur.

Solution :
On propose l'algorithme suivant :

Algorithme : Langage évitable
Données : un mot u, un tableau V de longueur m de mots, un tableau nV des longueurs des mots de V
Résultat : Est-ce que u évite V ?

```
pour i = 1 à m faire
    pour j = 1 à nV[i] faire
        k ← 1 ;
        tant que k ≤ nV[i] ∧ u[j + k] = V[i][k] faire
            k ← k + 1 ;
        si k > nV[i] alors
            retourner Faux ;
retourner Vrai ;
```

Si les m mots de V sont de taille au plus ℓ, alors la complexité totale est en $\mathcal{O}(n \times m \times \ell)$.

3. L'ensemble de la question 1 est-il inévitable ? L'ensemble $\{aaa, abb, baa, abab\}$ est-il inévitable ?

Solution :
On voit que pour tout $k \in \mathbb{N}$, $(aabb)^k$ évite V qui est donc évitable. Pour le deuxième ensemble, tous les mots de b^+ évitent V.

4. Montrer que $\{aaa, bab, baab, bbb\}$ est inévitable.

Solution :
On procède par l'absurde. Comme V contient aaa et bbb, un mot suffisamment long u qui évite V doit contenir un facteur ba. Ce facteur doit alors être suivi d'un a ou d'un b. Comme $bab \in V$, on en déduit que u contient un facteur baa. Mais alors ce facteur ne peut être suivi ni de a, ni de b, sinon aaa ou $baab$ seraient des facteurs de u.

5. Montrer le lemme suivant : pour tout $k \in \mathbb{N}$, il existe $n \in \mathbb{N}$ tel que pour tout mot $u \in \Sigma^*$ tel que $|w| > n$, il existe deux entiers $p < q$ tels que pour tout entier $\ell \in [\![0, k-1]\!]$, $u_{p+\ell} = u_{q+\ell}$.

Solution :
On pose $n = |\Sigma|^k + k$. Un mot de taille $> n$ mot a strictement plus de $|\Sigma|^k$ positions où commence un facteur de taille k. Comme il existe au plus $|\Sigma|^k$ facteurs de taille k, par le principe des tiroirs, on en déduit l'existence de deux positions où commence un même facteur de taille k.

6. En déduire un algorithme naïf qui, étant donné un ensemble fini $V \subset \Sigma^*$, détermine si V est inévitable. Déterminer sa complexité en temps et en espace.

Solution :

Soit k la longueur maximale d'un mot de V. Montrons que V est évitable si et seulement s'il existe un mot évitant V avec deux occurrences distinctes d'un même facteur de longueur k. Pour le sens direct, il existe une infinité de mots évitant V, en particulier un mot de taille $> n$ du lemme précédent qui permet de conclure. Pour le sens réciproque, soit u un mot évitant V avec deux occurrences d'un mot w de taille k. Sans perte de généralité, on peut supposer que u se termine à la fin de la deuxième occurrence de w. Soit $d > 0$ le décalage entre les deux occurrences de w. On peut alors construire des mots arbitrairement grand en rajoutant une occurrence de w commençant d positions après le début de la dernière occurrence de w, donc V est évitable.

Ainsi, V est inévitable si et seulement si tous les mots de longueur $> n$ ont un facteur dans V, n étant défini à la question précédente. On en déduit que V est inévitable si et seulement si tous les mots de longueur $n+1$ ont un facteur dans V. Le sens direct utilise l'équivalence précédente. Pour le sens réciproque, si tout mot de longueur $n+1$ a un facteur dans V, alors c'est également le cas pour tout mot de longueur $> n+1$ (on rajoute des lettres).

Ainsi, il suffit d'énumérer tous les mots de longueur $n+1$ et de tester s'il en existe un qui évite V pour répondre à la question. Comme $n = |\Sigma|^k + k$, et qu'il existe $|\Sigma|^{n+1}$ mots de taille $n+1$, on en déduit, en notant k la taille maximale d'un mot de V et m le nombre de mots de V que la complexité temporelle est $\mathcal{O}(mk4^k 2^{2^k})$. La complexité spatiale nécessite le stockage d'un mot de longueur $n+1$, soit $\mathcal{O}(2^k)$.

7. Déterminer une méthode permettant de décider si un langage rationnel est inévitable.

Solution :

La question revient à savoir si l'ensemble des mots qui évitent L est fini ou non, c'est-à-dire si $\Sigma^* \setminus \Sigma^* L \Sigma^*$ est fini ou non. À partir d'un automate reconnaissant L, on peut construire un automate reconnaissant $\Sigma^* L \Sigma^*$ (en rajoutant 2 états et des transitions). On déterminise ensuite cet automate. On peut alors construire un automate reconnaissant $\Sigma^* \setminus \Sigma^* L \Sigma^*$ (on complète l'automate et on inverse état final et non final). Enfin, on émonde (suppression des états non accessibles et non co-accessibles) l'automate ainsi obtenu et il suffit de vérifier l'existence de cycles. En termes de complexité, les étapes sont coûteuses à partir de la déterminisation qui peut former un automate de taille exponentielle en la taille de l'automate initial.

Définition

Soit L un langage rationnel et $V \subset \Sigma^*$. On dit que V est **inévitable pour** L s'il n'existe qu'un nombre fini de mots de L qui évitent V.

8. Soit L un langage rationnel et $V \subset \Sigma^*$ un ensemble fini. Déterminer une méthode permettant de décider si V est inévitable pour L.

Solution :

Comme précédemment, cela revient à décider si $L \cap (\Sigma^* \setminus \Sigma^* V \Sigma^*)$ est fini ou non. Il est possible de construire un automate pour ce langage, en rajoutant l'étape d'intersection (automate produit).

9. Montrer que pour tout ensemble inévitable $V \subset \Sigma^*$, il existe $V' \subset V$ tel que V' est fini et inévitable.

Solution :

Soit V inévitable. Soit n la longueur maximale d'un mot qui évite V. Ainsi, tout mot de longueur $n+1$ a un facteur dans V. Soit V' l'ensemble de ces facteurs. C'est un ensemble fini et inévitable, d'après une propriété prouvée précédemment.

10. Soit L un langage rationnel inévitable. Déterminer une méthode pour calculer un sous-ensemble de L fini et inévitable.

Solution :
Il suffit de suivre la méthode précédente : si $L \cap (\Sigma^* \setminus \Sigma^* V \Sigma^*)$ est fini, on peut déterminer la longueur maximale n d'un de ses mots. On peut alors générer tous les mots de longueur $n+1$ et ne garder que les facteurs qui sont dans V.

Exercice 16 (Langages continuables et mots primitifs) : On fixe un alphabet fini Σ de taille au moins 2. Dans ce 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 ?

   Solution :
   Le premier est non primitif car égal à $(abaa)^2$. Le deuxième est primitif car sa taille est un nombre premier.

2. Proposer un algorithme naïf qui, étant donné un mot, détermine s'il est primitif. Préciser la complexité en temps et en espace.

   Solution :
   Pour u de taille n, il s'agit de vérifier, pour chaque préfixe v de taille $\leq \frac{n}{2}$, 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. Donner un exemple de langage rationnel infini qui ne contient aucun mot primitif.

   Solution :
   Le langage aa^+ convient.

4. Donner un exemple de langage rationnel infini qui ne contient que des mots primitifs.

   Solution :
   Le langage a^*b convient.

5. Donner un exemple de langage rationnel infini non continuable. Existe-t-il des langages rationnels continuables dont le complémentaire est infini ?

   Solution :
   Le langage a^* convient : le mot b est un contre-exemple. Pour la deuxième partie de la question, $\Sigma^* a$ convient (car l'alphabet est de taille au moins 2).

6. Étant donné un automate $\mathcal{A}$, proposer un algorithme qui détermine si $\mathcal{L}(\mathcal{A})$ est continuable. Justifier sa correction et préciser la complexité en temps et en espace.

   Solution :
   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 $\mathcal{L}(\mathcal{A})$ est non co-accessible. La réciproque se fait de la même manière.

7. Montrer qu'un langage rationnel continuable contient une infinité de mots primitifs.

   Solution :
   Soit $\mathcal{A}$ un automate fini à n états dont $\mathcal{L}(\mathcal{A})$ est continuable. On peut supposer que tous les états de $\mathcal{A}$ sont accessibles et co-accessibles par la question précédente. Dès lors, $\forall u \in \Sigma^*, \exists v \in \Sigma^*$ tel que

$|v| < n$ et $uv \in \mathcal{L}(\mathcal{A})$. On pose alors, pour $i \in \mathbb{N}$, $i \geq n-1$, $u_i = ab^i$. On pose de plus v_i le plus petit mot tel que $w_i = u_i v_i \in \mathcal{L}(\mathcal{A})$. On a $|v_i| < n$, donc w_i est un mot primitif. En effet, si $w_i = x^p$ avec $p \geq 2$, alors x (la première occurrence) commence par un a (car u_i commence par un a), mais comme $|u_i| \geq n > |v_i|$, 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. Étant donné un langage rationnel continuable L reconnu par un automate $\mathcal{A}$, donner une borne supérieure de la taille du plus petit mot primitif de L.

Solution :
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 à la question 7 est-elle vraie ?

Solution :
La réciproque est fausse, par exemple avec a^*b (tous les mots sont primitifs, mais il n'est pas continuable).

Exercice 17 (Clôture par sur-mots et sous-mots) :

Définition

Soit Σ un alphabet et $u, v \in \Sigma^*$, $u = a_0 \dots a_{m-1}$ et $v = b_0 \dots b_{n-1}$. On dit que u est un **sous-mot** de v, noté $u \preccurlyeq v$, s'il existe une fonction strictement croissante $\varphi : [\![0, m-1]\!] \to [\![0, n-1]\!]$ telle que $\forall i \in [\![0, m-1]\!]$, $a_i = b_{\varphi(i)}$. On dit alors que v est un **sur-mot** de u.

Étant donné un langage L, on note $\widehat{L}$ le langage des sur-mots de mots de L, c'est-à-dire :

$$\widehat{L} = \{v \in \Sigma^* | \exists u \in L, u \preccurlyeq v\}$$

1. Soit $L_0 = ab^*a$ et $L_1 = (ab)^*$. Déterminer des expressions régulières pour $\widehat{L_0}$ et $\widehat{L_1}$.

Solution :
Pour L_0, la seule contrainte pour être un sur-mot est que le mot contienne 2 a, soit $\widehat{L_0} = \Sigma^* a \Sigma^* a \Sigma^*$.
Pour L_1, tout mot est un sur-mot du mot vide qui est dans L_1, donc $\widehat{L_1} = \Sigma^*$.

2. Montrer que pour tout langage L, on a $\widehat{\widehat{L}} = \widehat{L}$.

Solution :
L'inclusion $\supset$ est directe, car tout mot est un sur-mot de lui-même. L'inclusion $\subset$ découle de la transitivité de $\preccurlyeq$: si $u \preccurlyeq v \preccurlyeq w$, alors $u \preccurlyeq w$, donc $w \in \widehat{\widehat{L}} \Rightarrow w \in \widehat{L}$.

3. Existe-t-il des langage L' pour lesquels il n'existe aucun langage L tel que $\widehat{L} = L'$?

Solution :
Le langage L_0 convient comme contre-exemple, car si $L_0 = \widehat{L}$, alors $\widehat{L_0} = \widehat{L} = L_0$, ce qui est faux.

4. Montrer que si L est un langage rationnel, alors $\widehat{L}$ est également rationnel.

Solution :
On montre cette propriété par induction :
- $\widehat{\emptyset} = \emptyset$;
- si $a \in \Sigma$, $\widehat{a} = \Sigma^* a \Sigma^*$;
- si L_1, L_2 sont des langages rationnels, alors :
  - $\widehat{L_1 \cup L_2} = \widehat{L_1} \cup \widehat{L_2}$ (double inclusion assez simple) ;
  - $\widehat{L_1 L_2} = \widehat{L_1}\widehat{L_2}$ (on découpe le mot en deux) ;
  - $\widehat{L_1^*} = \Sigma^*$ (le mot vide est dans le langage).

On admet le théorème suivant :

Théorème

Pour toute suite $(u_n) \in (\Sigma^*)^{\mathbb{N}}$, il existe $i < j$ tels que $u_i \preccurlyeq u_j$.

5. Montrer que pour tout langage L, il existe un langage fini $F \subset L$ tel que $\widehat{F} = \widehat{L}$.

Solution :
Si L est fini, alors $F = L$ convient. Sinon, sachant que Σ^* est dénombrable, alors L l'est également. Soit donc $(u_n)_{n\in\mathbb{N}}$ une énumération des mots de L. Posons alors $F = \{u_n \in L \mid \forall m < n, u_m \not\preccurlyeq u_n\}$. F est bien fini, car dans le cas contraire, il existerait $i < j$, tels que $u_i, u_j \in F$ et $u_i \preccurlyeq u_j$ (par le théorème), ce qui contredirait la construction de F. Par construction également, il semble clair que $F \subset L$.
Montrons alors que $\widehat{L} = \widehat{F}$. L'inclusion $\supset$ est évidente. Réciproquement, soit $v \in \widehat{L}$ tel que $u_n \preccurlyeq v$. Deux cas se présentent : si $u_n \in F$, alors $v \in \widehat{F}$, sinon, $u_n \notin F$, donc il existe $m < n$ tel que $u_m \preccurlyeq u_n$.

Quitte à réitérer le processus, on peut supposer $u_m \in F$, et donc par transitivité de $\preccurlyeq$, on a bien $v \in \widehat{F}$.

Définition

Un langage L est dit **clos par sur-mots** si, pour $u \in L$ et $v \in \Sigma^*$ tels que $u \preccurlyeq v$, on a $v \in L$.

On définit de même un langage **clos par sous-mots**.

6. Montrer que tout langage clos par sur-mots est rationnel.

Solution :

En utilisant les deux questions précédentes, sachant qu'un langage fini est toujours rationnel, on en déduit que $\widehat{L}$ est toujours rationnel (que L soit rationnel ou non). On remarque enfin que si un langage L est clos par sur-mots, alors $\widehat{L} = L$ pour conclure.

7. Montrer que tout langage clos par sous-mots est rationnel.

Solution :

Soit L un langage clos par sous-mots et $\overline{L}$ son complémentaire. Montrons que $\overline{L}$ est clos par sur-mot. En effet, soit $u \in \overline{L}$ et $v \in \Sigma^*$ tels que $u \preccurlyeq v$. Alors si $v \notin \overline{L}$, on a $v \in L$, et donc u est un sous-mot d'un mot de L, donc est censé être dans L (car L est stable par sous-mots). C'est absurde, donc $v \in \overline{L}$, donc $\overline{L}$ est clos par sur-mots, donc rationnel, donc son complémentaire l'est également.

8. Montrer le théorème précédemment admis.

Solution :

Supposons qu'il existe une suite $(u_n)_{n\in\mathbb{N}}$ qui ne vérifie pas la condition (qu'on appellera une **mauvaise suite** pour le reste de la question). Soit alors v_0 de taille minimale telle qu'il existe une mauvaise suite commençant par v_0. Soit v_1 de taille minimale telle qu'il existe une mauvaise suite commençant par v_0, v_1. On construit de même tous les termes de la (mauvaise) suite $(v_n)_{n\in\mathbb{N}}$. Par construction, pour toute mauvaise suite $(w_n)_{n\in\mathbb{N}}$, si i est le plus petit indice pour lequel $v_i \neq w_i$, alors $|v_i| \leq |w_i|$.

L'alphabet Σ étant fini, soit $a \in \Sigma$ tel qu'il existe une infinité de mots de (v_n) commençant par a et soit $p \in \mathbb{N}$ le plus petit entier tel que v_p commence par a. On pose alors $(w_n)_{n\in\mathbb{N}}$ comme la suite formée de $v_0, \dots v_{p-1}$ et de la suite extraite de (v_n) des mots commençant par a auxquels on a retiré la première lettre.

Montrons que la suite (w_n) ainsi formée est une mauvaise suite. En effet, soient $i < j$. Si $j < p$, alors on a bien $w_i = v_i \not\preccurlyeq v_j = w_j$. Si $p \leq i$, alors $aw_i = v_i \not\preccurlyeq v_j = aw_j$, donc $w_i \not\preccurlyeq w_j$. Si $i < p \leq j$, alors $w_i = v_i \not\preccurlyeq v_j = aw_j$, donc $w_i \not\preccurlyeq w_j$, car w_i ne commence pas par a.

De plus, cette suite contredit la minimalité de (v_n), car $|v_p| > |w_p|$ et p est bien le premier indice où les deux suites diffèrent. On conclut par l'absurde.

Exercice 18 (Automates et palindromes) : Dans ce sujet, on fixe Σ un alphabet finif de taille au moins 2.

Définition

Pour $u = a_0 \dots a_{n-1} \in \Sigma^*$, on appelle **miroir de** u le mot $\overline{u} = a_{n-1} \dots a_0$. Un mot u est un **palindrome** si $u = \overline{u}$. On note $\Pi \subset \Sigma^*$ le langage des palindromes et $\Pi_n = \Pi \cap \Sigma^n$.

1. Soit A un automate fini déterministe complet à k états.
Montrer que pour $n \in \mathbb{N}$, $L(A) \cap \Sigma^{2n} = \Pi_{2n} \Rightarrow k \geq |\Sigma|^n$.

Solution :
Supposons $L(A) \cap \Sigma^{2n} = \Pi_{2n}$. Supposons qu'il existe $u \neq v$ deux mots de Σ^n tels que $\delta^*(q_0, u) = \delta^*(q_0, v)$. Sachant que $L(A) \cap \Sigma^{2n} = \Pi_{2n}$, on en déduit que $u\overline{u} \in L(A)$ et que $v\overline{v} \in L(A)$. Cela implique que $\delta^*(\delta^*(q_0, u), \overline{u}) \in F$, soit $\delta^*(\delta^*(q_0, v), \overline{u}) \in F$, soit $v\overline{u} \in L(A)$. C'est absurde, car $v\overline{u}$ n'est pas un palindrome, mais il est de taille $2n$ et dans $L(A)$. On en déduit que $\forall u, v \in \Sigma^n, u \neq v \Rightarrow \delta^*(q_0, u) \neq \delta^*(q_0, v)$. En particulier, cela implique qu'il y a au moins $|\Sigma^n| = |\Sigma|^n$ états.

2. En déduire que Π n'est pas rationnel.

Solution :
Si Π est rationnel et A un automate fini qui le reconnait, alors la propriété précédente s'applique pour toute valeur de $n \in \mathbb{N}$. Cela implique qu'il y a un nombre infini d'états, ce qui est absurde.

3. Soit A un automate fini. $L(A) \cap \Pi$ est-il rationnel ?

Solution :
Cela est impossible, par exemple si $L(A) = \Sigma^*$ (qui est bien reconnaissable).

4. Soit A un automate fini. $\{u \in \Sigma^* | u\overline{u} \in L(A)\}$ est-il rationnel ?

Solution :
Soit $A = (Q, \Sigma, \delta, I, F)$ un automate fini pas nécessairement déterministe. Pour $q \in Q$, posons $A_q = (Q, \Sigma, \delta, I, \{q\})$, qui reconnait $\{u \in \Sigma^*, q \in \delta^*(I, u)\}$. Posons également $\overline{A}_q = (Q, \Sigma, \overline{\delta}, F, \{q\})$, où $\overline{\delta}$ est définie de la façon suivante :

$$\forall p, p' \in Q, \forall a \in \Sigma, p' \in \delta(p, a) \Leftrightarrow p \in \overline{\delta}(p', a)$$

c'est-à-dire où l'on retourne les transitions de A. Une preuve simple permet de voir que $u \in L(\overline{A}_q) \Leftrightarrow \delta^*(q, \overline{u}) \cap F \neq \emptyset$.
On répond à la question par l'affirmative en remarquant que $\{u \in \Sigma^* | u\overline{u} \in L(A)\} = \bigcup_{q \in Q} L(A_q) \cap L(\overline{A}_q)$.
Une autre construction qui pourra servir pour la suite est de considérer $\overline{A} = (Q, \Sigma, \overline{\delta}, F, I)$, puis l'automate produit $A \times \overline{A} = (Q^2, \Sigma, \delta_\times, I \times F, F \times I)$ et enfin sa restriction en limitant les états finaux à $\{(q, q) | q \in Q\}$.

Définition

On note Π_{pair} le langage des palindromes de longueur paire : $\Pi_{\mathsf{pair}} = \bigcup_{n \in \mathbb{N}} \Pi_{2n}$.

5. Proposer un algorithme qui, étant donné un automate fini A, détermine si $L(A) \cap \Pi_{\mathsf{pair}}$ est vide, fini ou infini. Préciser la complexité en temps et en espace.

Solution :
On remarque que $|L(A) \cap \Pi_{\mathsf{pair}}| = |\{u \in \Sigma^* \mid u\overline{u} \in L(A)\}|$, car $f : \Sigma^* \to \Pi_{\mathsf{pair}}$, définie par $u \mapsto u\overline{u}$ définie une bijection de $\{u \in \Sigma^* | u\overline{u} \in L(A)\}$ vers $L(A) \cap \Pi_{\mathsf{pair}}$. Le problème revient donc à déterminer si ce langage est vide, fini ou infini. La construction de l'automate produit décrit précédemment peut

se faire en temps et espace quadratique. On peut alors l'émonder, tester s'il existe un état final accessible et tester l'existence d'un cycle, le tout en temps linéaire par plusieurs parcours de graphes.

6. Modifier l'algorithme précédent pour renvoyer le cardinal de $L(A) \cap \Pi_{\mathsf{pair}}$ lorsqu'il est fini, en supposant que l'automate donné en entrée est déterministe. La complexité est-elle modifiée ?

Solution :

Si l'automate est déterministe et que le langage cherché est fini, on peut procéder par programmation dynamique : le nombre de mots reconnus depuis un état $q \in Q$ est 1 ou 0 selon que l'état soit final ou non, plus le nombre de mots reconnus depuis chaque état vers lesquels q a une transition sortante (quitte à compter plusieurs fois si les transitions sont étiquetées par plusieurs lettres). Pour l'ordre des calculs, il suffit d'utiliser un ordre donné par un tri topologique (car l'automate ne contient pas de cycle), en commençant par la fin. Le calcul aurait alors une complexité linéaire en temps et en espace, en la taille de l'automate produit (c'est-à-dire quadratique en la taille de l'automate initial), ce qui n'augmente pas la complexité totale.

Notons que si l'automate n'est pas déterministe il faudrait le déterminiser, ce qui mènerait à une complexité exponentielle. Compter le nombre de mots reconnus par un automate quelconque est P-difficile.

7. Modifier les algorithmes précédents pour $L(A) \cap \Pi$.

Solution :

Pour traiter les palindromes de longueur impaire, on construit, pour chaque $a \in \Sigma$, un automate produit comme en 4, en ne considérant comme états finaux que les (q, q') où $q' \in \delta(q, a)$. Un tel automate permettra de reconnaitre les mots u tels que $ua\overline{u} \in L(A)$. On somme alors tous les cardinaux pour obtenir la valeur cherchée. La complexité est la même, en rajoutant un facteur $|\Sigma|$.

Exercice 19 (Shuffle d'un langage) :

Définition

Soit Σ un alphabet. Pour $u \in \Sigma^*$ et $A \subset \Sigma$, on note $\mathsf{proj}(u, A)$ la **projection** de u sur A, obtenue en supprimant de u toutes les lettres qui ne sont pas dans A. Par exemple, $\mathsf{proj}(abcabc, \{a, b\}) = abab$.

Pour L un langage, on note $\mathsf{proj}(L, A) = \{\mathsf{proj}(u, A) \mid u \in L\}$.

1. Montrer que si L est rationnel, alors la projection de L sur un alphabet est rationnelle.

Solution :

On montre ce résultat par induction :

- Si $L = \emptyset$ ou $L = a \in \Sigma$, alors le résultat est immédiat (la projection est soit $\emptyset$, soit a).
- Supposons que la projection de L_1 et L_2 sur A soit rationnelle. Alors :
  - $\mathsf{proj}(L_1L_2, A) = \mathsf{proj}(L_1, A)\mathsf{proj}(L_2, A) \in Rat_\Sigma$
  - $\mathsf{proj}(L_1 \cup L_2, A) = \mathsf{proj}(L_1, A) \cup \mathsf{proj}(L_2, A) \in Rat_\Sigma$
  - $\mathsf{proj}(L_1^*, A) = \mathsf{proj}(L_1, A)^* \in Rat_\Sigma$

Pour le reste du sujet, on fixe $k \geq 1$ et $\tilde{\Sigma} = (A_1, \dots, A_k)$ un k-uplet d'alphabets finis, non nécessairement disjoints. On note $\Sigma = A_1 \cup \dots \cup A_k$.

Définition

Soit $L \subset \Sigma^*$ un langage. On appelle **shuffle** de L (par rapport à $\tilde{\Sigma}$) le langage :

$$\mathsf{sh}(L) := \{u \in \Sigma^* \mid \forall i \in [\![1, k]\!], \exists u_i \in L, \mathsf{proj}(u, A_i) = \mathsf{proj}(u_i, A_i)\}$$

2. Soit $\tilde{\Sigma} = (\{a, b\}, \{b, c\})$ et $L = \{acb, bac\}$. Calculer $\mathsf{sh}(L)$.

Solution :

$\mathsf{sh}(L) = \{abc, acb, bac, bca, cab, cba\}$.

3. Montrer que le shuffle d'un langage rationnel est rationnel.

Solution :

Posons $L_i = \{u \in \Sigma^* \mid \exists v \in L, \mathsf{proj}(u, A_i) = \mathsf{proj}(v, A_i)\} = \{u \in \Sigma^* | \mathsf{proj}(u, A_i) \in \mathsf{proj}(L, A_i)\}$. Alors $\mathsf{sh}(L) = \bigcap_{i=1}^{k} L_i$.

Remarquons alors que si L est un langage rationnel et A un alphabet, alors $M = \{u \in \Sigma^* | \mathsf{proj}(u, A) \in L\}$ est rationnel.

En effet, soit $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un automate fini déterministe complet reconnaissant L. Posons $\mathcal{R}$ la relation d'équivalence entre états définie par $p\mathcal{R}q \Leftrightarrow \exists b \in \Sigma \setminus A, p = \delta(q, b)$ ou $q = \delta(p, b)$. Soient $Q_0, \dots, Q_m$ les classes d'équivalences ainsi définies. On pose $\mathcal{B} = (Q_B, \Sigma, \delta', I_B, F_B)$ l'automate non déterministe défini par :

- $Q_B = \{Q_0, \dots, Q_m\}$
- $I_B = Q_0$ est la classe d'équivalence contenant q_0.
- F_B est l'ensemble des classes d'équivalence contenant un état final.
- δ' est défini par :
  - Si $a \in A$, alors $\delta'(Q_i, a) = \{Q_j \in Q_B \mid \exists q_i \in Q_i, \delta(q_i, a) \in Q_j\}$.
  - Si $b \notin A$, alors $\delta'(Q_i, b) = \{Q_i\}$.

Alors $\mathcal{B}$ reconnaît M. En effet :

- Soit $u = u_1 \dots u_n \in L(\mathcal{B})$. Alors il existe une suite de transitions $Q_0 \xrightarrow{u_1} P_1 \dots \xrightarrow{u_n} P_n$ où $P_n \in F_B$. Cependant, les lettres de $\Sigma \setminus A$ ne permettent pas de changer d'état. Donc en les supprimant, on obtient un chemin acceptant $Q_0 \xrightarrow{u_{i_1}} P_{i_1} \dots \xrightarrow{u_{i_\ell}} P_{i_\ell} = P_n$. Dès lors, on peut trouver un chemin acceptant dans $\mathcal{A}$: $q_0 \xrightarrow{u_{i_1}} q_{i_1} \dots \xrightarrow{u_{i_\ell}} q_{i_\ell} \in F$. On en déduit que $u \in M$.
- La réciproque est similaire.

Finalement, par la question 1 et par clôture par intersection, on en déduit que le shuffle d'un langage rationnel est rationnel.

Définition

Soit L un langage. On dit que L est un **shuffle rationnel** si $L = \mathsf{sh}(L)$ et L est rationnel. On dit que c'est un **semi-shuffle rationnel** s'il existe $m \geq 1$ et des shuffles rationnels $L_1, \dots, L_m$ tels que $L = L_1 \cup \dots \cup L_m$.

4. Soit $\tilde{\Sigma} = (\{a, b\}, \{b, c\})$ et $L = (a + b + c)^*(ac + ca)(a + b + c)^*$. Montrer que L n'est pas un shuffle rationnel.

Solution :

$\mathsf{sh}(L) = (a + b + c)^* a (a + b + c)^* \cap (a + b + c)^* c (a + b + c)^* \neq L$

5. Montrer que le langage précédent n'est pas un semi-shuffle rationnel.

Solution :

Supposons que L est un semi-shuffle rationnel. Alors il existe L_1, …, L_m des shuffle rationnels tels que $L = \cup_{i=1}^m L_i$. Posons alors, pour $k \in [\![0, m]\!]$: $u_k = b^k a c b^{m-k}$. On remarque aisément que $u_k \in L$. Par le principe des tiroirs, on en déduit qu'il existe $0 \leq i < j \leq m$ et $\ell \in [\![0; m]\!]$ tels que $u_i, u_j \in L_\ell$. Posons alors $u = b^i a b^{j-i} c b^{m-j}$. On remarque que $\mathsf{proj}(u_i, \{a, b\}) = b^i a b^{m-i} = \mathsf{proj}(u, \{a, b\})$ et que $\mathsf{proj}(u_j, \{b, c\}) = b^j c b^{m-j} = \mathsf{proj}(u, \{b, c\})$. L_ℓ étant supposé un shuffle rationnel, cela implique que $u \in L_\ell \subset L$. Par l'absurde, on en déduit le résultat.

6. Soit $\tilde{\Sigma} = (\{a\}, \{c\})$. Montrer que la classe des semi-shuffles rationnels (par rapport à $\tilde{\Sigma}$) est strictement plus grande que la classe des shuffles rationnels.

Solution :

Soit $\tilde{\Sigma} = (\{a\}, \{c\})$. On pose $L_1 = a^*$ et $L_2 = c^*$. Il est clair que L_1 et L_2 sont des shuffles rationnels. De plus, si on pose $L = L_1 + L_2$, alors $\mathsf{sh}(L) = (a + c)^* \neq L$.

7. Montrer que la classe des shuffles rationnels (pour un $\tilde{\Sigma}$ bien choisi) n'est pas stable par complémentaire.

Solution :

On pose $L = \{abc\}$ et $\tilde{\Sigma} = (\{a, b\}, \{b, c\})$. Alors on a $acb \in \overline{L}$ et $bac \in \overline{L}$. Si $\overline{L}$ était un shuffle rationnel, cela impliquerait que $abc \in \overline{L}$, ce qui est absurde.

8. Montrer que la classe des semi-shuffles rationnels est stable par complémentaire.

Solution :

On commence par des résultats faciles à montrer : la rationalité est donnée par la stabilité des langages rationnels par les opérations ensemblistes. De plus, l'union finie de semi-shuffles rationnels est clairement un semi-shuffle rationnel. L'intersection de deux shuffles rationnels est un shuffle rationnel. L'intersection de semi-shuffles rationnels est donc un semi-shuffle rationnel (par les deux propriétés précédentes et la distributivité).

Finalement, montrons que le complémentaire d'un shuffle rationnel est un semi-shuffle rationnel, ce qui terminera la preuve :

Soit L un shuffle rationnel. Alors on a vu que :

$$L = \bigcap_{i=1}^{k} \{u \in \Sigma^* \mid \mathsf{proj}(u, A_i) \in \mathsf{proj}(L, A_i)\}$$

On en déduit :

$$\overline{L} = \bigcup_{i=1}^{k} \{u \in \Sigma^* \mid \mathsf{proj}(u, A_i) \notin \mathsf{proj}(L, A_i)\}$$

Montrons alors que $L_i = \{u \in \Sigma^* \mid \mathsf{proj}(u, A_i) \notin \mathsf{proj}(L, A_i)\}$ est un shuffle rationnel. On a déjà trivialement $L_i \subset \mathsf{sh}(L_i)$.

Réciproquement, soit $u \notin L_i$. Alors $\exists v \in L, \mathsf{proj}(u, A_i) = \mathsf{proj}(v, A_i)$ et de plus, $\forall w \in L_i, \mathsf{proj}(u, A_i) \neq \mathsf{proj}(w, A_i)$. Cela montre que $u \notin \mathsf{sh}(L_i)$.

Exercice 20 (Langages fins) :

Définition

Soit Σ un alphabet et $L \subset \Sigma^*$ un langage. La **densité** de L est la fonction $\rho_L : \mathbb{N} \to \mathbb{N}$ définie par $\rho_L(n) = |L \cap \Sigma^n|$.

1. Majorer et minorer $\rho_L(n)$ en fonction de $|\Sigma|$ et n. Ces bornes sont-elles atteintes ?

Solution :
$0 \leq \rho_L(n) \leq |\Sigma|^n$

2. Donner une fonction de $\mathbb{N}$ dans $\mathbb{N}$ qui n'est la densité d'aucun langage rationnel.

Solution :
Sachant que $\{u \mid |u| \text{ est une puissance de } 2\}$ n'est pas rationnel (lemme de pompage), on en déduit que f définie par $f(n) = |\Sigma|^n$ si n est une puissance de 2 et 0 sinon n'est la densité d'aucun langage rationnel sur Σ.

Définition

Un langage rationnel L est dit fin s'il existe $M \in \mathbb{N}$ qui majore la fonction ρ_L.

On cherche à montrer le théorème suivant :

Théorème

Un langage rationnel L est fin si et seulement s'il existe $k \in \mathbb{N}$ et des mots $u_1, \ldots, u_k, v_1, \ldots, v_k, w_1, \ldots, w_k \in \Sigma^*$ tels que $L = \bigcup_{i=1}^{k} u_i v_i^* w_i$.

3. Montrer le sens retour du théorème.

Solution :
Soit L vérifiant la deuxième condition. Alors $\rho_L(n) \leq \sum_{i=1}^{k} \rho_{u_i v_i^* w_i}(n) \leq \sum_{i=1}^{k} 1 = k$ (il y a au plus un mot de $u_i v_i^* w_i$ qui est de taille n). On en déduit que L est un langage fin.

Définition

Un automate fini déterministe est dit **monoboucle** si tout état q pouvant apparaître plusieurs fois dans un chemin acceptant vérifie les propriétés suivantes :

- il n'existe qu'un seul chemin de q vers q n'ayant pas q comme état intermédiaire ;
- il n'existe qu'un nombre fini de chemins de l'état initial vers q n'ayant pas q comme état intermédiaire ;
- il n'existe qu'un nombre fini de chemins de q vers un état final n'ayant pas q comme état intermédiaire.

4. Soit L un langage fin et $\mathcal{A}$ un automate fini déterministe reconnaissant L. Montrer que $\mathcal{A}$ est monoboucle.

Solution :
Soit L un langage fin et M le majorant associé. Soit $\mathcal{A}$ un automate reconnaissant L et q un état pouvant apparaître plusieurs fois dans un chemin acceptant.

- Supposons qu'il existe deux chemins de q vers q correspondant aux calculs des mots x et y. Soient u et w deux mots tels que $\delta^*(q_0, u) = q$ et $\delta^*(q, w) \in F$. Alors $u(x+y)^*w \in L$. En

particulier, pour $n \geq M$, les mots de la forme $ux^k yx^{n-k}w$ sont reconnu par $\mathcal{A}$. On en déduit que $\rho_L(|u| + |w| + |y| + n|x|) \geq n + 1 > M$. On conclut le premier point par l'absurde.

- Soit v et w des mots tels que $\delta^*(q, v) = q$ et $\delta^*(q, w) \in F$. Soit $n > M|v|$ et $u_1, \ldots, u_n$ des mots distincts tels que $\delta^*(q_0, u_i) = q$. Alors $\forall i \in [\![1, n]\!], u_i v^* w \subset L$.

  De plus, $u_i v^* w$ contient un mot de longueur k si et seulement si $|u_i| = k - |w| \mod |v|$ et $k \geq |u_i| + |w|$. Par le principe des tiroirs, il existe donc $M + 1$ mots $u_{i_0}, \ldots, u_{i_M}$ de taille égale modulo $|v|$. Dès lors, en posant $k = \max(|u_{i_j}|) + |w| + |v|$, il existe un mot de taille k dans chacun des $u_i v^* w$ et donc $\rho_L(k) \geq M + 1 > M$. On conclut par l'absurde sur la finitude de tels u_i.
- Idem que précédemment en échangeant le rôle des préfixes et suffixes.

5. Terminer la preuve du théorème.

Solution :

Soit L un langage fin et $\mathcal{A}$ un automate fini monoboucle le reconnaissant. Alors on peut écrire $L = L' \cup \bigcup_{q \in Q} L_q$ où :

- L' est l'ensemble des mots u dont le chemin acceptant dans $\mathcal{A}$ ne passe pas deux fois par le même état. Il est de cardinal fini.
- L_q est l'ensemble des mots u dont le chemin acceptant passe plusieurs fois par l'état q.

Par les propriétés des automates monoboucles, on peut écrire L_q comme :

$$L_q = \bigcup_{i=1}^{k} \bigcup_{j=1}^{\ell} u_i v^* w_j$$

On en déduit le résultat.

6. Donner un algorithme qui reçoit en entrée un automate fini déterministe $\mathcal{A}$ et qui renvoie `vrai` si le langage reconnu par $\mathcal{A}$ est fin et `faux` sinon.

Solution :

Il suffit de vérifier que l'automate est monoboucle :

(a) Émonder l'automate.

(b) Supprimer tous les états sans transition entrante.

(c) Supprimer tous les états sans transition sortante.

(d) S'il existe un état avec plus de deux transitions sortante, renvoyer vrai. Sinon, renvoyer faux. En effet, il suffit de vérifier qu'il ne reste qu'un ensemble de cycles disjoints.

Exercice 21 (Algorithmes randomisés) :

1. On dispose d'une fonction `foo()` qui renvoie aléatoirement et uniformément un entier de $[\![1, 5]\!]$. Écrire un algorithme qui renvoie aléatoirement et uniformément un entier de $[\![1, 7]\!]$, en se limitant à des opérations sur les entiers.

Solution :

Algorithme : foo2

Données :
Résultat : un entier tiré uniformément dans $[\![1, 7]\!]$
$a, b \leftarrow$ `foo()`, `foo()` ;
$c \leftarrow 5(a-1) + b - 1$;
si $c < 21$ **alors**
  retourner $c//3 + 1$;
sinon
  retourner *foo2()* ;

Il est clair que chaque valeur de $[\![1, 7]\!]$ peut être renvoyée avec la même probabilité, car c prend une valeur entre 0 et 24 uniformément. La fonction a $\frac{4}{25} = 16\%$ de chance de relancer un appel.

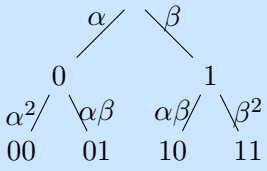
2. On dispose d'une fonction `bar()` qui renvoie 0 avec probabilité 0.42 et 1 avec probabilité 0.58. Écrire un algorithme qui renvoie 0 ou 1 aléatoirement, chacun avec probabilité $0,5$.

Solution :

Algorithme : bar2

Données :
Résultat : un entier tiré uniformément dans $[\![0, 1]\!]$
$a, b \leftarrow$ `bar()`, `bar()` ;
si $a \neq b$ **alors**
  retourner a ;
sinon
  retourner *bar2()* ;

Si 0 est tiré avec une probabilité α et 1 avec $1 - \alpha = \beta$, on obtient l'arbre de probabilité suivant :



On en déduit que la probabilité de tirer 01 est la même que celle de tirer 10, et ce quelle que soit la probabilité pour `bar` de tirer 0. Si $\alpha \in]0, 1[$, l'algorithme `bar2()` sera correct.

3. On dispose d'une fonction `baz(i)` qui renvoie aléatoirement et uniformément un entier de $[\![1, i]\!]$. Écrire un algorithme qui mélange uniformément un tableau de taille n en complexité temporelle $\mathcal{O}(n)$. L'algorithme devra travailler en place, c'est-à-dire ne pas effectuer de copie des données, mais faire des permutations entre éléments du tableau.

Solution :
On propose le mélange de Knuth vu en TP (les indices du tableau commencent à 1) :

Algorithme : Mélange de Knuth

Données : un tableau t de taille n
Résultat : un mélange uniforme de t par effets de bords
pour $i = 2$ *à* n **faire**
  $j \leftarrow$ `baz`(i) ;
  $t[i] \leftrightarrow t[j]$;

4. Un flux de valeurs de taille n défile successivement avec les défauts suivants :
   - n est trop grand pour qu'on puisse garder en mémoire toutes les valeurs déjà vues ;
   - n n'est pas connu à l'avance.

   Écrire un algorithme qui choisit k éléments parmi les n, de manière équiprobable. Lors de l'accès à la i-ème valeur du flux, on se limitera à un appel à `baz(i)` (`baz` étant la fonction définie à la question précédente).

Solution :

On propose l'algorithme suivant :

Algorithme : tirage dans un flux

Données : un entier k, un flux de valeurs F
Résultat : k éléments de F tirés de manière équiprobable
$t \leftarrow [-1, \dots, -1]$ (de taille k) ;
$i \leftarrow 1$;
tant que F *n'est pas terminé* **faire**
  $x \leftarrow$ valeur suivante de F ;
  $j \leftarrow$ `baz`(i) ;
  si $i \leq k$ **alors**
    $t[i] \leftarrow x$;
  sinon si $j \leq k$ **alors**
    $t[j] \leftarrow x$;
  $i \leftarrow i + 1$;
retourner t ;

Il s'agit maintenant de montrer que cet algorithme renvoie bien une sélection uniforme de k valeurs parmi les n du flux F :

(a) Commençons par les éléments après les k premiers :
   - Pour $i > k$, la probabilité que le i-ème élément du flux soit ajouté au tableau (disons à un indice j) est $\frac{k}{i}$.
   - Pour que cet élément ne soit jamais remplacé, il faut que l'indice j ne soit jamais tiré lors des appels à `baz` qui vont suivre. Cette probabilité vaudra successivement $1 - \frac{1}{i+1}$, $1 - \frac{1}{i+2}$, $\dots$, $1 - \frac{1}{n}$. Ainsi la probabilité que l'indice j ne soit jamais tiré est le produit de ces probabilités, soit $\prod_{m=i+1}^{n} \left(1 - \frac{1}{m}\right) = \prod_{m=i+1}^{n} \frac{m-1}{m} = \frac{i}{n}$.
   - Finalement, la probabilité que le i-ème élément du flux soit dans le tableau à la fin de l'algorithme est le produit des deux probabilités précédentes, soit $\frac{k}{i} \times \frac{i}{n} = \frac{k}{n}$.

(b) Pour les k premiers éléments du flux, on se contente de calculer la probabilité qu'ils ne soient jamais remplacés, soit $\prod_{m=k+1}^{n} \left(1 - \frac{1}{m}\right) = \frac{k}{n}$.

Ainsi, tous les éléments du flux ont bien la même probabilité de faire partie des k restants.

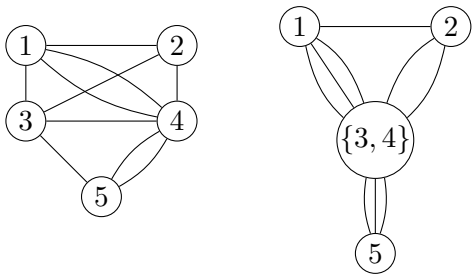
Définition

Un **multigraphe** est un couple $G = (V, E)$ tel que E est un multi-ensemble d'éléments de $\mathcal{P}_2(V)$, c'est-à-dire un graphe où il peut y avoir plusieurs arêtes entre les mêmes sommets. Une **coupe** d'un multigraphe $G = (V, E)$ est une partition de V en V_1, V_2. Une coupe (V_1, V_2) est dite **minimale** si elle minimise le nombre d'arêtes entre V_1 et V_2.

Soit $G = (V, E)$ un multigraphe et $u \neq v$ deux sommets de V. La **contraction** de $\{u, v\}$ dans G, notée $G/\{u, v\}$, est le multigraphe formé à partir de G où u et v ont été fusionnés, où les arêtes entre u et v ont disparu, et toute arête entre u et un sommet w devient une arête entre $\{u, v\}$ et w (et de même pour une arête entre v et w).

Exemple

Un multigraphe et sa contraction de $\{3,4\}$:



Algorithme : coupe
Données : Un graphe $G = (S, A)$
Résultat : Un contraction de G
tant que $|S| > 2$ **faire**
  Choisir $\{u, v\}$ dans A uniformément ;
  $G \leftarrow G/\{u, v\}$;
retourner S ;

On cherche, étant donné un multigraphe connexe, à trouver une coupe minimale par un algorithme randomisé.

5. On tire aléatoirement et uniformément une coupe. Déterminer la probabilité qu'elle soit minimale.

Solution :
Il s'agit ici de compter le nombre de coupes possibles, c'est-à-dire le nombre de partitions de taille 2 dans un ensemble à n éléments. Pour créer une coupe, on place le premier sommet dans V_1, et chaque autre sommet a 2 choix possibles (chacun des deux ensembles). Cela forme 2^{n-1} choix possibles, auquel il faut enlever le choix où tous les sommets se retrouvent dans V_1 (car ce n'est pas une coupe). Ainsi, la probabilité de choisir une coupe minimale si elle est unique est $\frac{1}{2^{n-1}-1}$. On peut montrer (voir plus loin) que le nombre maximal de coupes minimales (par exemple dans un cycle de taille n) est au plus $\binom{n}{2}$, donc cette probabilité est au plus $\frac{n(n-1)}{2^n-2}$.

On propose l'algorithme `coupe` ci-dessus :

6. Déterminer la complexité temporelle de l'opération de contraction.

Solution :
En utilisant une représentation par matrice d'adjacence, une contraction peut être effectuée par un simple parcours de toute la matrice, soit $\mathcal{O}(n^2)$.

7. Montrer que l'algorithme renvoie bien une coupe.

Solution :
À tout moment, les sommets de G forment une partition de l'ensemble initial V. C'est effectivement un invariant de boucle :

- Initialement, chaque sommet est un singleton et tous les sommets sont présents, donc c'est effectivement une partition ;
- à chaque contraction, deux ensembles sont fusionnés, aucun sommet ne disparait lors de la fusion, et les autres ensembles ne sont pas modifiés. Cela reste une partition.

À la fin de la boucle, il ne reste que deux sommets, donc une partition de taille 2. Il s'agit bien d'une coupe.

8. Déterminer la probabilité que l'algorithme `coupe` renvoie une coupe minimale.
On pourra se contenter d'une borne inférieure.
Comment utiliser `coupe` pour augmenter cette probabilité ?

Solution :
Supposons que la coupe minimale est unique et soit C l'ensemble des arêtes entre les deux ensembles de la coupe et $k = |C|$. Il faut calculer la probabilité qu'aucune arête de C ne soit contracté au cours de l'exécution de l'algorithme.
Dans un premier temps, il est clair que le degré minimal d'un sommet de G est au moins k, sinon isoler ce sommet formerait une coupe avec moins d'arêtes traversantes que C. On en déduit que

$|E| \geq \frac{nk}{2}$. Par ailleurs, la probabilité de tirer une arête de C lors de la première contraction est $\frac{k}{|E|} \leq \frac{2}{n}$. On en déduit que la probabilité d'éviter une arête de C est $\geq 1 - \frac{2}{n}$. Après contraction, on obtient un graphe d'ordre $n-1$. Ainsi, la probabilité p_n d'obtenir une coupe minimale dans un graphe d'ordre n vérifie $p_n \geq \left(1 - \frac{2}{n}\right) p_{n-1}$, soit $p_n \geq \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} = \frac{2}{n(n-1)}$ (ce qui est bien mieux que le tirage au hasard d'une coupe).

Pour améliorer la probabilité, on peut répéter l'algorithme plusieurs fois et garder la coupe minimale parmi les résultats obtenus. Par exemple, pour obtenir une probabilité au moins $\frac{n-1}{n}$, on peut répéter l'opération $\binom{n}{2} \ln n$ fois. En effet, la probabilité de ne jamais obtenir de coupe minimale est au plus :

$$\left(1 - \frac{2}{n(n-1)}\right)^{\frac{n(n-1)}{2} \ln n} \leq \left(\frac{1}{e}\right)^{\ln n} = \frac{1}{n}$$

Pour montrer la borne sur le nombre de coupes minimales, comme l'algorithme de Karger a une probabilité au moins $\frac{2}{n(n-1)}$ de renvoyer une coupe minimale en particulier, il ne peut pas y avoir plus de $\frac{n(n-1)}{2}$ coupes minimales. De plus, cette borne est atteinte pour les cycles de taille n, donc la borne est optimale.

Exercice 22 (Réseaux de tri) :

Définition

Soit T un tableau de taille n. On appelle **module de comparaison et d'échange** (ou simplement module) un couple (i, j) où $i, j \in [\![1, n]\!]$, $i < j$. On appelle **réseau de tri** une suite finie de modules.

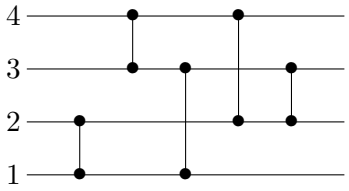
On définit l'**action** d'un module (i, j) sur T par l'échange des cellules $T[i]$ et $T[j]$ si et seulement si $T[j] < T[i]$ (et aucune modification sinon). On définit l'**action** d'un réseau de tri sur T comme l'action de chacun de ses modules dans l'ordre où ils apparaissent.

On représente graphiquement un réseau de tri pour un tableau T de la manière suivante :

- chaque indice de T est représenté par une ligne horizontale ;
- les modules sont représentés de gauche à droite, chacun d'entre eux étant représenté par une ligne verticale reliant les lignes des deux indices correspondant.

Exemple

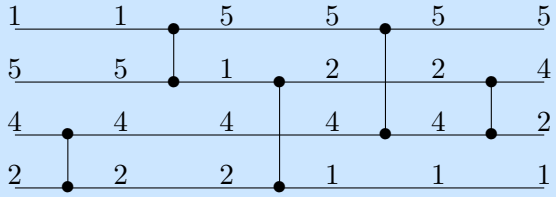
Le réseau de tri $(1, 2), (3, 4), (1, 3), (2, 4), (2, 3)$ est représenté par :



1. Quelle est l'action du réseau de tri donné en exemple sur le tableau $[2, 4, 5, 1]$? Ce réseau permet-il de trier tout tableau de taille 4 ?

   Solution :

   On obtient le déroulement suivant :

   Après les actions des deux premiers modules, les indices 1 et 3 contiennent les minimums de $\{T[1], T[2]\}$ et $\{T[3], T[4]\}$ respectivement (et de même pour les indices 2 et 4 qui contiennent les maximums). Après l'action du module $(1, 3)$, l'indice 1 contient le minimum du tableau. Après l'action de $(2, 4)$, l'indice 4 contient le maximum du tableau. Enfin, après l'action du module $(2, 4)$, les deux derniers éléments sont placés correctement. Ce réseau de tri trie effectivement tout tableau de taille 4.

2. Décrire comment construire un réseau de tri naïf qui trie tout tableau de taille n. Combien de modules contient-il asymptotiquement ?

   Solution :

   On peut implémenter un tri par insertion par exemple : en notant R_k le réseau $(k-1, k), (k-2, k-1), \ldots, (1, 2)$, il suffit de concaténer R_2, R_3, …, R_n. Il est clair que R_k contient $k-1$ modules, donc ce réseau contient $\frac{n(n-1)}{2}$ modules.

3. Montrer qu'un réseau de tri permet de trier tout tableau si et seulement si il permet de trier tout tableau de booléens.

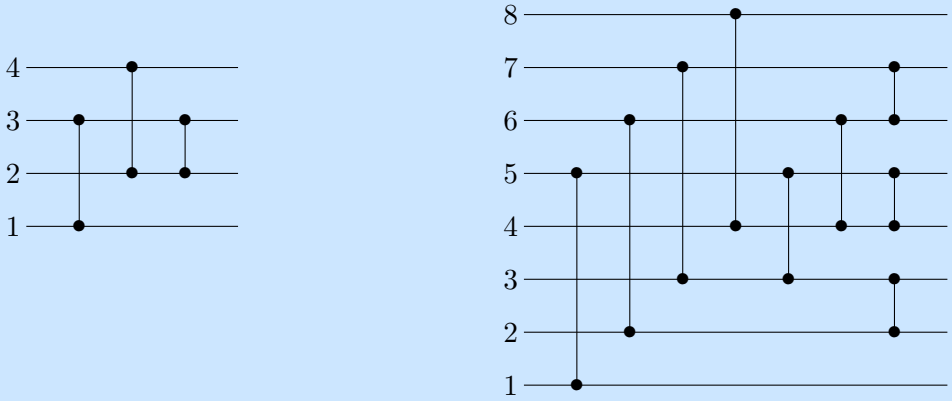
   Solution :

   L'un des deux sens est évident. Pour l'autre sens, on montre qu'une fonction croissante commute toujours avec un module, donc avec un réseau de tri (par récurrence). Dès lors, supposons R un réseau de tri qui trie correctement tout tableau de booléens. Soit T un tableau quelconque et i, j deux indices tels que $T[i] < T[j]$. En notant i' et j' les indices où se retrouvent ces éléments dans $R(T)$

et montrons que $i' < j'$. En effet, en posant $f(x)$ l'indicatrice de $x > T[i]$, le réseau trie correctement $f(T)$. Par la propriété précédente, on a bien $i' < j'$.

4. Décrire un réseau qui fusionne deux tableaux triés de taille 2. Faire de même pour un réseau qui fusionne deux tableaux triés de taille $4 = 2^2$.

Solution :
On s'inspire de la première question :



Le deuxième réseau illustre l'idée des questions suivantes.

5. On considère deux tableaux triés de taille paire $[a_1, \ldots, a_{2n}]$ et $[b_1, \ldots, b_{2n}]$. On note $[c_1, c_2, \ldots, c_{2n}]$ le résultat de la fusion de $[a_1, a_3, \ldots, a_{2n-1}]$ et $[b_1, b_3, \ldots, b_{2n-1}]$; et on note $[d_1, d_2, \ldots, d_{2n}]$ le résultat de la fusion de $[a_2, a_4, \ldots, a_{2n}]$ et $[b_2, b_4, \ldots, b_{2n}]$. Montrer que le tableau suivant est trié :

$$[c_1, \min(c_2, d_1), \max(c_2, d_1), \min(c_3, d_2), \max(c_3, d_2), \ldots, \min(c_{2n}, d_{2n-1}), \max(c_{2n}, d_{2n-1}), d_{2n}]$$

Solution :
Il est clair que c_1 est le plus petit élément du tableau, car $c_1 = \min(a_1, b_1)$. De même, d_{2n} est clairement la plus grande valeur du tableau.

Supposons que c_p soit la k-ème valeur dans le tableau $[a_1, a_3, \ldots, a_{2n-1}]$. On a donc $c_p = a_{2k-1}$. Il y a donc $2k-2$ éléments de a plus petits que c_p. Par ailleurs, comme c_p est en p-ème position dans le tableau d, il y a $p-1-(k-1) = p-k$ valeurs b_{2j-1} plus petites que c_p, soit $2(p-k)$ ou $2(p-k)-1$ valeurs plus petites que c_p dans b. Par conséquent, il a $2p-2$ ou $2p-3$ valeurs plus petites que c_p dans le tableau constitué de a et b trié. c_p est donc à la $2p-1$ ou $2p-2$-ème position dans le tableau.

Un raisonnement similaire permet de trouver que d_p se trouve à la $2p$ ou $2p+1$-ème position, ce qui permet de conclure.

6. En déduire un réseau qui fusionne deux tableaux triés de même taille $n = 2^k$. Combien de modules ce réseau comprend-il ?

Solution :
Pour fusionner deux tableaux de taille 2^k, on fusionne les sous-tableaux d'indices pairs et les sous-tableaux d'indices impairs, puis on utilise $2^k - 1$ modules pour les dernières comparaisons. En notant $M(k)$ le nombre de modules pour fusionner deux tableaux de taille 2^k, on a :

- $M(0) = 1$;
- pour $k > 0$, $M(k) = 2M(k-1) + 2^k - 1$.

On en déduit que $M(k) = k2^k + 1$ (par une récurrence rapide), soit $n \log_2 n + 1$ modules nécessaires pour fusionner deux tableaux de taille n qui est une puissance de 2.

7. En déduire un réseau de tri efficace qui trie un tableau quelconque de taille $n = 2^k$. Combien de modules ce réseau comprend-il ?

Solution :
C'est le principe du tri fusion :

- on trie un tableau de taille 1 sans module ;
- on trie un tableau de taille 2^{k+1} en triant chaque moitié de taille 2^k, puis en fusionnant ces deux moitiés.

En notant $M(k)$ le nombre de modules utilisés pour trier un tableau de taille 2^k, on a :

- $M(0) = 0$;
- pour $k > 0$, $M(k) = 2M(k-1) + k2^k + 1$.

À nouveau, on montre par récurrence que $M(k) = k(k+1)2^{k-1} + 2^k - 1$, soit $\mathcal{O}(n(\log n)^2)$ pour trier un tableau de taille n.

Définition

On dit que des modules consécutifs dans R peuvent **agir pendant la même unité de temps** s'ils n'ont aucun indice en commun. On définit le **temps** de R comme le nombre minimal d'unités de temps nécessaires pour exécuter R.

Exemple

Le réseau de tri donné en exemple précédemment possède un temps égal à 3. On peut faire agir $(0,1),(2,3)$ à la première unité de temps, $(0,2),(1,3)$ à la deuxième et $(1,2)$ à la troisième.

8. Déterminer le temps des réseaux de tri des questions 2 et 7 (en supposant les modules ordonnés de manière optimale).

Solution :
Pour le premier réseau, on remarque que R_k nécessite un temps égal à $k-1$ et que seuls le dernier module de R_k et le premier de R_{k+1} peuvent être exécutés en même temps. Le temps total est donc $\mathcal{O}(n^2)$.
Pour le deuxième réseau, on trouve des formules de récurrence, pour le réseau de fusion et le réseau de tri :

- pour la fusion, on remarque que la fusion des indices pairs et impairs peuvent se faire simultanément. De même, les modules supplémentaires peuvent agir sur une seule unité de temps. On obtient la formule de récurrence du temps qui est : $T(k) = T(k-1) + 1$. Sachant que $T(0) = 1$, on obtient $T(k) = k + 1$;
- pour le tri, c'est la même idée : le tri des deux sous-tableaux peuvent s'effectuer en même temps. On obtient $T(k) = T(k-1) + k$. Sachant que $T(0) = 0$, on obtient $T(k) = \frac{k(k+1)}{2}$.

On en déduit que le temps nécessaire pour trier un tableau de taille n est $\mathcal{O}((\log n)^2)$.

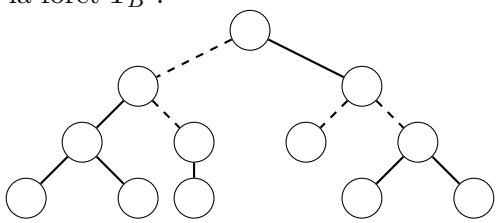
Exercice 23 (Comptage de sous-arbres) :

Définition

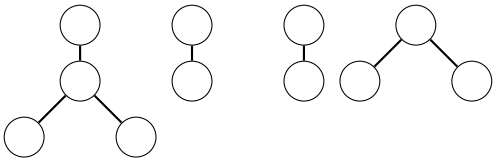
On appelle **forêt** un ensemble d'arbres binaires. Si T est un arbre binaire dont A sont les arêtes, pour $B \subseteq A$, on note T_B la forêt obtenue en conservant les arêtes de B, et en ne gardant que les sous-arbres de taille ≥ 2.

Exemple

La figure suivante représente à gauche un arbre T et un ensemble d'arêtes B en traits pleins et à droite la forêt T_B :



Arbre T



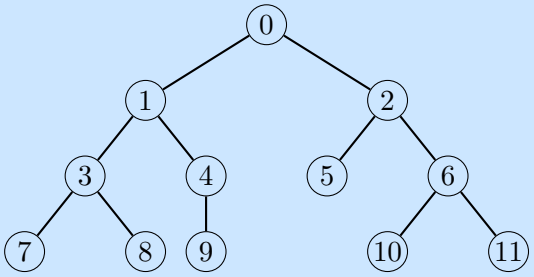
Forêt T_B

On s'intéresse dans ce problème, étant donné un arbre T d'arêtes A et une certaine condition Φ, à compter le nombre de sous-ensembles d'arêtes $B \subseteq A$ tels que la forêt T_B satisfasse la condition Φ.

1. Pour l'arbre T et la forêt T_B donnés en exemple, combien de sous-ensembles d'arêtes B' existe-t-il tels que $T_{B'} = T_B$ (à réétiquetage près) ? Combien de sous-ensembles d'arêtes B' existe-t-il tels que T_B consiste exactement en deux arêtes isolées ?

Solution :

Donnons des noms aux nœuds pour l'explication :



Le premier sous-arbre à obtenir dans T_B apparaît quatre fois dans T : $(1,3,7,8)$, $(0,1,3,4)$, $(0,2,5,6)$ et $(2,6,10,11)$. Distinguons :

- si on garde $(1,3,7,8)$, alors le dernier sous-arbre ne peut être que $(2,5,6)$ ou $(6,10,11)$. Dans le premier cas, on ne peut pas former les deux arbres de taille 2 ; dans le deuxième cas, on obtient la bonne forêt soit en conservant $(0,2)$ et $(4,9)$ (c'est l'ensemble B), soit en conservant $(2,5)$ et $(4,9)$;
- si on garde $(0,1,3,4)$, alors le dernier sous-arbre ne peut être que $(2,5,6)$ ou $(6,10,11)$. Dans les deux cas, on ne peut pas former les deux arbres de taille 2. Ce cas est donc à éliminer ;
- si on garde $(0,2,5,6)$, alors le dernier sous-arbre ne peut être que $(1,3,4)$ ou $(3,7,8)$. Dans les deux cas, on ne peut pas former les deux arbres de taille 2 ;
- si on garde $(2,6,10,11)$, alors le dernier sous-arbre ne peut être que $(1,3,4)$ ou $(3,7,8)$. Dans le premier cas, on ne peut pas former les deux arbres de taille 2. Dans le deuxième cas, on obtient la bonne forêt en conservant $(0,1)$ et $(4,9)$.

Il y a donc trois sous-ensembles B' (dont B) tels que $T_{B'} = T_B$.

Pour la deuxième partie de la question, il s'agit de dénombrer le nombre de façons de choisir deux arêtes qui n'ont pas de sommet en commun. Il y a 11 arêtes au total, et 14 paires d'arêtes adjacentes, soit $\binom{11}{2} - 14 = 41$ possibilités.

2. Proposer un algorithme naïf pour déterminer, étant donné un arbre T et un entier $h \in \mathbb{N}^*$, le nombre de sous-ensemble d'arêtes B tels que T_B contienne un arbre de hauteur au moins h. Donner sa complexité en fonction de T et de h.

Solution :
Vu qu'on parle d'algorithme naïf, on pense ici à parcourir tous les sous-ensembles d'arêtes et à calculer la hauteur maximale d'un sous-arbre pour chacun, et ne garder que ceux pour lesquels la hauteur dépasse h. On obtiendrait une complexité en $\mathcal{O}(n2^n)$, où $n = |T|$.

3. Proposer un algorithme plus efficace en utilisant de la programmation dynamique.
Discuter de la complexité en temps et en espace.

Solution :
Pour chaque nœud x, notons $T(x)$ le sous-arbre de T enraciné en x. Pour chaque entier $k \in [\![0, h-1]\!]$, définissons :

- $\varphi(x)$ comme le nombre de sous-ensembles d'arêtes de $T(x)$ ayant un sous-arbre de hauteur supérieure ou égale à h ;
- $\psi(x,k)$ comme le nombre de sous-ensembles d'arêtes de $T(x)$ ayant un sous-arbre enraciné en x de hauteur exactement k, et aucun sous-arbre de hauteur $\geq h$;
- $\sigma(x,k)$ comme le nombre de sous-ensembles d'arêtes de $T(x)$ ayant un sous-arbre enraciné en x de hauteur inférieure ou égale à k, et aucun sous-arbre de hauteur $\geq h$.

On remarque que $\sigma(x,k) = \sum_{i=0}^{k} \psi(x,k)$, et que $\varphi(x)+\sigma(x,h-1) = 2^{|T(x)|-1}$ (car $T(x)$ possède $T(x)-1$ arêtes).
Remarquons de plus que pour un nœud $x = N(g,d)$, on a :

- $\varphi(x)$ est la somme de :
  - $4(\varphi(g) \times \varphi(d) + \varphi(g) \times \sigma(d,h-1) + \sigma(g,h-1) \times \varphi(d))$ (tous les cas où l'un des deux enfants contient un sous-arbre de hauteur $\geq h$, fois 4, ce qui correspond au nombre de choix possibles parmi les arêtes (x,g) et (x,d)) ;
  - $\psi(g,h-1) \times \sigma(d,h-1) + \sigma(g,h-1) \times \psi(d,h-1)$ (on ne garde qu'une seule arête parmi (x,g) et (x,d), qui forme un arbre enraciné en x de hauteur exactement h) ;
  - $\psi(g,h-1) \times \sigma(d,h-2) + \sigma(g,h-2) \times \psi(d,h-1) + \psi(g,h-1) \times \psi(d,h-1)$ (on garde les deux arêtes (x,g) et (x,d), qui forment un arbre enraciné en x de hauteur exactement h) ;
- $\psi(x,0) = \sigma(g,h-1) \times \sigma(d,h-1)$ (on ne garde aucune arête parmi (x,g) et (x,d), les deux enfants n'ont pas de sous-arbre de hauteur $\geq h$) ;
- pour $k > 0$, $\psi(x,k)$ est la somme de :
  - $\psi(g,k-1) \times \sigma(d,k-1) + \sigma(g,k-1) \times \psi(d,k-1)$ (on ne garde qu'une seule arête parmi (x,g) et (x,d), qui forme un arbre enraciné en x de hauteur exactement k) ;
  - $\psi(g,k-1) \times \sigma(d,k-2) + \sigma(g,k-2) \times \psi(d,k-1) + \psi(g,k-1) \times \psi(d,k-1)$ (on garde les deux arêtes (x,g) et (x,d), qui forment un arbre enraciné en x de hauteur exactement k) ;

Cela donne alors l'idée d'un algorithme de programmation dynamique. Il y a de l'ordre de $\Theta(|T| \times h)$ valeurs à calculer au maximum, et chaque valeur prend un temps $\mathcal{O}(h)$ à calculer. On obtient une complexité temporelle en $\mathcal{O}(|T|h^2)$ et spatiale en $\mathcal{O}(|T|h)$. À noter, on n'a pas pris en compte la complexité des opérations arithmétiques, qui n'est pas négligeable dans le cas général, sachant que les valeurs à calculer sont de l'ordre de $2^{|T|}$ au maximum. Cela rajouterait un facteur $|T|$ dans la complexité si on considérait des entiers de taille non bornée (si $|T| > 64$, par exemple).

4. Comment compter le nombre de sous-ensembles B tels que T_B contienne un arbre de hauteur **exactement** h ?

Solution :
Il suffit de faire deux appels à l'algorithme précédent, avec h et $h+1$, et de faire la différence entre les deux résultats obtenus. La complexité est la même.

Définition

On définit le **diamètre** d'une forêt F comme le plus grand entier $d \in \mathbb{N}^*$ tel qu'il existe un chemin simple non orienté de longueur d dans F.

5. Proposer un algorithme qui, étant donné T et d, calcule le nombre de sous-ensembles B tels que T_B soit de diamètre au moins d.

Solution :
On peut appliquer le même type d'algorithme que précédemment, en remarquant que :

- si on ne garde ni (x, g), ni (x, d), alors le diamètre maximal est celui dans l'un des enfants ;
- si on ne garde que (x, g) (resp. (x, d)), alors le diamètre maximal est le max entre le diamètre maximal dans g, dans d et $1 +$ la hauteur de l'arbre enraciné en g (resp. d) ;
- si on garde (x, g) et (x, d), le diamètre maximal est le max entre le diamètre maximal dans g, dans d et $2 +$ la somme des hauteurs des arbres enracinés en g et d.

On obtiendrait la même complexité.

Définition

Pour un alphabet Σ, on appelle **Σ-arbre** un arbre dont chaque arête est étiquetée par un élément de Σ. Étant donné un Σ-arbre T et un sous-ensemble d'arêtes B, on définit T_B comme précédemment ; il s'agit d'une forêt de Σ-arbres.

Étant donné un mot $u \in \Sigma^*$, on dit qu'une forêt F de Σ-arbres **contient** u s'il existe un chemin simple non-orienté dans F dont la séquence d'étiquette est égale à u.

6. Proposer un algorithme qui, étant donné un Σ-arbre T et un mot $u \in \Sigma^*$, calcule le nombre de sous-ensembles B tels que T_B contienne u.

Solution :
Même idée que précédemment, en gardant en mémoire l'existence de suffixes et préfixes de u qui apparaissent dans un sous arbre en terminant/commençant à la racine.

7. Proposer un algorithme qui, étant donné un Σ-arbre T et un langage rationnel L, calcule le nombre de sous-ensembles B tels que T_B contienne un mot de L.

Solution :
Même idée que précédemment, en gardant en mémoire les états accessibles et co-accessibles en lisant un mot de bas en haut ou de haut en bas (selon le type d'état).

8. On ne fait plus à présent l'hypothèse que les arbres sont binaires. Comment adapter les algorithmes précédents dans ce cas ? Comment la complexité évolue-t-elle en fonction du degré maximal des arbres ?

Solution :
On peut adapter les algorithmes précédents, mais les formules nécessite d'envisager, pour un degré d, les 2^d cas possibles d'arêtes conservées entre x et ses d enfants.

Exercice 24 (Théorie de Sprague-Grundy) :

Définition

Un **jeu** est un triplet (S, A, s_0) où S est un ensemble fini d'**états**, $A \subset S^2$ 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, $(s, \varphi(s)) \in T$. 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, i.e. qu'il existe $n_f \in \mathbb{N}$ tel que $\varphi_{n_f \bmod 2}(s_{n_f})$ n'est pas défini.

Solution :

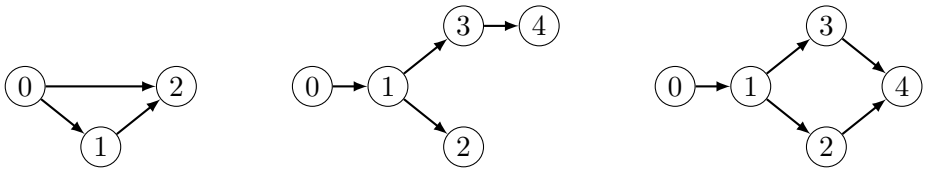
Si (s_k) n'est pas finie, alors par le principe des tiroirs, il existe $i < j$ tels que $s_i = s_j$. Cela signifie que le graphe contient un cycle par définition des stratégies. C'est contradictoire avec l'hypothèse. On en déduit par ailleurs que $n_f \leq |S|$.

Définition

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 ?



Solution :

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 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 ?

Solution :

On peut modéliser ce jeu par $J = (S, A, s_0)$ où :

- $S = [\![0, n]\!]$;
- $s_0 = n$;
- $A = \{(s, s - i) \mid s \in S, i \in [\![1, 4]\!], s - i \geq 0\}$.

Pour déterminer une stratégie gagnante, le mieux est de commencer par étudier de petites valeurs de n :

- si $n \leq 4$, 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.

Définition

Pour un jeu (S, A, s_0), on définit, pour chacun de ses états $s \in S$, sa **valeur de Grundy** par :

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

4. Justifier que cette relation définit bien G de manière unique.

Solution :

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 $\leq k$, 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. Donner une condition nécessaire et suffisante sur G pour que le joueur 0 ait une stratégie gagnante. Qu'en est-il du joueur 1 ?

Solution :

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. 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 ?

Solution :
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).

Algorithme : Stratégies gagnantes

Données : un jeu $J = (S, A, s_0)$, $n = |S|$
Résultat : un joueur ayant une stratégie gagnante
$G \leftarrow [-1, \dots, -1]$ (de taille n) ;
Fonction `Calcul_G(`s`)`
  si $G[s] = -1$ **alors**
    $m \leftarrow 0$;
    pour t *voisin de* s **faire**
      `Calcul_G` (t) ;
      $m \leftarrow m + 1$;
    $T \leftarrow$ tableau de taille $m + 1$ contenant des Faux ;
    pour t *voisin de* s **faire**
      $T[G[t]] \leftarrow$ Vrai ;
    pour i *de* 0 *à* m **faire**
      si $T[i]$ **alors**
        $G[s] \leftarrow i$;
        Sortir de la boucle ;
`Calcul_G` (s_0) ;
si $G[s_0] \neq 0$ **alors**
  retourner 0 ;
sinon
  retourner 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 $\mathcal{O}(|S|)$.

Dans le **jeu de Nim**, il y a n tas de $t_0, \dots, t_{n-1}$ 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 $t_0 \oplus \cdots \oplus t_{n-1} \neq 0$, où $\oplus$ représente le ou-exclusif (XOR) bit à bit (ou l'addition modulo 2, bit à bit).

Solution :
Montrons ce résultat par récurrence sur le nombre total k de jetons restant. Appelons **score** d'une configuration la valeur $t_0 \oplus \cdots \oplus t_{n-1}$.

- 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 $k + 1 - k_i \leq k$ 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.

Définition

Si $J_1 = (S_1, A_1, s_1)$ et $J_2 = (S_2, A_2, s_2)$ sont deux jeux, alors 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.

Solution :

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. Quelle est la valeur de Grundy des états du jeu de Nim ? Commenter.

Solution :

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 $t_0 \oplus \cdots \oplus t_{n-1}$, ce qui est cohérent avec les questions 5 et 7.