Second Concours à l'ENS Paris-Saclay et l'ENS Rennes
Interrogation orale d'informatique
Déroulement de l'épreuve

Le sujet se compose de deux problèmes indépendants. Le jury recommande de traiter les deux sujets, au moins partiellement.

Le temps de préparation est de 2 heures. Ensuite le candidat exposera les résultats obtenus pendant une heure à l'oral. Bien entendu, le jury posera des questions et pourra revenir sur des questions omises par le candidat.

Il est autorisé d'admettre le résultat d'une question pour ne pas rester bloqué lors de la phase de préparation.

Il est enfin demandé au candidat de mentionner en début d'oral quelles questions il a traitées dans les deux problèmes. Cela permettra au jury de gérer au mieux le temps et de permettre au candidat de présenter l'intégralité des résultats obtenus lors de la préparation.

La rigueur du raisonnement scientifique est un point capital dans l'évaluation de l'épreuve.

Graphe d'intervalles

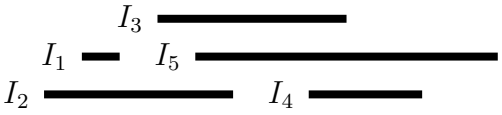
Un **graphe non orienté** G est un couple (V, E) où V est un ensemble de sommets et E un ensemble d'arêtes. Notons $n = |V|$, $m = |E|$. Les sommets sont numérotés de 1 à n. Les sommets u et v sont dits **voisins** s'il y a une arête entre u et v.

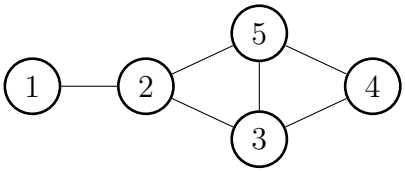
Un graphe est un **graphe d'intervalles** si

1. chaque sommet v est associé à un intervalle I_v sur une droite réelle ayant une date de début d_v et une date de fin f_v ;
2. deux sommets u et v sont reliés par une arête si et seulement si les intervalles correspondant à u et v s'intersectent ($I_u \cap I_v \neq \emptyset$).

Par la suite, nous supposons que toutes les dates de début et de fin sont différentes.

Voici un exemple de graphes d'intervalle dont la numérotation des sommets suit l'ordre croissant de la date de fin de l'intervalle.





Problème de clique. On dit qu'un sous-ensemble de sommets $K \subset V$ d'un graphe $G = (V, E)$ est une **clique** s'il existe une arête entre toute paire de sommets de K dans G. La **taille** de K est le nombre de sommets de K. Une clique K est **maximum** s'il n'existe pas de clique K' de taille strictement supérieure à celle de K.

Question 1. Proposer un algorithme calculant une clique maximum en $O(n \log n)$ opérations, dans un graphe d'intervalles représenté par les intervalles.

Un **ensemble indépendant** S de G est un ensemble de sommets deux à deux non voisins dans G. La taille d'un ensemble indépendant est égale au nombre de sommets qu'il contient. Nous cherchons un ensemble indépendant de G de taille maximum. Par la suite, nous notons S^* une solution optimale (un ensemble indépendant de taille maximum). Considérons l'algorithme glouton suivant, dépendant d'un ordre $\mathcal{T}$ des sommets :

Algorithme 1 : Glouton(G)

Données : Un graphe d'intervalles $G = (V, E)$
Résultat : Un sous-ensemble de sommets
$S \leftarrow \emptyset$; $Q \leftarrow V$;
tant que $(Q \neq \emptyset)$ **faire**
  Choisir un sommet v de Q minimal dans l'ordre $\mathcal{T}$;
  si $\{v\} \cup S$ *reste un ensemble indépendant de* G **alors**
    $S \leftarrow \{v\} \cup S$;
    Supprimer les voisins de v dans Q ;
  fin
fin
Retourner S

Question 2. Donner un exemple où l'algorithme ne retourne pas la solution optimale

1. si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant des dates de début d_v ;
2. si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant sur la longueur des intervalles $f_v - d_v$.

Question 3. Soit S la solution retournée par l'algorithme si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant sur la longueur des intervalles $f_v - d_v$.

1. Montrer que chaque sommet j de S est voisin d'au plus deux sommets de S^*.
2. Montrer que $|S^*| \leq 2|S|$.

Notons j le sommet de G telle que sa date de fin f_j soit la plus petite possible.

Question 4. Montrer qu'il existe une solution optimale qui contient le sommet j.

Question 5. Montrer que l'algorithme retourne une solution optimale si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant de la date de fin.

Question 6. Simplifier l'algorithme glouton si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant de date de fin en supposant donné le graphe par les tableaux de dates de début et de fin. Évaluer sa complexité.

Nous associons une fonction de poids positif sur chaque sommet $w : V \to \mathbb{N}$. Nous cherchons un sous-ensemble indépendant S qui maximise la somme $\sum_{u \in S} w(u)$ de tous les poids de ces sommets. Nous notons S^* un ensemble indépendant de G qui maximise cette quantité.

Convention : À partir de maintenant, nous supposons que les sommets sont numérotés de 1 à n en fonction de leur date de fin f_v : $i < j$ signifie que $f_i < f_j$.

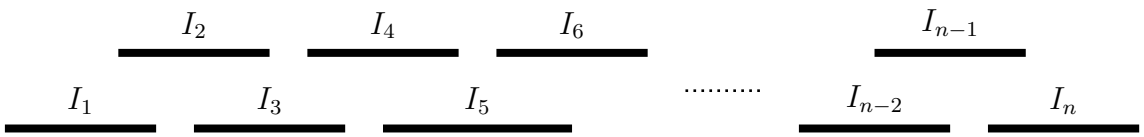
Notons $p(j)$ le plus grand sommet dont l'intervalle finit avant d_j. S'il n'en existe pas, nous supposons que $p(j) = 0$. Dans l'exemple, $p(1) = p(2) = 0$, $p(3) = p(5) = 1$ et $p(4) = 2$.

Question 7. Donner un algorithme en $O(n \log n)$ opérations qui calcule la fonction p pour tous les sommets.

Question 8. Montrer que si $j \in S^*$, alors $S^* \cap \{1, \ldots, p(j)\}$ est un ensemble indépendant de G qui maximise la somme de tous les poids de ces sommets dans le sous-graphe de G ayant les sommets 1 à $p(j)$.

Question 9. Notons $OPT(j)$ le poids d'une solution optimale dans le sous-graphe ayant les sommets 1 à j. Donner une formule de récurrence pour $OPT(j)$

Question 10. Écrire un algorithme récursif qui calcule $OPT(n)$. Appliquer l'algorithme récursif sur l'instance de la figure suivante. Évaluer la complexité de cet algorithme dans le cas général.



Question 11. Comment réduire la complexité de l'algorithme précédent ? Donner sa complexité après amélioration.

Dérivée d'expressions rationnelles

Soit A un alphabet fini et non vide. On rappelle qu'une **expression rationnelle** sur A est toute formule sur l'alphabet $A \uplus \{0, 1, +, \cdot, *, (,)\}$ qui peut être obtenue de la manière suivante :

— 0, 1 et a, pour tout a dans A, sont des expressions rationnelles ;
— si E et F sont deux expressions rationnelles, alors $(E + F)$, $(E \cdot F)$ et (E^*) sont des expressions rationnelles.

Comme habituellement, on s'autorise à supprimer des parenthèses, grâce à la convention de priorité : $*$ a priorité sur $\cdot$ qui a priorité sur $+$. Le langage $\mathcal{L}(E)$ dénoté par l'expression rationnelle E est défini de manière inductive :

— $\mathcal{L}(0) = \emptyset$, $\mathcal{L}(1) = \{\varepsilon\}$, $\mathcal{L}(a) = \{a\}$ pour tout a dans A ;
— $\mathcal{L}(E + F) = \mathcal{L}(E) \cup \mathcal{L}(F)$,
— $\mathcal{L}(E \cdot F) = \mathcal{L}(E) \cdot \mathcal{L}(F)$, avec $X \cdot Y = \{x \cdot y \mid x \in X, y \in Y\}$, pour tous langages X, Y,
— $\mathcal{L}(E^*) = \mathcal{L}(E)^*$ avec $X^* = \{x_1 \cdots x_n \mid n \geq 0, x_1, \ldots, x_n \in X\}$, pour tout langage X.

Un langage est dit rationnel s'il s'écrit $\mathcal{L}(E)$ pour une expression rationnelle E. Le **terme constant** $c(L)$ d'un langage L sur l'alphabet A vaut $c(L) = 1$ si $\varepsilon \in L$ et $c(L) = 0$ sinon.

1. Donner une définition inductive du terme constant $c(E)$ d'une expression rationnelle E qui satisfait $c(E) = c(\mathcal{L}(E))$.

Dans ce sujet, on va s'intéresser à des algorithmes pour transformer une expression rationnelle en un automate fini reconnaissant le langage dénoté par l'expression. On rappelle brièvement la définition des automates finis que nous allons considérer. Un **automate fini** $\mathcal{A}$ sur l'alphabet A est la donnée d'un tuple $\mathcal{A} = \langle Q, \delta, I, T\rangle$ où Q est un ensemble fini d'états, $\delta \subseteq Q \times A \times Q$ est l'ensemble des transitions, $I \subseteq Q$ est l'ensemble des états initiaux et $T \subseteq Q$ est l'ensemble des états terminaux. La transition $(p, a, q) \in \delta$ est notée par la flèche $p \xrightarrow{a} q$. Un **calcul** de $\mathcal{A}$ est une séquence finie de transitions $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$: on note un tel calcul $p_0 \xrightarrow{a_1 a_2 \cdots a_n} p_n$. Son étiquette est le mot $a_1 a_2 \cdots a_n$ de A^*. Le calcul est acceptant si $p_0 \in I$ et $p_n \in T$. Le **langage** de $\mathcal{A}$ est le sous-ensemble $\mathcal{L}(\mathcal{A})$ de A^* qui contient l'ensemble des étiquettes des calculs acceptants de $\mathcal{A}$. L'automate $\mathcal{A}$ est dit **déterministe** si I possède un unique élément et pour tout $p \in Q$ et $a \in A$, il existe au plus un état $q \in Q$ tel que $(p, a, q) \in T$.

2. Soit $E_1 = (a+b)^* \cdot \Big(a \cdot \big(b \cdot (a+b)^*\big)\Big)$. Donner un automate fini reconnaissant le langage $\mathcal{L}(E_1)$.

On va étudier un algorithme de dérivation pour la construction d'un automate fini reconnaissant le langage dénoté par une expression rationnelle. On définit la dérivée $\frac{\partial}{\partial a}E$ d'une expression rationnelle E par une lettre $a \in A$ par induction :

— $\frac{\partial}{\partial a}0 = \frac{\partial}{\partial a}1 = \frac{\partial}{\partial a}b = 0$ pour tout $b \neq a$ dans A et $\frac{\partial}{\partial a}a = 1$;
— $\frac{\partial}{\partial a}(E + F) = \frac{\partial}{\partial a}E + \frac{\partial}{\partial a}F$, $\frac{\partial}{\partial a}(E \cdot F) = \left[\frac{\partial}{\partial a}E\right] \cdot F + c(E) \cdot \frac{\partial}{\partial a}F$, $\frac{\partial}{\partial a}E^* = \left[\frac{\partial}{\partial a}E\right] \cdot E^*$.

On s'autorise les identités triviales suivantes (c'est-à-dire qu'on s'autorise toujours à remplacer une expression par son expression réduite équivalente) pour simplifier les expressions

rationnelles :

$$E + 0 \equiv 0 + E \equiv E, \qquad E \cdot 0 \equiv 0 \cdot E \equiv 0, \qquad E \cdot 1 \equiv 1 \cdot E \equiv E \tag{\textbf{Tr}}$$

3. Calculer $\frac{\partial}{\partial a} E_1$ et $\frac{\partial}{\partial b} E_1$.

Étant donnés un mot $u \in A^*$ et un langage $L \subseteq A^*$, on appelle **quotient** de L par u, et on note $u^{-1}L$, le langage $u^{-1}L = \{v \in A^* \mid uv \in L\}$.

4. Donner une définition par récurrence de la dérivée d'une expression rationnelle E par un mot u, notée $\frac{\partial}{\partial u} E$, qui satisfait $\mathcal{L}(\frac{\partial}{\partial u} E) = u^{-1}\mathcal{L}(E)$. On démontrera rapidement cette égalité. *[Indication : il s'agit donc de définir $\frac{\partial}{\partial \varepsilon} E$, ainsi que $\frac{\partial}{\partial u} E$ en fonction de $\frac{\partial}{\partial u'} E$ pour u' un mot de longueur inférieure à celle de u.]*

5. Sans propriétés supplémentaires (par exemple l'associativité de $+$ et $\cdot$), démontrer que E_1 possède un nombre infini de dérivées différentes par rapport à des mots de A^*.

On ajoute les identités suivantes :

$$(E + F) + G \equiv E + (F + G) \tag{$\mathbf{A_s}$}$$
$$E + F \equiv F + E \tag{\textbf{C}}$$
$$E + E \equiv E \tag{\textbf{I}}$$

6. Montrer que l'ensemble des dérivées d'une expression rationnelle (par rapport à des mots de A^*) est fini si l'on s'autorise d'utiliser les identités ($\mathbf{A_s}$), (**C**) et (**I**).

7. En déduire que l'ensemble D_E des dérivées d'une expression rationnelle E, où l'on s'autorise d'utiliser ($\mathbf{A_s}$), (**C**) et (**I**), est effectivement calculable et qu'on peut ainsi construire un automate fini déterministe $\mathcal{A}_E$ reconnaissant le langage $\mathcal{L}(E)$. Construire cet automate pour l'expression E_1 de la question **2**.

On peut observer que l'automate $\mathcal{A}_E$ est certes déterministe, mais pas nécessairement minimal, comme le montre l'exemple de l'expression E_1. Pour éviter une explosion exponentielle, on s'intéresse maintenant à obtenir un automate $\mathcal{B}_E$ non déterministe à partir d'une définition alternative de dérivées d'expressions. Ainsi, on définit, comme un ensemble d'expression, la **B**-dérivée $\frac{\partial'}{\partial' a} E$ d'une expression rationnelle E par une lettre $a \in A$ par induction :

- $\frac{\partial'}{\partial' a} 0 = \frac{\partial'}{\partial' a} 1 = \frac{\partial'}{\partial' a} b = \emptyset$ pour tout $b \neq a$ dans A et $\frac{\partial'}{\partial' a} a = \{1\}$;
- $\frac{\partial'}{\partial' a}(E + F) = \frac{\partial'}{\partial' a} E \cup \frac{\partial}{\partial a} F$;
- $\frac{\partial'}{\partial' a}(E \cdot F) = \left[\frac{\partial}{\partial a} E\right] \cdot \{F\} \cup \{c(E)\} \cdot \frac{\partial}{\partial a} F$;
- $\frac{\partial'}{\partial' a} E^* = \left[\frac{\partial}{\partial a} E\right] \cdot \{E^*\}$.

Tous les calculs sont menés modulo les identités triviales (**Tr**).

8. Étendre, comme dans la question **4**, la notion de **B**-dérivée pour définir la **B**-dérivée $\frac{\partial'}{\partial' u} E$ par rapport à un mot $u \in A^*$. On appelle terme dérivé de E tout expression différente de 0 qui apparait dans au moins un ensemble $\frac{\partial'}{\partial' u} E$ pour $u \in A^*$. Donner, en la prouvant, une borne sur le nombre de termes dérivés d'une expression E.

9. Donner alors la définition d'un automate non-déterministe $\mathcal{B}_E$ qui reconnait le langage $\mathcal{L}(E)$, et dont les états sont les termes dérivés de E auxquels on ajoute E (s'il n'est pas déjà présent). Construire $\mathcal{B}_{E_1}$ pour l'expression E_1 de la question **2**.

10. Vérifier que, pour toute expression rationnelle E, l'automate $\mathcal{A}_E$ est l'automate déterminisé de l'automate $\mathcal{B}_E$.

Graphe d'intervalles

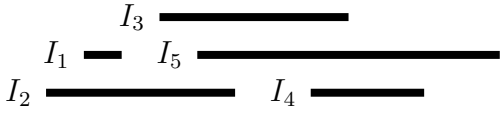
Un **graphe non orienté** G est un couple (V, E) où V est un ensemble de sommets et E un ensemble d'arêtes. Notons $n = |V|$, $m = |E|$. Les sommets sont numérotés de 1 à n. Les sommets u et v sont dits **voisins** s'il y a une arête entre u et v.

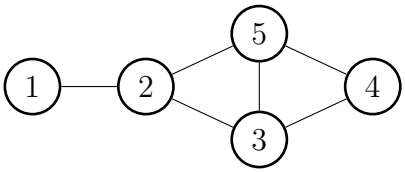
Un graphe est un **graphe d'intervalles** si

1. chaque sommet v est associé à un intervalle I_v sur une droite réelle ayant une date de début d_v et une date de fin f_v ;
2. deux sommets u et v sont reliés par une arête si et seulement si les intervalles correspondant à u et v s'intersectent ($I_u \cap I_v \neq \emptyset$).

Par la suite, nous supposons que toutes les dates de début et de fin sont différentes.

Voici un exemple de graphes d'intervalle dont la numérotation des sommets suit l'ordre croissant de la date de fin de l'intervalle.





Problème de clique. On dit qu'un sous-ensemble de sommets $K \subset V$ d'un graphe $G = (V, E)$ est une **clique** s'il existe une arête entre toute paire de sommets de K dans G. La **taille** de K est le nombre de sommets de K. Une clique K est **maximum** s'il n'existe pas de clique K' de taille strictement supérieure à celle de K.

Question 1. Proposer un algorithme calculant une clique maximum en $O(n \log n)$ opérations, dans un graphe d'intervalles représenté par les intervalles.

Une clique correspond à un ensemble d'intervalles qui partagent un même point de la droite. Donc pour trouver une clique maximum, il faut trouver un point de la droite qui est contenu dans le plus possible d'intervalles.

Afin d'avoir une faible complexité, nous allons coder le graphe dans un tableau T_G de $2n$ éléments tel que chaque sommet v sera représenté par 2 éléments (v, d) et (v, f). Le premier (resp. le deuxième) représentera le début (resp. la fin) de l'intervalle. De plus, le tableau T_G sera trié en fonction de l'ordre $\prec$ suivant : $(v, a) < (u, b)$ si $a_v < b_u$. L'intervalle $[T_G[i], T_G[i+1]]$ représente un segment de la droite qui est déterminé par des extrémités d'intervalles de sommets.

Entrée : un graphe d'intervalles G
Sortie : un sous-ensemble de sommets S_{max} ;

1. Construire le tableau T_G // $O(n \log n)$ *opérations*
2. $S \leftarrow \emptyset$; *//clique courante*
3. $nb \leftarrow 0$; *//cardinalité de la clique courante*
4. $S_{max} \leftarrow \emptyset$; *//clique maximum courante*
5. $nb_{max} \leftarrow 0$; *//taille de la clique maximum courante*

6. pour i allant de 1 à $2n$ faire :
   (a) $(v, x) \leftarrow T_G[i]$;
   (b) si $x = d$, alors *//Nouveau sommet*
      i. $nb \leftarrow nb + 1$; $S \leftarrow S \cup \{v\}$;
      ii. si ($nb > nb_{max}$ alors ($nb_{max} \leftarrow n$ et $S_{max} \leftarrow S$) ;
   (c) sinon *//fin de l'interval d'un sommet*
      i. $nb \leftarrow nb - 1$; $S \leftarrow S \backslash \{v\}$;
7. retourner S_{max} ;

Complexité : $O(n \log n)$ opérations.

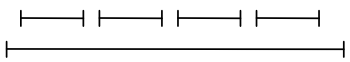
Un **ensemble indépendant** S de G est un ensemble de sommets deux à deux non voisins dans G. La taille d'un ensemble indépendant est égale au nombre de sommets qu'il contient. Nous cherchons un ensemble indépendant de G de taille maximum. Par la suite, nous notons S^* une solution optimale (un ensemble indépendant de taille maximum). Considérons l'algorithme glouton suivant, dépendant d'un ordre $\mathcal{T}$ des sommets :

Algorithme 1 : Glouton(G)

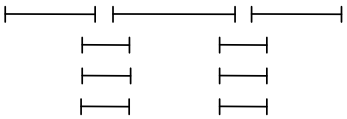
Données : Un graphe d'intervalles $G = (V, E)$
Résultat : Un sous-ensemble de sommets
$S \leftarrow \emptyset$; $Q \leftarrow V$;
tant que ($Q \neq \emptyset$) **faire**
  Choisir un sommet v de Q minimal dans l'ordre $\mathcal{T}$;
  si $\{v\} \cup S$ *reste un ensemble indépendant de* G **alors**
    $S \leftarrow \{v\} \cup S$;
    Supprimer les voisins de v dans Q ;
  fin
fin
Retourner S

Question 2. Donner un exemple où l'algorithme ne retourne pas la solution optimale

1. si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant des dates de début d_v ;



2. si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant sur la longueur des intervalles $f_v - d_v$.



Question 3. Soit S la solution retournée par l'algorithme si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant sur la longueur des intervalles $f_v - d_v$.

1. Montrer que chaque sommet j de S est voisin d'au plus deux sommets de S^*.

   Supposons par contradiction qu'il existe un sommet j de S voisin de trois sommets i_1, i_2, i_3 de S^*. Par définition de S^*, ces trois sommets ne sont pas voisins entre eux car S^* est un ensemble indépendant du graphe d'intervalle. Donc ils sont ordonnés séquentiellement sur la ligne. Sans perte de généralité, nous supposerons qu'ils sont ordonnés avec i_1 en premier, puis i_2, et enfin i_3. Plus formellement, $d_1 < f_1 < d_2 < f_2 < d_3 < f_3$.

   Comme le sommet j est voisin de i_1 et i_3, cela signifie que $d_j < f_1$ et $f_j < f_3$. Donc $f_2 - d_2 < f_j - d_j$.

   Comme le sommet i_2 n'a jamais été sélectionné par l'algorithme, il doit avoir été encore non-sélectionné lorsque l'algorithme a sélectionné le sommet j. Ceci est une contradiction, puisque l'algorithme aurait dû traiter i_2 avant j car $f_2 - d_2 < f_j - d_j$.

2. Montrer que $|S^*| \leq 2|S|$.

   L'intuition est que lorsque l'algorithme glouton sélectionne le sommet j, il peut éliminer deux sommets de la solution optimale. Plus formellement, posons $f\colon S^* \to 2^S$ la fonction associant à un sommet $v \in S^*$, $f(v) = \{v\}$ si $v \in S$ et $f(v) = \{\text{ensemble des voisins de } v \text{ dans } S\}$ sinon. Dans le cas où $v \notin S$, on a nécessairement $f(v) \neq \emptyset$ puisque sinon v aurait été ajouté à S lors de son étude dans l'algorithme glouton, puisqu'il ne briserait pas l'indépendance. Ainsi, $1 \leq |f(v)|$, d'où

$$
\begin{aligned}
|S^*| &= \sum_{v \in S^*} 1 \\
&\leq \sum_{v \in S^*} |f(v)| \\
&\leq \sum_{i \in S^*} \sum_{v \in S} \mathbb{1}_{v \text{ voisin de } i} \\
&\leq \sum_{v \in S} \sum_{i \in S^*} \mathbb{1}_{v \text{ voisin de } i} \\
&\leq \sum_{v \in S} 2 \qquad \text{(question précédente)} \\
|S^*| &\leq 2|S|
\end{aligned}
$$

Notons j le sommet de G telle que sa date de fin f_j soit la plus petite possible.

Question 4. Montrer qu'il existe une solution optimale qui contient le sommet j.

Si S^* contient le sommet j, alors il n'y a rien à prouver.

Maintenant, supposons que S^* ne contient pas le sommet j. Notons i le sommet de S^* de valeur f_i minimale.

Considérons l'ensemble $S' = S^* - \{i\} \cup \{j\}$. Tout d'abord, nous pouvons remarquer que $f_j < f_i$ par hypothèse.

Cela signifie que les sommets de S' à l'exception de j ont une date de début supérieure à f_i et donc f_j. Donc les sommets de $S' \backslash \{j\}$ ne sont pas voisins de j. Donc S' est un ensemble indépendant. Comme $|S'| = |S^*|$, S' est aussi un ensemble indépendant de taille maximum de G.

Question 5. Montrer que l'algorithme retourne une solution optimale si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant de la date de fin.

Soit S la solution retournée par l'algorithme glouton. Soit S^* une solution optimale.

Pour prouver que S est une solution optimale, on montre par récurrence l'assertion suivante : il existe une solution optimale qui contient les k premiers sommets insérés par l'algorithme glouton. Pour $k = 1$, c'est la question précédente. Supposons que l'hypothèse de récurrence est vraie pour k : c'est-à-dire les ensembles S^* et S ont les k premiers éléments identiques.

Soit i le $(k+1)$-ème élément de S^*. Si i est dans S, l'hypothèse de récurrence est vrai pour $k+1$. Supposons maintenant que i n'est pas dans S.

Soit j le $(k+1)$-ème élément de S. S'il est dans S^*, alors cela impliquerait que l'algorithme glouton choisisse i avant j. Ce qui est en contradiction avec le fait que j soit le $(k+1)$-ème élément de S. Donc j n'est pas dans S^*.

Nous allons prouver que i peut être remplacé par j.

En effet il suffit de remarquer que $f_j \leq f_i$: j se termine avant i. Si ce n'était pas le cas, l'algorithme glouton aura choisit le sommet j.

On obtient une autre solution $S' = S^* - \{i\} \cup \{j\}$ telle que $|S'| = |S^*|$ et donc toujours optimale.

Question 6. Simplifier l'algorithme glouton si l'ordre $\mathcal{T}$ sur les sommets est l'ordre croissant de date de fin en supposant donné le graphe par les tableaux de dates de début et de fin. Évaluer sa complexité.

Algorithme 2 : Glouton (G)

Données : Un graphe d'intervalles $G = (V, E)$
Résultat : Un sous-ensemble de sommets
Numéroter les sommets de telle façon que $f_i < f_j$;
Initialiser des variables : $S \leftarrow \emptyset$; $date\_fin \leftarrow \min\{d_v | v \in V\}$;
pour v *allant de* 1 *à* n **faire**
  si $d_v > date\_fin$ **alors**
    $S \leftarrow \{v\} \cup S$;
    $date\_fin \leftarrow f_v$;
  fin
fin
Retourner S;

Complexité : $\mathcal{O}(n \log n)$ opérations.

Nous associons une fonction de poids positif sur chaque sommet $w : V \to \mathbb{N}$. Nous cherchons un sous-ensemble indépendant S qui maximise la somme $\sum_{u \in S} w(u)$ de tous les poids de ces sommets. Nous notons S^* un ensemble indépendant de G qui maximise cette quantité.

Convention : À partir de maintenant, nous supposons que les sommets sont numérotés de 1 à n en fonction de leur date de fin f_v : $i < j$ signifie que $f_i < f_j$.

Notons $p(j)$ le plus grand sommet dont l'intervalle finit avant d_j. S'il n'en existe pas, nous supposons que $p(j) = 0$. Dans l'exemple, $p(1) = p(2) = 0$, $p(3) = p(5) = 1$ et $p(4) = 2$.

Question 7. Donner un algorithme en $O(n \log n)$ opérations qui calcule la fonction p pour tous les sommets.

Afin d'avoir une faible complexité, nous allons construire le codage de la question 1.

Algorithme 3 : Calcul de la fonction p dans G.

Données : Un graphe d'intervalles $G = (V, E)$ numéroté par leur date de fin.
Résultat : un tableau p de n éléments
Construire le tableau T_G // $O(n \log n)$ *opérations* ;
$dernier\_intervalle \leftarrow 0$;
pour ℓ *allant de* 1 *à* $2n$ **faire**
  $(v, x) \leftarrow T_G[\ell]$;
  si $x == f$ **alors**
    $dernier\_intervalle \leftarrow v$;
  sinon
    $p(v) \leftarrow dernier\_intervalle$;
  fin
fin
Retourner le tableau p;

Complexité : $\mathcal{O}(n \log n)$ opérations.

Question 8. Montrer que si $j \in S^*$, alors $S^* \cap \{1, \ldots, p(j)\}$ est un ensemble indépendant de G qui maximise la somme de tous les poids de ces sommets dans le sous-graphe de G ayant les sommets 1 à $p(j)$.

Soit $G_{p(j)}$ le sous graphe de G qui contient uniquement les sommets dont le numéro est plus petit que $p(j)$. Nous allons prouver par contradiction que $S^* \cap \{1, 2, \ldots, p(j)\}$ est une solution optimale dans $G_{p(j)}$. Supposons que ce n'est pas le cas.

Soit S' une solution optimale dans $G_{p(j)}$. Soit $S'' = (S^* \backslash \{1, 2, \ldots, p(j)\}) \cup S'$.

Nous allons prouver que (1) S'' est un ensemble indépendant de G, puis (2), le poids de S'' est strictement plus grand que S^*. Ce qui nous amènera à une contradiction avec le fait que S^* est un ensemble indépendant de G qui maximise la somme des poids de tous ces sommets.

Prouvons le point (1). Il suffit de prouver qu'aucun sommet de $S^* \backslash \{1, 2, \ldots, p(j)\}$ est voisin d'un sommet de S'. Soit v un sommet de S'. Soit ℓ un sommet de $S^* \backslash \{1, 2, \ldots, p(j)\}$. Par définition, nous avons $f_v < f_{p(j)}$. Comme $v < \ell$, nous avons $f_v < f_\ell$. Si $j = \ell$, alors v n'est pas voisin de j (car $f_v < f_{p(j)} < d_j$). Supposons que $j < \ell$. Comme $j \in S^*$, et que S^* est indépendant, ℓ n'est pas voisin de j. Donc, comme nous avons $d_v < f_j < f_\ell$, cela implique que $d_v < f_j < d_\ell < f_\ell$. Donc v et ℓ ne sont pas voisin.

nous avons $\forall \ell,\ p(j) < \ell \leq n$.

tous les sommets de numéro strictement plus grand que $p(j)$ ne sont pas voisins avec v car

Tous les sommets de S' ont aussi un numéro inférieur à $p(j)$. Comme les sommets de $S^* \cap \{1, 2, \ldots, p(j)\}$ ont des numéros supérieurs à $p(j)$, nous pouvons déduire que S'' est un ensemble indépendant de G,

Focalisons nous sur le point (2). $S^* \cap \{1, 2, \ldots, p(j)\}$ est un ensemble indépendant dans $G_{p(j)}$. Comme S' une solution optimale dans $G_{p(j)}$, nous avons

$$\sum_{u \in S'} w(u) > \sum_{u \in S^* \cap \{1,2,\ldots,p(j)\}} w(u) \qquad (1)$$

Maintenant, focalisons nous sur la somme $\sum_{u \in S''} w(u)$

$$\begin{aligned}
\sum_{u \in S''} w(u) &= \sum_{u \in S^* \backslash \{1,2,\ldots,p(j)\}} w(u) + \sum_{u \in S'} w(u) \\
&= \sum_{u \in S^*} w(u) - \sum_{u \in S^* \cap \{1,2,\ldots,p(j)\}} w(u) + \sum_{u \in S'} w(u) \\
\sum_{u \in S''} w(u) &> \sum_{u \in S^*} w(u) \qquad \text{(car l'inégalité 1 est appliquée.)}
\end{aligned}$$

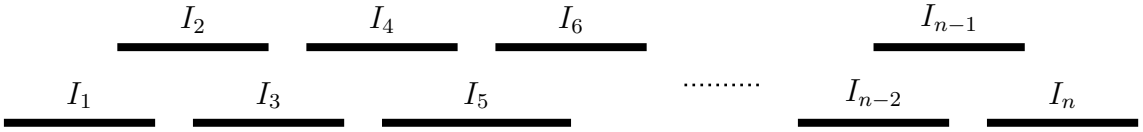
Question 9. Notons $OPT(j)$ le poids d'une solution optimale dans le sous-graphe ayant les sommets 1 à j. Donner une formule de récurrence pour $OPT(j)$

Soit S^* une solution optimale.
Montrons que si $j \notin S^*$, alors $S^* \cap \{1, \ldots, j-1\}$ est un ensemble indépendant de G qui maximise la somme de tous les poids de ces sommets dans le sous-graphe G_{j-1} ayant les sommets 1 à $j-1$. Pour cela, on peut procéder par contradiction. Soit S' une solution optimale dans G_{j-1}. Soit $S'' = (S^* \backslash \{1, 2, \ldots, j-1\}) \cup S'$. Nous allons prouver que (1) S'' est un ensemble indépendant de G, puis (2), le poids de S'' est strictement plus grand que S^*. Ce qui nous amènera à une contradiction avec le fait que S^* est un ensemble indépendant de G qui maximise la somme des poids de tous ces sommets. On utilise les mêmes techniques que la question précédente.
Nous définissons $OPT(0) = 0$, en nous basant sur la convention que la solution optimale d'un graphe sans sommet est l'ensemble vide. En utilisant la question précédente,

$$OPT(j) = \begin{cases} w(j) + OPT(p(j)) & \text{si} \quad j \text{ est dans une solution optimale} \\ OPT(j-1) & \text{si} \quad j \text{ n'est pas dans une solution optimale} \end{cases}$$

Donc $OPT(j) = \max(w(j) + OPT(p(j)), OPT(j-1))$.

Question 10. Écrire un algorithme récursif qui calcule $OPT(n)$. Appliquer l'algorithme récursif sur l'instance de la figure suivante. Évaluer la complexité de cet algorithme dans le cas général.



Algorithme 4 : calculer-OPT(G, j)

Données : Un graphe d'intervalles $G = (V, E)$, un sommet j
Calculer la valeur p pour tous les sommets de j;
si $j = 0$ **alors**
  retourner 0 ;
sinon
  retourner max $\big(w(j)+$ calculer-OPT$(G, p(j))$, calculer-OPT$(G, j-1)\big)$;
fin

Nous allons compter le nombre de fois qu'on fait un appel récursif. Soit $F(j)$ le nombre un appel récursif de la fonction calculer-OPT ayant $(G,\ j)$ comme entrée

$$\begin{aligned} F(j) &= F(j-1) + F(j-2) \\ F(0) &= 1 \\ F(1) &= 0 \end{aligned}$$

Il s'agit de la suite de Fibonacci, qui grossit exponentiellement en fonction de j.

Question 11. Comment réduire la complexité de l'algorithme précédent ? Donner sa complexité après amélioration.

Algorithme 5 : calculer-OPT(G, j)

Données : Un graphe d'intervalles $G = (V, E)$
Calculer la valeur p pour tous les sommets de j;
Créer un tableau $OPT[0, \ldots, n]$ dont tous les éléments sont initialisés à 0 ;
pour $j \leftarrow 1$ **à** n **faire**
  $OPT[j] \leftarrow \max(w(j) + OPT[p(j)], OPT[j-1])$;
fin

Complexité : $O(n)$ opérations.

Dérivée d'expressions rationnelles - Correction

Soit A un alphabet fini et non vide. On rappelle qu'une **expression rationnelle** sur A est toute formule sur l'alphabet $A \uplus \{0, 1, +, \cdot, *, (,)\}$ qui peut être obtenue de la manière suivante :

— 0, 1 et a, pour tout a dans A, sont des expressions rationnelles ;
— si E et F sont deux expressions rationnelles, alors $(E + F)$, $(E \cdot F)$ et (E^*) sont des expressions rationnelles.

Comme habituellement, on s'autorise à supprimer des parenthèses, grâce à la convention de priorité : $*$ a priorité sur $\cdot$ qui a priorité sur $+$. Le langage $\mathcal{L}(E)$ dénoté par l'expression rationnelle E est défini de manière inductive :

— $\mathcal{L}(0) = \emptyset$, $\mathcal{L}(1) = \{\varepsilon\}$, $\mathcal{L}(a) = \{a\}$ pour tout a dans A ;
— $\mathcal{L}(E + F) = \mathcal{L}(E) \cup \mathcal{L}(F)$,
— $\mathcal{L}(E \cdot F) = \mathcal{L}(E) \cdot \mathcal{L}(F)$, avec $X \cdot Y = \{x \cdot y \mid x \in X, y \in Y\}$, pour tous langages X, Y,
— $\mathcal{L}(E^*) = \mathcal{L}(E)^*$ avec $X^* = \{x_1 \cdots x_n \mid n \geq 0, x_1, \ldots, x_n \in X\}$, pour tout langage X.

Un langage est dit rationnel s'il s'écrit $\mathcal{L}(E)$ pour une expression rationnelle E. Le **terme constant** $c(L)$ d'un langage L sur l'alphabet A vaut $c(L) = 1$ si $\varepsilon \in L$ et $c(L) = 0$ sinon.

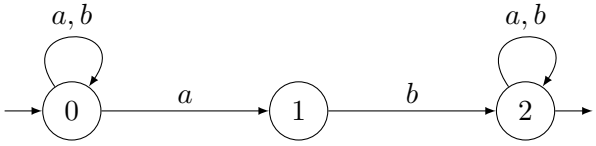
1. Donner une définition inductive du terme constant $c(E)$ d'une expression rationnelle E qui satisfait $c(E) = c(\mathcal{L}(E))$.

Solution : On définit $c(E)$ par induction sur l'expression E :
— $c(1) = 1$, $c(0) = c(a) = 0$ pour tout $a \in A$;
— $c(E + F) = \max(c(E), c(F))$, $c(E \cdot F) = \min(c(E), c(F))$ et $c(E^*) = 1$.

Dans ce sujet, on va s'intéresser à des algorithmes pour transformer une expression rationnelle en un automate fini reconnaissant le langage dénoté par l'expression. On rappelle brièvement la définition des automates finis que nous allons considérer. Un **automate fini** $\mathcal{A}$ sur l'alphabet A est la donnée d'un tuple $\mathcal{A} = \langle Q, \delta, I, T \rangle$ où Q est un ensemble fini d'états, $\delta \subseteq Q \times A \times Q$ est l'ensemble des transitions, $I \subseteq Q$ est l'ensemble des états initiaux et $T \subseteq Q$ est l'ensemble des états terminaux. La transition $(p, a, q) \in \delta$ est notée par la flèche $p \xrightarrow{a} q$. Un **calcul** de $\mathcal{A}$ est une séquence finie de transitions $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$: on note un tel calcul $p_0 \xrightarrow{a_1 a_2 \cdots a_n} p_n$. Son étiquette est le mot $a_1 a_2 \cdots a_n$ de A^*. Le calcul est acceptant si $p_0 \in I$ et $p_n \in T$. Le **langage** de $\mathcal{A}$ est le sous-ensemble $\mathcal{L}(\mathcal{A})$ de A^* qui contient l'ensemble des étiquettes des calculs acceptants de $\mathcal{A}$. L'automate $\mathcal{A}$ est dit **déterministe** si I possède un unique élément et pour tout $p \in Q$ et $a \in A$, il existe au plus un état $q \in Q$ tel que $(p, a, q) \in T$.

2. Soit $E_1 = (a+b)^* \cdot \Big(a \cdot \big(b \cdot (a+b)^*\big)\Big)$. Donner un automate fini reconnaissant le langage $\mathcal{L}(E_1)$.

Solution :



On va étudier un algorithme de dérivation pour la construction d'un automate fini reconnaissant le langage dénoté par une expression rationnelle. On définit la dérivée $\frac{\partial}{\partial a}E$ d'une expression rationnelle E par une lettre $a \in A$ par induction :

— $\frac{\partial}{\partial a}0 = \frac{\partial}{\partial a}1 = \frac{\partial}{\partial a}b = 0$ pour tout $b \neq a$ dans A et $\frac{\partial}{\partial a}a = 1$;

— $\frac{\partial}{\partial a}(E+F) = \frac{\partial}{\partial a}E + \frac{\partial}{\partial a}F$, $\frac{\partial}{\partial a}(E \cdot F) = \left[\frac{\partial}{\partial a}E\right] \cdot F + c(E) \cdot \frac{\partial}{\partial a}F$, $\frac{\partial}{\partial a}E^* = \left[\frac{\partial}{\partial a}E\right] \cdot E^*$.

On s'autorise les identités triviales suivantes (c'est-à-dire qu'on s'autorise toujours à remplacer une expression par son expression réduite équivalente) pour simplifier les expressions rationnelles :

$$E + 0 \equiv 0 + E \equiv E, \qquad E \cdot 0 \equiv 0 \cdot E \equiv 0, \qquad E \cdot 1 \equiv 1 \cdot E \equiv E \qquad (\mathbf{Tr})$$

3. Calculer $\frac{\partial}{\partial a}E_1$ et $\frac{\partial}{\partial b}E_1$.

Solution :

$$\begin{aligned}
\frac{\partial}{\partial a}E_1 &= \left[\frac{\partial}{\partial a}(a+b)^*\right] \cdot (a \cdot (b \cdot (a+b)^*)) + c((a+b)^*) \cdot \left[\frac{\partial}{\partial a}(a \cdot (b \cdot (a+b)^*))\right] \\
&= \left[\frac{\partial}{\partial a}(a+b)\right] \cdot (a+b)^* \cdot (a \cdot (b \cdot (a+b)^*)) \\
&\qquad + \left[\frac{\partial}{\partial a}a\right] \cdot (b \cdot (a+b)^*) + c(a) \cdot \left[\frac{\partial}{\partial a}(b \cdot (a+b)^*)\right] \\
&= E_1 + (b \cdot (a+b)^*)
\end{aligned}$$

De même, on trouve $\frac{\partial}{\partial b}E_1 = E_1$.

Étant donnés un mot $u \in A^*$ et un langage $L \subseteq A^*$, on appelle **quotient** de L par u, et on note $u^{-1}L$, le langage $u^{-1}L = \{v \in A^* \mid uv \in L\}$.

4. Donner une définition par récurrence de la dérivée d'une expression rationnelle E par un mot u, notée $\frac{\partial}{\partial u}E$, qui satisfait $\mathcal{L}(\frac{\partial}{\partial u}E) = u^{-1}\mathcal{L}(E)$. On démontrera rapidement cette égalité. *[Indication : il s'agit donc de définir $\frac{\partial}{\partial \varepsilon}E$, ainsi que $\frac{\partial}{\partial u}E$ en fonction de $\frac{\partial}{\partial u'}E$ pour u' un mot de longueur inférieure à celle de u.]*

Solution : On commence par observer que pour toute lettre a, et langages L et K, $a^{-1}(L \cup K) = a^{-1}L \cup a^{-1}K$, $a^{-1}(L \cdot K) = (a^{-1}L) \cdot K \cup c(L) \cdot a^{-1}K$ et $a^{-1}(L^*) = (a^{-1}L) \cdot L^*$. Ainsi, on obtient par une induction sur les expressions que $\mathcal{L}(\frac{\partial}{\partial a}E) = a^{-1}\mathcal{L}(E)$.
Ensuite, on définit $\frac{\partial}{\partial u}E$ par induction sur u. On pose $\frac{\partial}{\partial \varepsilon}E = E$, puis $\frac{\partial}{\partial ua}E = \frac{\partial}{\partial a}(\frac{\partial}{\partial u}E)$. On vérifie alors par induction sur u que $\mathcal{L}(\frac{\partial}{\partial u}E) = u^{-1}\mathcal{L}(E)$, puisque $(ua)^{-1}L = a^{-1}(u^{-1}L)$.

5. Sans propriétés supplémentaires (par exemple l'associativité de $+$ et $\cdot$), démontrer que E_1 possède un nombre infini de dérivées différentes par rapport à des mots de A^*.

Solution : On vérifie par exemple que

$$\frac{\partial}{\partial(ab)^n}E_1 = E_1 + \underbrace{(a+b)^* + \cdots + (a+b)^*}_{n \text{ fois}}$$

On ajoute les identités suivantes :

$$(E+F)+G \equiv E+(F+G) \qquad (\mathbf{A_s})$$
$$E+F \equiv F+E \qquad (\mathbf{C})$$
$$E+E \equiv E \qquad (\mathbf{I})$$

6. Montrer que l'ensemble des dérivées d'une expression rationnelle (par rapport à des mots de A^*) est fini si l'on s'autorise d'utiliser les identités $(\mathbf{A_s})$, $(\mathbf{C})$ et $(\mathbf{I})$.

Solution : On montre par induction que le nombre de dérivées $\mathbf{nd}(E)$ d'une expression E modulo $(\mathbf{A_s})$, $(\mathbf{C})$ et $(\mathbf{I})$ est fini.
Si E est une expression atomique, $\mathbf{nd}(E)$ vaut 1 ou 2.
Pour la somme, on obtient facilement que $\frac{\partial}{\partial f}(E+F) = \frac{\partial}{\partial f}E + \frac{\partial}{\partial f}F$, et donc que $\mathbf{nd}(E+F) \leq \mathbf{nd}(E)\mathbf{nd}(F)$.
Pour la concaténation, la formule est plus compliquée :

$$\frac{\partial}{\partial a_1a_2\cdots a_n}(E\cdot F) = \left[\frac{\partial}{\partial a_1a_2\cdots a_n}E\right]\cdot F + c\left(\frac{\partial}{\partial a_1a_2\cdots a_{n-1}}E\right)\cdot\frac{\partial}{\partial a_n}F + c\left(\frac{\partial}{\partial a_1a_2\cdots a_{n-2}}E\right)\cdot\frac{\partial}{\partial a_{n-1}a_n}F + \cdots + c(E)\frac{\partial}{\partial a_1a_2\cdots a_n}F$$

Ainsi $\frac{\partial}{\partial f}(E\cdot F)$ est la somme de $(\frac{\partial}{\partial f}E)\cdot F$ et, au plus, de $|f|$ dérivées de F. Cependant, grâce aux identités $(\mathbf{A_s})$ et $(\mathbf{C})$, on peut regrouper les expressions identiques, qui se fondent en une seule modulo $(\mathbf{I})$. On obtient donc une somme de dérivées de F toutes distinctes : il y a $2^{\mathbf{nd}(F)}$ telles sommes. Ainsi, $\mathbf{nd}(E\cdot F) \leq \mathbf{nd}(E)2^{\mathbf{nd}(F)}$.
Pour l'étoile de Kleene, on voit que

$$\frac{\partial}{\partial ab}(E^*) = \left[\frac{\partial}{\partial ab}E\right]\cdot E^* + c\left(\frac{\partial}{\partial a}E\right)\cdot\frac{\partial}{\partial b}E\cdot E^*$$

et

$$\frac{\partial}{\partial abc}(E^*) = \left[\frac{\partial}{\partial abc}E\right]\cdot E^* + c\left(\frac{\partial}{\partial ab}E\right)\cdot\frac{\partial}{\partial c}E\cdot E^* + c\left(\frac{\partial}{\partial a}E\right)\cdot\frac{\partial}{\partial bc}E\cdot E^* + c\left(\frac{\partial}{\partial a}E\right)\cdot c\left(\frac{\partial}{\partial b}E\right)\cdot\frac{\partial}{\partial c}E\cdot E^* \quad (1)$$

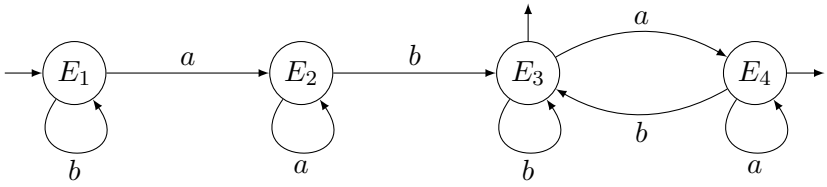
Plus généralement, $\frac{\partial}{\partial f}(E^*)$ est une somme d'au plus $|f|$ termes de la forme $\frac{\partial}{\partial g}E\cdot E^*$, avec g un suffixe de f. Grâce au même raisonnement que précédemment, il existe au plus $2^{\mathbf{nd}(E)}$ telles sommes, donc $\mathbf{nd}(E^*) \leq 2^{\mathbf{nd}(E)}$.

7. En déduire que l'ensemble D_E des dérivées d'une expression rationnelle E, où l'on s'autorise d'utiliser $(\mathbf{A_s})$, $(\mathbf{C})$ et $(\mathbf{I})$, est effectivement calculable et qu'on peut ainsi construire un automate fini déterministe $\mathcal{A}_E$ reconnaissant le langage $\mathcal{L}(E)$. Construire cet automate pour l'expression E_1 de la question **2**.

Solution : On remarque tout d'abord que les calculs modulo ($\mathbf{A_s}$), (**C**) et (**I**) sont effectifs, puisqu'on peut ordonner lexicographiquement les expressions pour faire en sorte que chaque expression se réduise en une expression unique de longueur minimale. Si on note ensuite $D_E^{(k)}$ l'ensemble des dérivées de E par rapport à tous les mots de A^* de longueur inférieure ou égale à k, on obtient une suite croissante d'expressions. Si pour un certain ℓ, $D_E^{(\ell)} = D_E^{(\ell+1)}$ (ce qui est sûr d'arriver puisque $\mathbf{nd}(E)$ est fini), alors

$$D_E^{(\ell+2)} = \left\{ \frac{\partial}{\partial a} F \mid F \in D_E^{(\ell+1)}, a \in A \right\} = \left\{ \frac{\partial}{\partial a} F \mid F \in D_E^{(\ell)}, a \in A \right\} = D_E^{(\ell+1)} = D_E^{(\ell)}$$

On obtient ainsi que $D_E^{(\ell+p)} = D_E^{(\ell)}$ pour tout p, et donc que $D_E = D_E^{(\ell)}$.
L'automate des expressions dérivées est alors $\mathcal{A}_E = \langle D_E, \delta_E, I_E, T_E \rangle$ avec $\delta_E = \{(F, a, \frac{\partial}{\partial a} F) \mid F \in D_E, a \in A\}$, $I_E = \{E\}$ et $T_E = \{F \in D_E \mid c(F) = 1\}$. Pour l'expression E_1, on note $E_2 = \frac{\partial}{\partial a} E_1 = E_1 + b \cdot (a+b)^*$, $E_3 = \frac{\partial}{\partial ab} E_1 = E_1 + (a+b)^*$ et $E_4 = \frac{\partial}{\partial aba} E_1 = E_1 + b \cdot (a+b)^* + (a+b)^*$. Ainsi $\mathcal{A}_E$ est l'automate



On peut observer que l'automate $\mathcal{A}_E$ est certes déterministe, mais pas nécessairement minimal, comme le montre l'exemple de l'expression E_1. Pour éviter une explosion exponentielle, on s'intéresse maintenant à obtenir un automate $\mathcal{B}_E$ non déterministe à partir d'une définition alternative de dérivées d'expressions. Ainsi, on définit, comme un ensemble d'expression, la **B**-dérivée $\frac{\partial'}{\partial' a} E$ d'une expression rationnelle E par une lettre $a \in A$ par induction :

- $\frac{\partial'}{\partial' a} 0 = \frac{\partial'}{\partial' a} 1 = \frac{\partial'}{\partial' a} b = \emptyset$ pour tout $b \neq a$ dans A et $\frac{\partial'}{\partial' a} a = \{1\}$;
- $\frac{\partial'}{\partial' a}(E + F) = \frac{\partial'}{\partial' a} E \cup \frac{\partial}{\partial a} F$;
- $\frac{\partial'}{\partial' a}(E \cdot F) = \left[\frac{\partial}{\partial a} E\right] \cdot \{F\} \cup \{c(E)\} \cdot \frac{\partial}{\partial a} F$;
- $\frac{\partial'}{\partial' a} E^* = \left[\frac{\partial}{\partial a} E\right] \cdot \{E^*\}$.

Tous les calculs sont menés modulo les identités triviales (**Tr**).

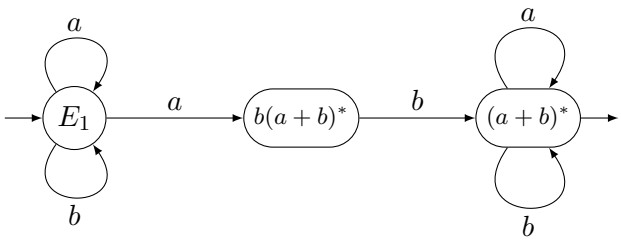
8. Étendre, comme dans la question **4**, la notion de **B**-dérivée pour définir la **B**-dérivée $\frac{\partial'}{\partial' u} E$ par rapport à un mot $u \in A^*$. On appelle terme dérivé de E tout expression différente de 0 qui apparait dans au moins un ensemble $\frac{\partial'}{\partial' u} E$ pour $u \in A^*$. Donner, en la prouvant, une borne sur le nombre de termes dérivés d'une expression E.

Solution : Comme avant, on pose $\frac{\partial'}{\partial' \epsilon} E = \{E\}$ et $\frac{\partial'}{\partial' ua} E = \frac{\partial'}{\partial' a}\left(\frac{\partial'}{\partial' u} E\right)$, en ayant étendu la **B**-dérivation à un ensemble d'expression via : $\frac{\partial'}{\partial' u}(\bigcup_{i \in I} E_i) = \bigcup_{i \in I} \frac{\partial'}{\partial' u} E_i$. On note $\ell(E)$ le nombre d'occurrences de lettres de A dans E et on montre par induction que E possède au plus $\ell(E)$ termes dérivés.
C'est vrai pour les expressions atomiques. Pour l'induction, on prend E_1, E_2 :

- Si $E = E_1 + E_2$, les termes dérivés de E sont ceux de E_1 et de E_2 : il y en a donc au plus $\ell(E_1) + \ell(E_2) = \ell(E)$.
- Si $E = E_1 \cdot E_2$, les termes dérivés de E sont ceux de E_1 concaténés avec E_2 et éventuellement ceux de E_2 : il y en donc au plus $\ell(E_1) + \ell(E_2) = \ell(E)$.
- Si $E = E_1^\star$, les termes dérivés de E sont ceux de E_1 concaténés avec $E_1^\star$: il y en a donc au plus $\ell(E_1) = \ell(E)$.

9. Donner alors la définition d'un automate non-déterministe $\mathcal{B}_E$ qui reconnait le langage $\mathcal{L}(E)$, et dont les états sont les termes dérivés de E auxquels on ajoute E (s'il n'est pas déjà présent). Construire $\mathcal{B}_{E_1}$ pour l'expression E_1 de la question **2**.

Solution : Soit P_E l'ensemble des termes dérivés de E auxquels on ajoute E. On définit $\mathcal{B}_E = \langle P_E, \delta'_E, I'_E, T'_E \rangle$ avec $\delta'_E = \{(F, a, G) \mid F \in P_E, a \in A, G \in \frac{\partial}{\partial a} F\}$, $I'_E = \{E\}$ et $T'_E = \{F \in P_E \mid c(F) = 1\}$. Pour l'expression E_1, les termes dérivés sont $P_{E_1} = \{E_1, b(a+b)^*, (a+b)^*\}$ et on obtient l'automate $\mathcal{B}_{E_1}$ suivant :



10. Vérifier que, pour toute expression rationnelle E, l'automate $\mathcal{A}_E$ est l'automate déterminisé de l'automate $\mathcal{B}_E$.

Solution : Il suffit de vérifier par récurrence sur la longueur de u que $\frac{\partial}{\partial u} E$ est la somme (et non l'union) des états atteints par le mot u dans l'automate $\mathcal{B}_E$.

Basé sur le livre *Éléments de théorie des automates* de Jacques Sakarovitch.