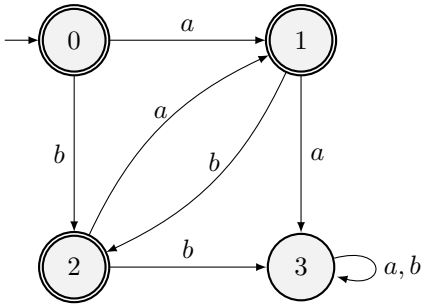
TD corrigé langages reconnaissables

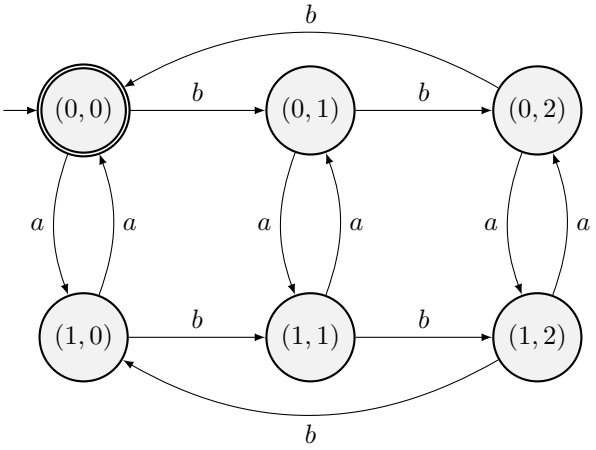
On rappelle qu'un langage est **reconnaissable** s'il est le langage des mots acceptés par un automate.
On montrera qu'un langage est reconnaissable si et seulement si il est rationnel, donc tous les résultats de ce TD sur les langages reconnaissables s'appliquent aussi aux langages rationnels.

I Petites questions

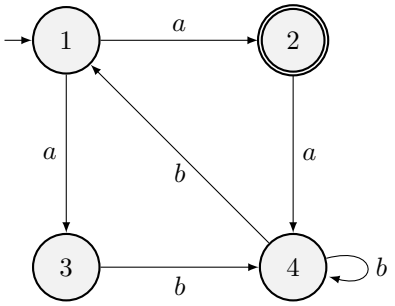
1. Dessiner un automate déterministe complet reconnaissant le langage des mots sur $\{a, b\}$ sans lettres consécutives égales.
   ► Idée: on se retrouve dans l'état 1 (respectivement 2) si la dernière lettre lue est un a (resp. b).



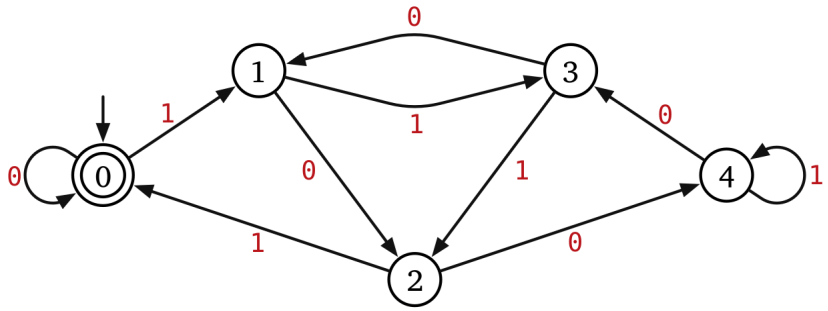
2. Dessiner un automate reconnaissant le langage des mots sur $\{a, b\}$ ayant un nombre pair de a et dont le nombre de b est multiple de 3.
   ► On peut construire un automate reconnaissant les mots ayant un nombre pair de a en utilisant 2 états (suivant la parité du nombre de a lus jusqu'à présent). De façon similaire, les mots ayant un nombre de b multiple de 3 peuvent être reconnus par un automate avec 3 états, pour chaque reste modulo 3 du nombre de b déjà lus.
   Pour vérifier les deux en même temps, on peut utiliser l'automate « produit » vu en cours, pour reconnaître l'intersection des deux langages précédents. L'état nommé (i, j) correspond à la lecture d'un mot dont le nombre de a est i modulo 2 et le nombre de b est j modulo 3:



3. Déterminiser l'automate suivant en utilisant l'algorithme du cours:



4. Montrer que le langage sur $\{0,1\}$ des écritures en base 2 des multiples de 5 est reconnaissable. Si $n \in \mathbb{N}$ est quelconque, le langage sur $\{0,1\}$ des écritures en base 2 des multiples de n est-il reconnaissable?
   ▶ Idée: construire un automate dont les états sont les restes possibles modulo 5. Si l'automate dans un état q a déjà lu $n_1...n_p$ (c'est à dire $< n_1...n_p >_2 \equiv q[5]$) et qu'il lit n_{p+1} alors il doit aller dans l'état $2q + n_{p+1}$ car $n_1...n_p n_{p+1} = 2\times < n_1...n_p >_2 + n_{p+1} \equiv 2q + n_{p+1}[5]$.
   Ce langage est donc reconnu par $A = (\{0,1\}, \{0,1,2,3,4\}, \{0\}, \{0\}, \delta)$ où $\delta(q,a) = (2q + a\ [5])$:



5. À quelle condition nécessaire et suffisante simple le langage reconnu par un automate est vide? Décrire un algorithme pour le savoir.
   ▶ Il est vide si et seulement si il n'existe pas de chemin d'un état initial vers un état final. On peut le décider en effectuant un parcours du graphe de l'automate.

6. À quelle condition nécessaire et suffisante simple le langage reconnu par un automate est fini? Décrire un algorithme pour le savoir.
   ▶ Il est fini si et seulement si son graphe n'a pas de cycle. Nous avons vu un algorithme pour cela dans le cours sur les graphes.

7. Décrire un algorithme pour déterminer si deux automates admettent le même langage.
   ▶ Soient $A_1 = (\Sigma, Q_1, i_1, F_1, \delta_1)$ et $A_2 = (\Sigma, Q_2, i_2, F_2, \delta_2)$. On peut construire des automates A et A' reconnaissant $L(A_1)\backslash L(A_2)$ et $L(A_2)\backslash L(A_1)$, en utilisant « l'automate produit » décrit dans le cours. Il suffit alors de déterminer si $L(A) = \emptyset$ et $L(A') = \emptyset$, en utilisant la question 3.

8. Soit A un automate à n états. Montrer que si $L(A)$ est non vide alors il contient un mot de longueur $\leq n-1$.
   ▶ Soit $m = m_1 m_2 ... m_k \in L(A)$ un mot de longueur minimum accepté par A. Soit $q_1 \xrightarrow{m_1} q_2 \xrightarrow{m_2} ... \xrightarrow{m_k} q_{k+1}$ un chemin acceptant de A dont l'étiquette est m. Supposons $k \geq n$. q_1, ..., q_{k+1} ne peuvent pas tous être différents (car il n'y a que n états), disons $q_i = q_j$ avec $i < j$. Alors $q_1 \xrightarrow{m_1} q_2 \xrightarrow{m_2} ... \xrightarrow{m_i} q_i = q_j \xrightarrow{m_{j+1}} ... \xrightarrow{m_k} q_{k+1}$ est un chemin d'un état initial à un état final donc $m_1...m_i m_{j+1}...m_k$ est un mot accepté de longueur strictement inférieure à k, ce qui est absurde.

9. On dit qu'un automate est **émondé** si, pour tout état q, il existe d'une part un chemin d'un état initial à q et d'autre part un chemin de q à un état final. Montrer que tout automate est équivalent à un automate émondé.
   ▶ Il suffit d'enlever de l'automate tous les états qui ne conviennent pas (ainsi que toutes les transitions vers/depuis ces états). Comme un chemin acceptant ne peut clairement pas passer par un de ces états, ceci ne change pas les mots acceptés par l'automate.

II Clôture des langages reconnaissables

Si $m = m_1...m_n$ est un mot, on définit son miroir $\widetilde{m} = m_n...m_1$.
Si L est un langage, on définit son miroir $\widetilde{L} = \{\widetilde{m} \mid m \in L\}$.

1. Montrer que le miroir d'un langage reconnaissable est reconnaissable.
   ▶ Soit $A = (\Sigma, Q, I, F, E)$ un langage reconnaissant L. Alors $\widetilde{A} = (\Sigma, Q, F, I, \widetilde{E})$ reconnaît $\widetilde{A}$, où on a inversé toutes les transitions ($\widetilde{E} = \{(q_1, a, q_2) \mid (q_2, a, q_1) \in E\}$). En effet il existe un chemin $q_1 \in I \xrightarrow{m_1} q_2 \xrightarrow{m_2} ... \xrightarrow{m_k} q_{k+1} \in F$ dans A si et seulement si il existe un chemin $q_{k+1} \in F \xrightarrow{m_k} q_k \xrightarrow{m_{k-1}} ... \xrightarrow{m_1} q_1 \in I$ dans $\widetilde{A}$.

Si L est un langage sur Σ, on définit:

- $Pref(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^*,\ uv \in L\}$: ensemble des préfixes des mots de L.
- $Suff(L) = \{u \in \Sigma^* \mid \exists v \in \Sigma^*,\ vu \in L\}$: ensemble des suffixes des mots de L.
- $Fact(L) = \{u \in \Sigma^* \mid \exists v, w \in \Sigma^*,\ vuw \in L\}$: ensemble des facteurs des mots de L.

2. Montrer que si L est reconnaissable alors $Pref(L)$, $Suff(L)$, $Fact(L)$ le sont aussi. Décrire un algorithme pour obtenir les automates correspondants.
   ▶ Soit $A = (\Sigma, Q, I, F, E)$ un langage reconnaissant L. Soit Q' l'ensemble des états à partir desquels un état final est atteignable (par un chemin). Un mot m appartient à $Pref(L)$ si et seulement si il existe un chemin étiqueté par m d'un état de I vers un état de Q'. Donc (Σ, Q, I, Q', E) reconnaît $Pref(L)$. De même, (Σ, Q, Q', F, E) reconnaît

$Suff(L)$ et (Σ, Q, Q', Q', E) reconnaît $Fact(L)$.

Autre solution: après avoir démontré que $Pref(L)$ est reconnaissable, on peut remarquer que $Suff(L) = \widetilde{Pref(\widetilde{L})}$ ($m \in \widetilde{Pref(\widetilde{L})} \iff \widetilde{m} \in Pref(\widetilde{L}) \iff \exists v \in \Sigma^*,\ \widetilde{m}v \in \widetilde{L} \iff \exists v \in \Sigma^*,\ \widetilde{v}m \in L \iff m \in Suff(L)$, où on a utilisé le fait que $\widetilde{xy} = \widetilde{y}\,\widetilde{x}$).
On peut aussi en déduire que $Fact(L)$ est reconnaissable en remarquant que $Fact(L) = Suff(Pref(L))$ $(= Pref(Suff(L)))$.
En effet $m \in Suff(Pref(L)) \iff \exists u \in \Sigma^*,\ um \in Pref(L) \iff \exists u \in \Sigma^*,\ \exists v \in \Sigma^*,\ umv \in L \iff m \in Fact(L)$.

3. Montrer que si L est rationnel alors $Pref(L)$, $Suff(L)$, $Fact(L)$ le sont aussi (puisqu'on va montrer que rationnel = reconnaissable, c'est une preuve alternative à la précédente).
► Soit e est une expression rationnelle dont le langage est L. On définit par induction une expression rationnelle $P(e)$ de langage $Pref(L)$:
   - Si $e = a \in \Sigma$: $P(e) = \varepsilon + a$.
   - Si $e = e_1 + e_2$: $P(e) = P(e_1) + P(e_2)$.
   - Si $e = e_1 e_2$: $P(e) = P(e_1) + e_1 P(e_2)$.
   - Si $e = e_1^*$: $P(e) = e_1^* P(e_1)$.

   De même pour $Suff(L)$ et $Fact(L)$.

III Lemme de l'étoile

1. Montrer le **lemme de l'étoile**: si L est un langage reconnaissable alors il existe un entier k tel que tout mot de L de longueur supérieure ou égale à k s'écrit uvw avec $|v| \geq 1$ et $\forall n \in \mathbb{N}$, $uv^n w \in L$.
► Soit A un automate reconnaissant L et k son nombre d'états. Soit $m \in L$ de longueur $\geq k$, accepté par un chemin $q_1 \xrightarrow{m_1} q_2 \xrightarrow{m_2} \dots \xrightarrow{m_k} q_{k+1} \xrightarrow{m_{k+1}} \dots \xrightarrow{m_n} q_{n+1}$. q_1, ..., q_{k+1} ne peuvent pas tous être différents (car il n'y a que n états): disons $q_i = q_j$ avec $i < j$. Alors la portion du chemin de q_i à q_j est un cycle, que l'on peut répéter un nombre quelconque de fois (éventuellement 0 fois, ce qui revient à enlever le cycle) et obtenir un chemin acceptant. Le lemme est donc vérifié avec $u = m_1 ... m_{i-1}$, $v = m_i ... m_{j-1}$, $w = m_j ... m_n$.

2. Montrer que $L_1 = \{a^n b^n \mid n \in \mathbb{N}\}$, $L_2 = \{m \in a, b^* \mid |m|_a = |m|_b\}$ (où $|m|_a$ est le nombre de a de m) et $L_3 = \{a^p \mid p \text{ est premier}\}$ ne sont pas reconnaissables.
► Supposons que L_1 soit reconnaissable. Soit k l'entier donné par le lemme de l'étoile. $a^k b^k \in L_1$ donc, d'après le lemme de l'étoile, il existe des mots u, v, w tels que $a^k b^k = uvw$ et $\forall n$, $uv^n w \in L_1$. Si v ne contient que des a alors $uv^2 w$ contient plus de a que de b, ce qui est impossible. De même si v ne contient que des b. Si v contient des a et des b alors il est de la forme $v = a^k b^p$ (avec $k \geq 1$, $p \geq 1$) et alors $v^2 = a^k b^p a^k b^p \notin L_1$: c'est absurde. L_1 ne vérifie pas le lemme de l'étoile donc il ne peut pas être reconnaissable.
Si L_2 était reconnaissable alors, comme l'intersection de langages reconnaissables est reconnaissable (cf cours), $L(a^*b^*) \cap L_2 = L_1$ serait reconnaissable, ce qui n'est pas le cas. Donc L_2 n'est pas reconnaissable.

3. Soit $\alpha \in \mathbb{R} - \mathbb{Q}$ et $L(\alpha)$ l'ensemble des préfixes des décimales de α. Montrer que $L(\alpha)$ n'est pas reconnaissable.
►

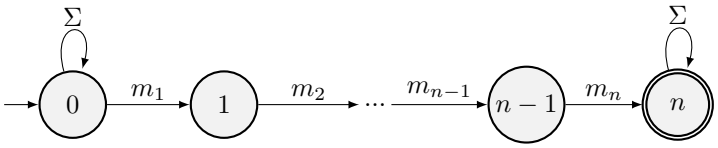
IV Algorithme KMP

On s'intéresse à la recherche d'un facteur m dans un texte, sur un alphabet Σ.

1. Écrire une fonction naïve de type `'a list -> 'a list -> bool` pour résoudre ce problème.
Quelle est sa complexité?
► On peut utiliser une fonction auxiliaire `prefixe` determinant si un mot `l1` est préfixe de `l2`, linéaire en la taille de `l1`. `facteur m texte` appelle |`texte`| fois `prefixe m` donc est en complexité O(|`texte`| |`m`|).

```
let rec prefixe l1 l2 = match l1, l2 with
  | [], _ -> true
  | _, [] -> false
  | e1::q1, e2::q2 -> e1 = e2 && prefixe q1 q2;;
let rec facteur m texte = match texte with
  | [] -> false
  | e::q -> prefixe m texte || facteur m q;;
```

2. Expliquer comment construire un automate déterministe complet reconnaissant les mots contenant $m = m_1 ... m_n$ comme facteur.
Une fois cet automate construit, quelle est la complexité pour savoir si un mot contient m comme facteur?
► On peut déterminiser l'automate suivant reconnaissant $\Sigma^* m \Sigma^*$ (une transition étiqueté par Σ signifie qu'il y a une transition pour chaque lettre de Σ), ce qui peut être de complexité exponentiel (cf exemple du cours):



Par contre, une fois l'automate déterministe complet construit, on peut savoir si un texte appartient à son langage (c'est à dire: savoir si le texte contient m) en $O(|m|)$, puisqu'il suffit de suivre la seule transition possible pour chaque lettre de m, et de regarder si on aboutit dans un état final.

On note $Pref(m)$ l'ensemble des préfixes de m.
L'algorithme KMP (Knuth-Morris-Pratt) consiste à considérer l'automate $K = (\Sigma, Pref(m), \varepsilon, \{m\}, \delta)$, où l'état initial est ε, l'état final est m et $\delta(p, a)$ est le plus long suffixe du mot pa qui est aussi préfixe de m.

3. Dessiner K si $m = 01000$ (avec $\Sigma = \{0, 1\}$). Quel est le langage reconnu par K? Comment modifier K pour qu'il reconnaisse $\Sigma^* m \Sigma^*$?

4. Écrire une fonction `delta : 'a list -> 'a list -> 'a list` telle que `delta u m` renvoie le plus long suffixe de `u` qui est préfixe de `m`.
   ▶

```
let rec delta u m = match u with
  | [] -> []
  | e::q -> if prefixe u m then u
    else delta q m;;
```

5. Comparer la complexité de l'algorithme KMP avec les deux méthodes précédentes.

Il est possible de calculer δ plus rapidement pour construire l'automate K, avec une complexité $O(|m|)$.

V Résiduel

Soit L un langage sur un alphabet Σ. Soit $u \in \Sigma^*$. On définit le langage $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$ (qu'on appelle résiduel de L).

1. Montrer que si L est reconnaissable alors $u^{-1}L$ est reconnaissable.
   ▶ Soit $A = (\Sigma, Q, i, F, \delta)$ un automate déterministe complet reconnaissant L (où i est l'état initial) et q_u l'état obtenu en lisant u dans A (c'est à dire $q_u = \delta^*(i, u)$). Alors: $m \in u^{-1}L \iff um \in L \iff$ dans A, il existe un chemin étiqueté par um de i vers un état final $\iff$ dans A, il existe un chemin étiqueté par m de q_u vers un état final. Donc $(\Sigma, Q, q_u, F, \delta)$ reconnaît $u^{-1}L$.
   Remarque: il n'est pas obligatoire de considérer un automate déterministe complet, mais cela simplifie légèrement la preuve.

2. Quels sont tous les résiduels possibles de a^*b^*? De $\{a^n \mid n \text{ est pair}\}$?
   ▶ Soit u un mot sur $\{a, b\}$. Si u contient un facteur ba alors $u^{-1}L(a^*b^*) = \emptyset$. Si u ne contient que des a alors $u^{-1}L(a^*b^*) = L(a^*b^*)$. Si u ne contient un certain nombre de a suivit d'un certain nombre de b (mais au moins un b), alors $u^{-1}L(a^*b^*) = L(b^*)$. Les résiduels possibles de $L(a^*b^*)$ sont donc $\emptyset$, $L(b^*)$, $L(a^*b^*)$.
   Les résiduels possibles de $\{a^n \mid n \text{ est pair}\}$ sont $\{a^n \mid n \text{ est pair}\}$ et $\{a^n \mid n \text{ est impair}\}$ (suivant que u contienne un nombre pair ou impair de a).

3. Montrer que si L est reconnaissable alors $\{u^{-1}L \mid u \in \Sigma^*\}$ est fini (il n'y a qu'un nombre fini de valeurs possibles pour $u^{-1}L$ quand u varie dans Σ^*).
   ▶ Soit $A = (\Sigma, Q, i, F, \delta)$ un automate déterministe complet reconnaissant L (le même que pour la question précédente). Soit $n = |Q|$. Supposons qu'il existe $n+1$ résiduels différents, notés $u_0^{-1}L$, $u_1^{-1}L$, ..., $u_n^{-1}L$. Comme il y a n états dans A, les états $\delta^*(i, u_0), ..., \delta^*(i, u_n)$ ne peuvent pas tous être différents: disons par exemple $\delta^*(i, u_k) = \delta^*(i, u_\ell)$. Mais alors $u_k^{-1}L$ et $u_\ell^{-1}L$ sont reconnus par le même automate donc sont égaux, ce qui est absurde. Donc il y a au plus $|Q|$ langages résiduels différents.

4. Montrer que $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas reconnaissable.
   ▶ Soit $k \in \mathbb{N}$. Alors $b^k \in (a^k)^{-1}L$ mais $\forall p \neq k$, $b^k \notin (a^p)^{-1}L$. Donc L possède une infinité de résiduels différents: $(a^k)^{-1}L$ pour $k \in \mathbb{N}$. Par contraposition de la question précédente, L n'est donc pas reconnaissable.

Un mot m est un palindrome s'il se lit de la même façon dans les deux sens (ou encore: $\widetilde{m} = m$).

5. Écrire une fonction Caml pour déterminer si une liste de lettres est un palindrome. Complexité?
   ▶ `let palindrome m = List.rev m = m;;`

6. Montrer que l'ensemble des palindromes (sur un alphabet à au moins 2 lettres) n'est pas reconnaissable.
   ▶ Soit L le langage des palindromes sur un alphabet Σ et a, b deux lettres de Σ. Alors $a^k bb \in (a^k)^{-1}L$ mais $\forall p \neq k$, $a^k bb \notin (a^p)^{-1}L$. Donc L a une infinité de résiduels différents donc n'est pas reconnaissable.

VI Automate minimal

Soit $A = (\Sigma, Q, q_i, F, \delta)$ un automate déterministe complet de langage L. On veut trouver un automate déterministe équivalent à A avec un nombre minimum d'états.
Soit $\widetilde{Q} = \{u^{-1}L \mid u \in \Sigma^*\}$ l'ensemble des résiduels de L et $\widetilde{A} = (\Sigma, \widetilde{Q}, \varepsilon^{-1}L, \widetilde{F}, \widetilde{\delta})$ où $\widetilde{F}$ est l'ensemble des résiduels contenant le mot vide ε et $\widetilde{\delta}(u^{-1}L, a) = (ua)^{-1}L$.

1. Montrer que $u^{-1}L = v^{-1}L \Longrightarrow \forall a \in \Sigma,\ (ua)^{-1}L = (va)^{-1}L$. Ceci montre que δ est bien défini.
   ▶ $m \in (ua)^{-1}L \Longleftrightarrow uam \in L \Longleftrightarrow am \in u^{-1}L \Longleftrightarrow am \in v^{-1}L \Longleftrightarrow vam \in L \Longleftrightarrow m \in (va)^{-1}L$.

2. Montrer que $\widetilde{A}$ est un automate déterministe complet équivalent à A avec un nombre minimum d'états.
   ▶ Par définition de $\widetilde{\delta}$, il est évident que $\widetilde{A}$ est déterministe complet. De plus, il est clair qu'en lisant un mot m dans $\widetilde{A}$ depuis l'état initial, on se retrouve dans l'état $m^{-1}L$ (on pourrait le démontrer par récurrence sur $|m|$, en utilisant la définition de $\widetilde{\delta}$). Donc $m \in L(\widetilde{A}) \Longleftrightarrow m^{-1}L \in \widetilde{F} \Longleftrightarrow \varepsilon \in m^{-1}L \Longleftrightarrow m \in L$. Donc $\widetilde{A}$ a bien le même langage que A.
   Soient $u_1^{-1}L$, u_2^{-1}, ... $u_n^{-1}L$ les différents résiduels de L. Alors $\delta^*(q_i, u_1)$, ..., $\delta^*(q_i, u_n)$ sont des états différents ($\delta^*(q_i, u_k) = \delta^*(q_i, u_p) \Longrightarrow u_k^{-1}L = u_p^{-1}L$) donc tout automate reconnaissant L doit avoir au moins n états.

3. Montrer que l'automate K obtenu par l'algorithme KMP (cf exercice ci-dessus) est un automate minimal reconnaissant $\Sigma^* m \Sigma^*$.
   ▶ Soit $L = \Sigma^* m \Sigma^*$. Alors les résiduels de K sont exactement les $u^{-1}L$ pour $u \in Pref(m)$. Il y en a donc autant que d'états dans K, ce qui montre que le nombre d'états de K est minimum.

On veut un algorithme pour construire $\widetilde{A}$. Pour cela on numérote les états dans Q par des entiers de 0 à $n-1$ et on va calculer une matrices de booléens `diff` indicée par les états tel que:

$$\texttt{diff}.(q_1).(q_2) \Longleftrightarrow \exists m \in \Sigma^*, \text{ exactement un des états } \delta^*(q_1,\, m) \text{ et } \delta^*(q_2,\, m) \text{ est final}$$

On rappelle que $\delta^*(q,\, m)$ est l'état obtenu (unique si l'automate est déterministe complet) en lisant le mot m à partir de l'état q.

4. Écrire en pseudo-code un algorithme par programmation dynamique pour calculer $\texttt{diff}.(q_1).(q_2)$, $\forall q_1, q_2 \in Q$.
   ▶ Soit $\texttt{diff}.(q_1).(q_2).(k) \Longleftrightarrow \exists m \in \Sigma^*$ de taille au plus k tel que exactement un des états $\delta^*(q_1,\, m)$ et $\delta^*(q_2,\, m)$ soit final.
   Alors $\texttt{diff}.(q_1).(q_2).(k) = \texttt{diff}.(q_1).(q_2).(k-1) \vee \bigvee_{a \in \Sigma} \texttt{diff}.(\delta(q_1, a)).(\delta(q_2, a)).(k-1)$ ($\vee$ étant le « ou » logique).

5. Si $\texttt{diff}.(q_1).(q_2) = \texttt{false}$, comment peut-on regrouper q_1 et q_2 en un seul état pour obtenir un automate équivalent qui soit toujours déterministe complet?
   ▶

En regroupant tant que possible avec la question précédente, on obtient un automate dont on peut montrer qu'il est minimal.