Bases de Données II, Mons, 5 juin 2025

NOM + PRÉNOM :	
Orientation + Année :	

Cet examen contient 10 questions. Durée : exactement 3 heures. Les questions sont censées être claires. Aucune clarification supplémentaire ne sera donnée pendant l'examen. Si une question vous semble ambiguë ou incomplète, écrivez vos hypothèses et répondez en fonction de celles-ci. **Il est permis de détacher la dernière page.**

Un fleuriste livre des bouquets à la maison. Chaque espèce de fleur (tulipe, rose...) a un prix exprimé en centimes d'euro. Par exemple, le prix d'une rose est de 150 centimes d'euro, indépendamment de sa couleur ; voir la ligne

```
<fleur fnom="rose" prix="150"/>
```

Un bouquet rassemble des fleurs de différentes espèces et couleurs. Par exemple, le bouquet *Exotique* est composé d'un seul tournesol et trois iris bleus. Voir les lignes :

```
<bouquet bnom="Exotique">
   <fleur fnom="tournesol" couleur="jaune" nombre="1"/>
   <fleur fnom="iris" couleur="bleu" nombre="3"/>
</bouquet>
```

La figure 3 montre le document XML et la figure 4 montre le DTD.

La balise `ventes-par-jour` est utilisée pour encoder les ventes journalières. Par exemple, à la date du 18 septembre 2022, le fleuriste a vendu trois bouquets *Valentin* et six bouquets *Printemps*. Voir les lignes :

```
<date mois="sep 2022" jour="18">
  <vente bnom="Valentin">3</vente>
  <vente bnom="Printemps">6</vente>
</date>
```

Question 1 Écrivez une requête en **XPath** qui renvoie le nom de chaque bouquet, à l'exception de celui ayant le plus grand nombre de couleurs distinctes. **La fonction `count` est la seule fonction d'agrégation autorisée. L'utilisation de `max`, `min` et `distinct-values` est interdite.** Évitez également les axes `parent` et `ancestor`.

Pour le document XML de la figure 3, la réponse est comme suit :

```
bnom="Valentin"
bnom="Exotique"
bnom="Printemps"
```

Notez que le bouquet *Belge*, ayant le plus grand nombre de couleurs distinctes (soit 3), ne fait pas partie de la réponse. Notez également que le bouquet *Printemps* ne contient que deux couleurs, à savoir jaune et rose.

.../5

```
//bouquet[
            count(fleur[not(@couleur=following-sibling::fleur/@couleur)]/@couleur)
             < //bouquet/count(fleur[not(@couleur=following-sibling::fleur/@couleur)]/@couleur)
        ]/@bnom
```

Explication : Pour chaque bouquet, `count` prend en compte une couleur uniquement lorsqu'il s'agit de la dernière fleur de cette couleur présente dans le bouquet. Ainsi, au lieu de compter les couleurs, il suffit de compter les fleurs correspondant à ces dernières occurrences. La requête peut donc être simplifiée comme suit :

```
//bouquet[
            count(fleur[not(@couleur=following-sibling::fleur/@couleur)])
              < //bouquet/count(fleur[not(@couleur=following-sibling::fleur/@couleur)])
        ]/@bnom
```

Question 2 Écrivez une requête en **XPath** qui renvoie le nom du bouquet contenant le plus grand nombre de fleurs. **La fonction `sum` est la seule fonction d'agrégation autorisée. L'utilisation de `max` et `min` est interdite.**

Pour le document XML de la figure 3, la réponse est la suivante :

```
bnom="Printemps"
```

Notez que le bouquet *Printemps* contient 17 fleurs, et aucun autre bouquet ne contient autant de fleurs.

.../5

```
//bouquet[not(sum(fleur/@nombre)<//bouquet/sum(fleur/@nombre))]/@bnom
```

Question 3 Pour le schéma

$$\{ABC, BCD, BFG, CDE, EFG\},$$

cochez la case qui précède une assertion correcte :

☐ le schéma est α-acyclique ;

☐ le schéma n'est pas α-acyclique.

Si le schéma est α-acyclique, donnez un arbre de jointure pour ce schéma. Si le schéma n'est pas α-acyclique, expliquez pourquoi.

.../10

Question 4 Écrivez un programme en XSLT qui, pour chaque bouquet, renvoie le nombre total de fleurs de chaque espèce, sans répétitions. **Il n'est pas permis d'utiliser `xsl:for-each` et `xsl:if`. Il n'est pas non plus permis d'utiliser la fonction `distinct-values` ou l'instruction `<xsl:with-param>`.** L'usage de la fonction `sum` est permis. Les résultats doivent être groupés comme suit :

```
<bouquets>
   <bouquet bname="Valentin">
      <flower fname="rose" quantity="10" />
   </bouquet>
   <bouquet bname="Belge">
      <flower fname="tulipe" quantity="13" />
   </bouquet>
   <bouquet bname="Exotique">
      <flower fname="tournesol" quantity="1" />
      <flower fname="iris" quantity="3" />
   </bouquet>
   <bouquet bname="Printemps">
      <flower fname="rose" quantity="10" />
      <flower fname="tulipe" quantity="4" />
      <flower fname="dahlia" quantity="3" />
   </bouquet>
</bouquets>
```

Notez que la sortie contient des noms en anglais, tels que `flower` et `fname` au lieu de `fleur` et `fnom`. **Il est essentiel que dans le résultat, à l'intérieur d'un même bouquet, une espèce de fleur n'apparaisse qu'une seule fois, avec `quantity` indiquant le nombre total de fleurs de cette espèce.** Par exemple, le bouquet Belge comprend un total de 13 tulipes.

. . . /10

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
   <bouquets>
      <xsl:apply-templates select="//bouquet"/>
   </bouquets>
</xsl:template>

<xsl:template match="bouquet">
   <bouquet bname="{@bnom}">
      <xsl:apply-templates select="fleur[not(@fnom=following-sibling::fleur/@fnom)]"/>
   </bouquet>
</xsl:template>

<xsl:template match="fleur">
   <flower fname="{@fnom}">
      <xsl:attribute name="quantity">
         <xsl:value-of select="sum(parent::bouquet/fleur[@fnom=current()/@fnom]/@nombre)"/>
      </xsl:attribute>
   </flower>
</xsl:template>

</xsl:stylesheet>
```

Question 5 Rédigez une requête en **XQuery** qui retourne, pour chaque bouquet, les différentes espèces de fleurs présentes ainsi que le prix total correspondant à chaque espèce. L'utilisation de la fonction `distinct-values` est autorisée. Pour le document XML de la figure 3, la réponse est :

```
<bouquets>
   <bouquet bname="Valentin">
      <flower fname="rose" price="1500"/>
   </bouquet>
   <bouquet bname="Belge">
      <flower fname="tulipe" price="1300"/>
   </bouquet>
   <bouquet bname="Exotique">
      <flower fname="tournesol" price="300"/>
      <flower fname="iris" price="750"/>
   </bouquet>
   <bouquet bname="Printemps">
      <flower fname="rose" price="1500"/>
      <flower fname="tulipe" price="400"/>
      <flower fname="dahlia" price="360"/>
   </bouquet>
</bouquets>
```

Notez que la sortie utilise des noms en anglais, comme `flower` et `fname`, au lieu de `fleur` et `fnom`. Il est essentiel que chaque espèce de fleur n'apparaisse qu'une seule fois dans un bouquet, avec `price` indiquant le prix total des fleurs de cette espèce. Par exemple, le bouquet Belge contient 13 tulipes à 100 centimes chacune, ce qui donne un prix total de 1300 centimes pour les tulipes.

Formulez votre requête de manière à rendre sa logique aisément compréhensible.

.../10

```
<bouquets>{
for $b in /fleuriste/bouquets/bouquet
return <bouquet bname="{$b/@bnom}">{
           for $f in //fleurs/fleur[@fnom=$b/fleur/@fnom]
           let $q:=sum($b/fleur[@fnom=$f/@fnom]/@nombre)
           return <flower fname="{$f/@fnom}" price="{$f/@prix*$q}"/>
       }</bouquet>
}</bouquets>
```

Question 6 Simplifiez la requête (UCQ) suivante ou expliquez pourquoi aucune simplification n'est possible :

$$\begin{cases} Answer(x) & \leftarrow & R(x,y_1,z_1), R(x,y_2,z_1), R(y_1,y_2,z_2), R(x,y_3,z_3), R(x,y_4,z_3), R(y_4,y_3,z_3) \\ Answer(y) & \leftarrow & R(y,x_1,z_1), R(y,x_2,z_2), R(x_1,x_2,z_3), R(y,x_3,z_4), R(y,x_4,z_5), R(x_4,x_3,z_6) \end{cases}$$

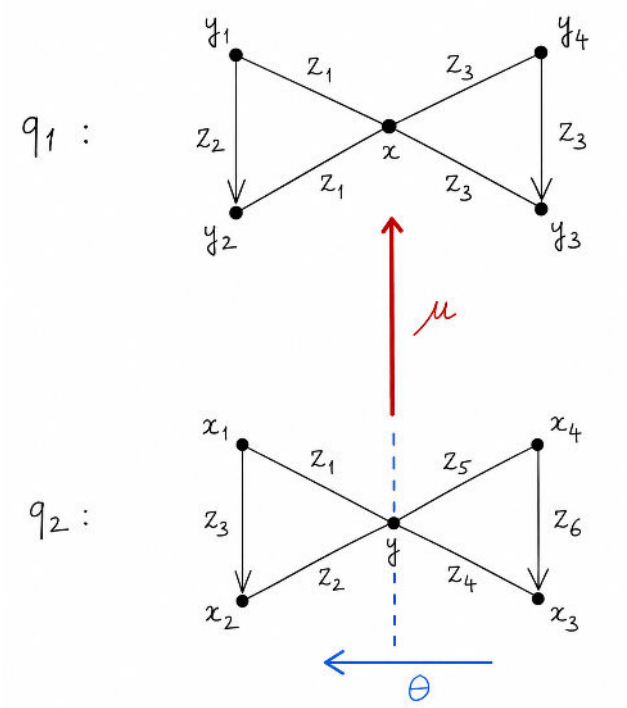
Détaillez les calculs.

.../10

Soit q_1 la première requête, et q_2 la deuxième.

Soit μ la substitution définie par :

$\mu(y) = x,$
$\mu(x_1) = y_1,$
$\mu(z_1) = z_1,$
$\mu(x_2) = y_2,$
$\mu(z_2) = z_1,$
$\mu(z_3) = z_2,$
$\mu(x_3) = y_3,$
$\mu(z_4) = z_3,$
$\mu(x_4) = y_4,$
$\mu(z_5) = z_3,$
$\mu(z_6) = z_3.$



Alors μ est un **homomorphisme** de q_2 vers q_1. Donc, $q_1 \sqsubseteq q_2$. Donc, $q_1 \sqcup q_2 \equiv q_2$. Il suffit donc de conserver q_2.

Soit θ la substitution définie par :

$$\theta(x_3) = x_2,$$
$$\theta(x_4) = x_1,$$
$$\theta(z_5) = z_1,$$
$$\theta(z_4) = z_2,$$
$$\theta(z_6) = z_3,$$

et θ est l'identité sur les autres variables.

Si l'on applique θ à la deuxième règle, on obtient la règle équivalente suivante :

$$Answer(y) \leftarrow R(y,x_1,z_1),\ R(y,x_2,z_2),\ R(x_1,x_2,z_3) \tag{1}$$

Cette requête est **minimale**.

- Le premier atome ne peut être éliminé qu'à l'aide d'une substitution contenant $x_1 \mapsto x_2$, ce qui transformerait le troisième atome en un atome de la forme $R(x_2,x_2,\cdot)$, qui n'existe pas dans la requête.
- Le deuxième atome ne peut pas être éliminé pour des raisons symétriques.
- Le troisième atome ne peut être éliminé qu'avec une substitution contenant $x_1 \mapsto y$, ce qui transformerait le premier atome en un atome de la forme $R(y,y,\cdot)$, qui n'existe pas non plus.

En conclusion, voici la requête équivalente minimale (avec renommage des variables pour une meilleure lisibilité) :

$$Answer(u) \leftarrow R(u,v,z_1),\ R(u,w,z_2),\ R(v,w,z_3)$$

Question 7 Si A est un ensemble, on note $|A|$ le nombre d'éléments de A.

Un *graphe dirigé simple* est un couple $G = (V, E)$ avec V un ensemble fini de *sommets* et E un ensemble d'*arcs* (u, v) avec $u, v \in V$, $u \neq v$. On traite seulement des graphes sans sommets isolés, i.e., chaque sommet a au moins un arc sortant ou entrant. Chaque arc du graphe est coloré en une et une seule couleur.

Dans une base de données, un prédicat EDB R d'arité 3 est utilisé pour stocker les arcs du graphe et leur couleur. Par exemple, $R(a, c, \mathsf{red})$ signifie qu'il existe un arc dirigé de a à c avec la couleur rouge.

On définit un *chemin dirigé de u_1 à u_n* comme une séquence $\langle u_1, u_2, \ldots, u_n\rangle$ de sommets telle que $n \geq 2$ et pour chaque $i \in \{1, 2, \ldots, n-1\}$, (u_i, u_{i+1}) est un arc du graphe. La *longueur* de ce chemin est de $n - 1$, i.e., la longueur est le nombre d'arcs parcourus. Noter que la condition $n \geq 2$ dans la définition précédente implique que la longueur d'un chemin n'est jamais zéro. Noter aussi qu'un sommet peut se répéter dans un chemin. On définit le *colorama* d'un chemin comme la séquence des couleurs de ses arcs. Formellement, le *colorama* du chemin $\langle u_1, u_2, \ldots, u_n\rangle$ est la sequence $\langle c_1, c_2, \ldots, c_{n-1}\rangle$ telle que pour chaque $i \in \{1, \ldots, n-1\}$, $R(u_i, u_{i+1}, c_i)$ est vrai. On dira qu'un chemin *utilise* la couleur c si c apparaît dans son colorama. Deux coloramas $\langle c_1, c_2, \ldots, c_k\rangle$ et $\langle d_1, d_2, \ldots, d_\ell\rangle$ sont *égaux* si $k = \ell$ et $c_i = d_i$ pour tout $i \in \{1, 2, \ldots, k\}$.

Par exemple, pour le graphe de la Figure 5, le colorama du chemin $\langle f, d, a, b, f, g, h, i, j\rangle$ est

$$\langle \mathsf{green}, \mathsf{red}, \mathsf{green}, \mathsf{blue}, \mathsf{pink}, \mathsf{red}, \mathsf{green}, \mathsf{blue}\rangle.$$

Écrivez un programme en Datalog avec $\neq$ et négation stratifiée pour le prédicat IDB unaire T tel que $T(x)$ est vrai si x est un sommet pour lequel il existe un chemin de x à x dont le colorama

— utilise **exactement trois couleurs distinctes**, ni plus ni moins ; et
— est tel que deux couleurs adjacentes sont toujours différentes.

De façon formelle, $T(x)$ est vrai s'il existe un chemin de x à x avec un colorama de la forme $\langle c_1, c_2, \ldots, c_k\rangle$ où $|\{c_1, c_2, \ldots, c_k\}| = 3$ et pour tout $i \in \{1, 2, \ldots k-1\}$, $c_i \neq c_{i+1}$.

Voici un exemple d'un colorama avec exactement trois couleurs—à savoir green, red et blue—dans lequel deux couleurs adjacentes sont toujours différentes :

$$\langle \mathsf{green}, \mathsf{red}, \mathsf{green}, \mathsf{blue}, \mathsf{red}, \mathsf{blue}, \mathsf{green}, \mathsf{red}, \mathsf{green}, \mathsf{red}\rangle.$$

Pour le graphe de la Figure 5, le programme doit renvoyer, entre autres, $T(f)$, car $\langle f, d, a, b, f\rangle$ est un chemin de f à f avec colorama $\langle \mathsf{green}, \mathsf{red}, \mathsf{green}, \mathsf{blue}\rangle$. Notez qu'il n'y a aucun problème au fait que la couleur green soit utilisée deux fois sur ce chemin, car ces deux occurrences ne sont pas adjacentes.

Avant de donner votre programme, expliquez en français la stratégie suivie par votre programme. Avant chaque règle (ou groupe de règles), expliquez en français ce que cette règle (ou ces règles) vont calculer.

. . . /10

Imaginons un chemin de x à y utilisant exactement trois couleurs, sans que deux arêtes adjacentes aient la même couleur. Supposons que ce chemin commence par une arête de couleur C, suivie d'une arête de couleur D, avec $C \neq D$. À un certain moment, une troisième couleur E doit apparaître pour la première fois, ce qui donne lieu à une séquence de couleurs (un *colorama*) de la forme $\langle \ldots, C, D, E\rangle$ ou $\langle \ldots, D, C, E\rangle$. On peut alors conclure qu'il existe un endroit dans cette séquence où les trois couleurs apparaissent consécutivement.

Le prédicat $P(x, y, z, c, d, e, \ell, r)$, d'arité 8, signifie qu'il existe un chemin de x à y utilisant exactement et uniquement les trois couleurs c, d et e, tel que deux arêtes adjacentes aient toujours des couleurs distinctes. De plus, ℓ est la couleur de la première arête, et r celle de la dernière arête du chemin. Les couleurs ℓ et r sont nécessaires pour vérifier que l'ajout d'une arête, à gauche ou à droite, respecte bien la contrainte selon laquelle deux arêtes adjacentes ne peuvent pas avoir la même couleur.

```
        p(X,Z,C,D,E,C,E) :- r(X,Y,C), r(Y,U,D), r(U,Z,E), C != D, D != E, C != E.
 p(X,Z,C,D,E,LeftColor,C) :- p(X,Y,C,D,E,LeftColor,RightColor), r(Y,Z,C), C != RightColor.
 p(X,Z,C,D,E,LeftColor,D) :- p(X,Y,C,D,E,LeftColor,RightColor), r(Y,Z,D), D != RightColor.
 p(X,Z,C,D,E,LeftColor,E) :- p(X,Y,C,D,E,LeftColor,RightColor), r(Y,Z,E), E != RightColor.
p(X,Z,C,D,E,C,RightColor) :- r(X,Y,C), p(Y,Z,C,D,E,LeftColor,RightColor), C != LeftColor.
p(X,Z,C,D,E,D,RightColor) :- r(X,Y,D), p(Y,Z,C,D,E,LeftColor,RightColor), D != LeftColor.
p(X,Z,C,D,E,E,RightColor) :- r(X,Y,E), p(Y,Z,C,D,E,LeftColor,RightColor), E != LeftColor.
                    t(X) :- p(X,X,C,D,E,LeftColor,RightColor).
```

Une autre stratégie consisterait à calculer le chemin de x à y de gauche à droite, tout en maintenant un booléen (`UsesE`) que l'on passe de 0 à 1 dès que la troisième couleur E est rencontrée. Dans ce programme, il suffit de mémoriser la dernière couleur du chemin (`RightColor`).

```
color(C)             :- r(X,Y,C).
p(X,Z,C,D,E,0,D)     :- r(X,Y,C), r(Y,Z,D), color(E), C != D, D != E, C != E.
p(X,Z,C,D,E,UsesE,C) :- p(X,Y,C,D,E,UsesE,RightColor), r(Y,Z,C), C != RightColor.
p(X,Z,C,D,E,UsesE,D) :- p(X,Y,C,D,E,UsesE,RightColor), r(Y,Z,D), D != RightColor.
p(X,Z,C,D,E,1,E)     :- p(X,Y,C,D,E,UsesE,RightColor), r(Y,Z,E), E != RightColor.
t(X)                 :- p(X,X,C,D,E,1,RightColor).
```

Finalement, on peut raffiner ce programme en observant que les couleurs C, D et E peuvent être permutées sans perte d'information. Dans le programme suivant, le prédicat $P(x, y, e, c, d, b)$, d'arité 6, encode un chemin de x à y où c et d désignent respectivement l'avant-dernière et la dernière couleur utilisées. Le booléen b (valeur de `ThreeColorsUsed`) passe de 0 à 1 dès que la troisième couleur distincte est rencontrée.

```
                    color(C) :- r(X,Y,C).
               p(X,Z,E,C,D,0) :- r(X,Y,C), r(Y,Z,D), color(E), C != D, D != E, C != E.
p(X,Z,E,D,C,ThreeColorsUsed) :- p(X,Y,E,C,D,ThreeColorsUsed), r(Y,Z,C).
               p(X,Z,C,D,E,1) :- p(X,Y,E,C,D,ThreeColorsUsed), r(Y,Z,E).
                         t(X) :- p(X,X,C,D,E,1).
```

Question 8 Écrivez un programme en Datalog avec $\neq$ et négation stratifiée pour le prédicat IDB unaire $S(x)$ tel que $S(x)$ est vrai si x est un sommet respectant les deux conditions suivantes :

1. il existe un chemin de x à x ; et
2. aucun chemin de x à x n'utilise la couleur green.

Pour le graphe de la Figure 5, le programme doit renvoyer $S(h)$ et $S(g)$. Notez que $S(f)$ n'est pas dans la réponse, car $\langle f, d, b, f\rangle$ est un chemin de f à f qui utilise la couleur green.

Avant de donner votre programme, expliquez en français la stratégie suivie par votre programme. Avant chaque règle (ou groupe de règles), expliquez en français ce que cette règle (ou ces règles) vont calculer.

.../10

`greenPath`(x, y) signifie qu'il existe un chemin de x à y qui utilise au moins une arête verte. Le cas de base correspond à un chemin constitué d'une seule arête verte. Ensuite, un chemin qui contient une arête verte peut être prolongé à gauche et à droite par des arêtes de n'importe quelle couleur. Il est important que cette extension s'effectue dans les deux sens, ce qui permet à l'arête verte de se trouver au milieu du chemin.

```
greenPath(X,Y) :- r(X,Y,green).
greenPath(X,Z) :- greenPath(X,Y), r(Y,Z,C).
greenPath(X,Z) :- r(X,Y,C), greenPath(Y,Z).
```

`reachable`(x, y) signifie qu'il existe un chemin de x à y.

```
reachable(X,Y) :- r(X,Y,C).
reachable(X,Z) :- reachable(X,Y), r(Y,Z,C).
```

La règle suivante conclut ce programme.

```
s(X) :- reachable(X,X), not greenPath(X,X).
```

Plusieurs étudiants ont écrit un programme qui renvoie tout sommet x tel qu'il existe un chemin de x à x n'utilisant pas d'arête verte. Or, ce n'est pas la requête demandée.

Il y a une différence entre :
— il existe un chemin de x à x qui n'utilise pas d'arête verte ;
— aucun chemin de x à x n'utilise une arête verte.
Par exemple, pour la base de données suivante, a n'est pas une réponse à la requête demandée, même s'il existe un chemin de a à a qui n'utilise pas d'arête verte.

R	1	2	3
	a	b	red
	b	a	red
	a	c	red
	c	a	green

Question 9 Pour le schéma $ABCDEF$, déterminez si $\bowtie [ABC, BDE, ADF]$ est une conséquence logique de l'ensemble $\{A \to B, B \to C, CD \to E, E \to F\}$. Détaillez les calculs. En plus, cochez la case qui précède une assertion correcte :

☒ $\{A \to B, B \to C, CD \to E, E \to F\} \models \bowtie [ABC, BDE, ADF]$;
☐ $\{A \to B, B \to C, CD \to E, E \to F\} \not\models \bowtie [ABC, BDE, ADF]$.

. . . /10

Application du chase.

A	B	C	D	E	F
a	b	c	d_1	e_1	f_1
a_2	b	c_2	d	e	f_2
a	b_3	c_3	d	e_3	f
a	b	c	d	e	f

Appliquez $A \to B$:

A	B	C	D	E	F
a	b	c	d_1	e_1	f_1
a_2	b	c_2	d	e	f_2
a	b	c_3	d	e_3	f
a	b	c	d	e	f

Appliquez $B \to C$ deux fois :

A	B	C	D	E	F
a	b	c	d_1	e_1	f_1
a_2	b	c	d	e	f_2
a	b	c	d	e_3	f
a	b	c	d	e	f

Appliquez $CD \to E$:

A	B	C	D	E	F
a	b	c	d_1	e_1	f_1
a_2	b	c	d	e	f_2
a	b	c	d	e	f
a	b	c	d	e	f

Il est justifié de conclure que l'implication logique est vraie et de s'arrêter là.

Question 10 Soit $\mathcal{H} = (V, E)$ un hypergraphe. Un *cycle de Berge* est une séquence alternée :

$$v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_1$$

où :
- $k \geq 2$,
- $v_1, \ldots, v_k$ sont des sommets distincts de V,
- $e_1, \ldots, e_k$ sont des hyperarêtes distinctes de E,
- pour chaque $i \in \{1, \ldots, k\}$, on a $\{v_i, v_{i+1}\} \subseteq e_i$ (avec $v_{k+1} = v_1$).

Par exemple, un graphe avec les hyperarêtes ABD, BCD, et ACE contient un cycle de Berge :

$$A, ABD, B, BCD, C, ACE, A.$$

Démontrez qu'un hypergraphe sans cycle de Berge est α-acyclique.

$\boxed{\ldots/5}$

Soit $\mathcal{H}$ un hypergraphe sans cycle de Berge. Il s'agit de montrer que $\mathcal{H}$ est α-acyclique.

Deux stratégies sont possibles :
- démontrer que l'algorithme GYO permet de supprimer toutes les hyperarêtes ;
- démontrer que $\mathcal{H}$ admet un arbre de jointure.

Adoptons la deuxième stratégie.

Choisissons un attribut, et construisons une chaîne composée de toutes les hyperarêtes contenant cet attribut. Sans perte de généralité, on peut supposer que l'attribut choisi est A (on peut toujours renommer les attributs) ; appelons cette chaîne la *chaîne de A*.

Choisissons ensuite un deuxième attribut dans la *chaîne de A*, que l'on peut supposer être B. Notons que B n'apparaît qu'une seule fois dans la *chaîne de A*. En effet, si B apparaissait deux fois dans la *chaîne de A*, disons dans les hyperarêtes ABX et ABY, alors la séquence A, ABX, B, ABY, A formerait un cycle de Berge, ce qui contredirait l'hypothèse selon laquelle $\mathcal{H}$ ne contient pas de tels cycles.

On ajoute alors une chaîne contenant toutes les hyperarêtes comportant B, appelée la *chaîne de B*.

On choisit ensuite un troisième attribut, noté C, dans le graphe construit jusqu'à présent, et l'on ajoute une chaîne contenant toutes les hyperarêtes comportant C : c'est la *chaîne de C*.

Et ainsi de suite. Voir figure 1 pour un exemple.

Les arbres ainsi construits sont bien des arbres de jointure. En effet, par construction, tous les nœuds contenant un même attribut forment une chaîne. De plus, tout attribut qui n'a pas encore été traité n'apparaît que dans un seul nœud.

En particulier, la situation illustrée par la figure 2, dans laquelle un nouvel attribut J apparaît dans deux nœuds distincts, ne peut pas se produire. En effet, une telle configuration engendrerait un cycle de Berge : J, $\boxed{CJ\cdots}$, C, $\boxed{BC\cdots}$, B, $\boxed{ABD\cdots}$, D, $\boxed{DJ}$, J, ce qui contredirait l'hypothèse selon laquelle $\mathcal{H}$ ne contient aucun cycle de Berge.

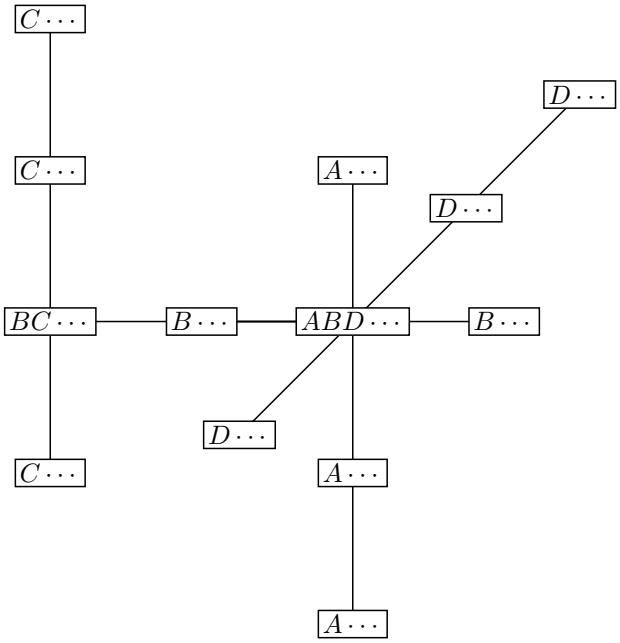


FIGURE 1 – Résultat d'une construction.

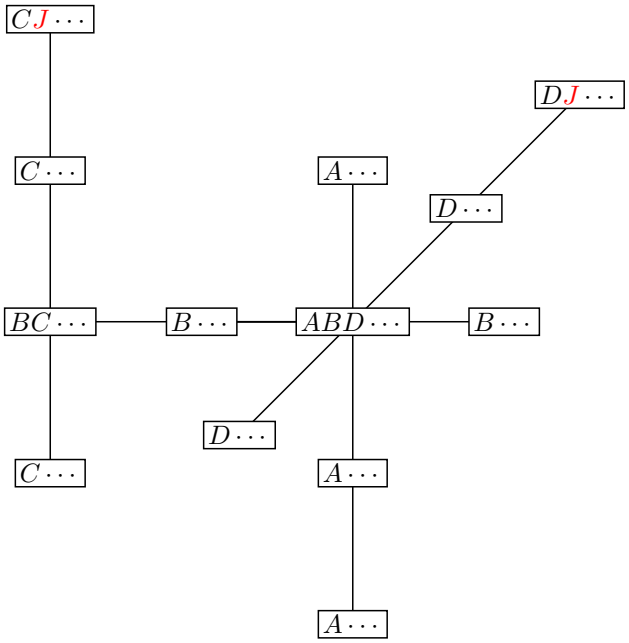


FIGURE 2 – Cycle de Berge.

```xml
<?xml version="1.0" encoding="utf-8"?>
<fleuriste>
  <fleurs>
    <!-- Les prix sont en centimes d'euro -->
    <fleur fnom="tulipe" prix="100"/>
    <fleur fnom="rose" prix="150"/>
    <fleur fnom="iris" prix="250"/>
    <fleur fnom="tournesol" prix="300"/>
    <fleur fnom="dahlia" prix="120"/>
  </fleurs>
  <bouquets>
    <bouquet bnom="Valentin">
      <fleur fnom="rose" couleur="rouge" nombre="10"/>
    </bouquet>
    <bouquet bnom="Belge">
      <fleur fnom="tulipe" couleur="noir" nombre="3"/>
      <fleur fnom="tulipe" couleur="jaune" nombre="4"/>
      <fleur fnom="tulipe" couleur="rouge" nombre="6"/>
    </bouquet>
    <bouquet bnom="Exotique">
      <fleur fnom="tournesol" couleur="jaune" nombre="1"/>
      <fleur fnom="iris" couleur="bleu" nombre="3"/>
    </bouquet>
    <bouquet bnom="Printemps">
      <fleur fnom="rose" couleur="jaune" nombre="10"/>
      <fleur fnom="tulipe" couleur="jaune" nombre="4"/>
      <fleur fnom="dahlia" couleur="rose" nombre="3"/>
    </bouquet>
  </bouquets>
  <ventes-par-jour>
    <date mois="sep 2021" jour="14">
      <vente bnom="Exotique">4</vente>
    </date>
    <date mois="juin 2022" jour="30">
      <vente bnom="Valentin">4</vente>
      <vente bnom="Printemps">1</vente>
      <vente bnom="Belge">2</vente>
    </date>
    <date mois="sep 2022" jour="18">
      <vente bnom="Valentin">3</vente>
      <vente bnom="Printemps">6</vente>
    </date>
    <date mois="sep 2022" jour="19">
      <vente bnom="Valentin">2</vente>
      <vente bnom="Belge">6</vente>
    </date>
  </ventes-par-jour>
</fleuriste>
```

FIGURE 3 – Document XML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE fleuriste [
<!ELEMENT fleuriste (fleurs, bouquets, ventes-par-jour)>
<!ELEMENT fleurs (fleur)*>
<!ELEMENT bouquets (bouquet)*>
<!ELEMENT bouquet (fleur)*>
<!ELEMENT fleur (#PCDATA)>
<!ELEMENT ventes-par-jour (date)*>
<!ELEMENT date (vente)*>
<!ELEMENT vente (#PCDATA)>
<!ATTLIST fleur fnom CDATA #REQUIRED>
<!ATTLIST fleur prix CDATA #IMPLIED>
<!ATTLIST fleur couleur CDATA #IMPLIED>
<!ATTLIST fleur nombre CDATA #IMPLIED>
<!ATTLIST bouquet bnom CDATA #REQUIRED>
<!ATTLIST date mois CDATA #REQUIRED>
<!ATTLIST date jour CDATA #REQUIRED>
<!ATTLIST vente bnom CDATA #REQUIRED>
]>
```

FIGURE 4 – DTD

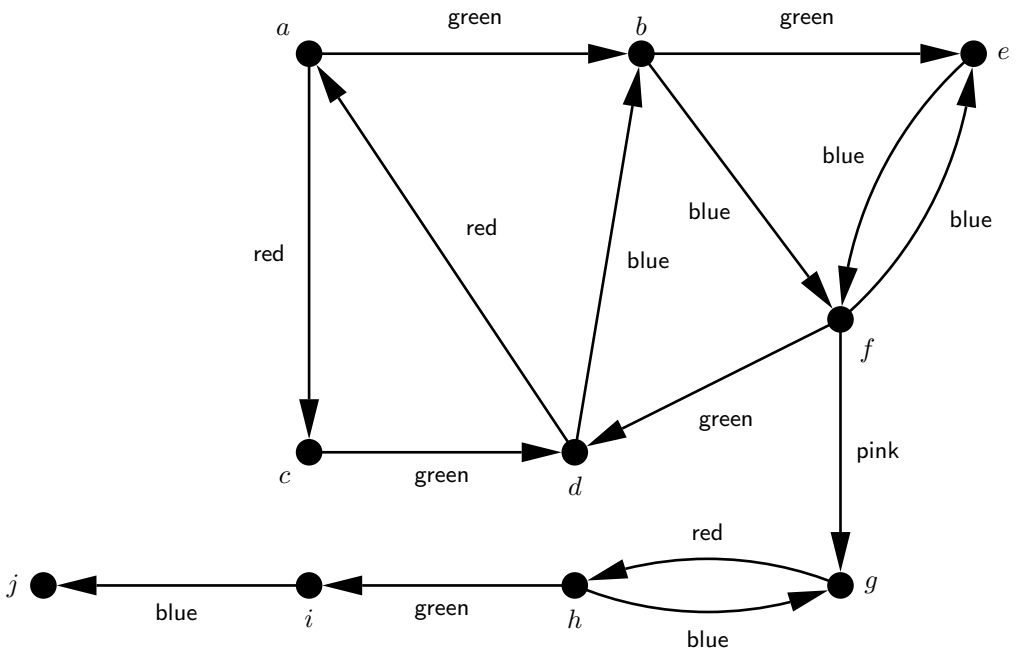


FIGURE 5 – Graphe dirigé avec des arcs colorés