Consignes pour la SAE S1.02 “comparaison d’approches algorithmiques”

Cette SAE compte pour 40% de la note de l’UE1.2

Vous devez réaliser, par groupe de deux étudiants, un exécutable JAVA sur le thème des relations binaires qui ont été définies dans le cours de maths discrètes. Cet exécutable sera à rendre à la fin du semestre 1 (dimanche 15 janvier à 23h59).
Le texte qui suit précise les définitions mathématiques et les consignes pour réaliser cette SAE.

1) Rappel de la définition d’une relation binaire:

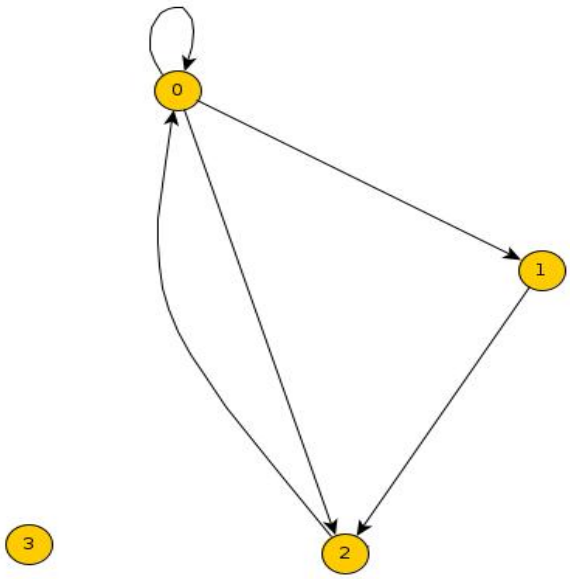
(Nous avons légèrement adapté les définitions du cours de maths pour faciliter la programmation en java)

Pour simplifier, les éléments d’un ensemble E de cardinal n (avec n un entier > 0) seront notés comme les entiers de 0 à $n-1$:

$$E = \{0, 1, 2, \ldots, n-1\}$$

Une relation binaire R de E vers E (avec $|E| = n \neq 0$) est un sous-ensemble de $E \times E$ c’est à dire un ensemble de couples d’éléments de E .

Exemple:
La relation binaire R définie par $R = \{(1,2), (0,2), (0,0), (2,0), (0,1)\}$ et $n = 4$ (la donnée de l’entier n permet de préciser l’ensemble E sur lequel la relation opère).
Cette relation peut être représentée par le graphe simple orienté dont voici une représentation:



Cette relation binaire peut aussi être représentée par une matrice (booléenne) d’adjacence $M = (m_{ij})$ de dimension $n \times n$:

$$M = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Dans cette matrice chaque coefficient est un booléen(0 ou 1 ou, ce qui revient au même F ou V). Pour tout $i \in \{0, 1, \ldots, n-1\}$ et pour tout $j \in \{0, 1, \ldots, n-1\}$ le coefficient $m_{i,j} = 1$ ssi l’arc (i, j) appartient à la relation binaire (remarque: on compte comme en “java”. La première ligne de la matrice est la ligne 0 et la première colonne est la colonne 0). Les autres coefficients sont nuls.

Il est possible aussi de représenter cette relation binaire par un tableau contenant pour chaque élément x de E l’ensemble de ses successeurs (c’est à dire les éléments y de E tels que $(x, y) \in R$):

sommets	successeurs
0	$\{0,1,2\}$
1	$\{2\}$
2	$\{0\}$
3	$\emptyset$

(informatiquement il s'agit donc d'un tableau d'ensembles d'entiers)

De manière symétrique, il est possible aussi de représenter cette relation binaire par un tableau contenant pour chaque élément x de E l'ensemble de ses prédécesseurs (c'est à dire les éléments y de E tels que $(y,x) \in R$):

sommets	prédécesseurs
0	$\{0,2\}$
1	$\{0\}$
2	$\{1,0\}$
3	$\emptyset$

On note xRy lorsque $(x,y) \in R$. Dans notre exemple on a $0R0$ et aussi $1R2$

2) Description générale du travail à réaliser:

L'objectif de cette SAE sera de programmer des méthodes en java (ces méthodes seront associées à la classe **RelationBinaire**) permettant de donner les principales caractéristiques d'une relation binaire (réflexif, antiréflexif, symétrique, antisymétrique, et transitif), de savoir si une relation binaire est une relation d'ordre, de donner la relation binaire correspondant au diagramme de Hasse d'une relation d'ordre, de donner la fermeture transitive d'une relation binaire.

Pour réaliser ces méthodes nous aurons besoins de méthodes "outils":

A) **Une relation binaire étant représentée par une matrice booléenne, nous aurons besoin de méthodes outils issues de la logique et du calcul matriciel**:

- Addition terme à terme de 2 matrices booléennes carrées de même dimension (correspondant au "ou" logique terme à terme)
- Multiplication terme à terme de 2 matrices booléennes carrées de même dimension (correspondant au "et" logique terme à terme). Attention, cette opération n'a rien à voir avec la multiplication matricielle.
- Négation d'une matrice booléenne carrée (on prend la négation de tous les coefficients un par un)
- Implication entre 2 matrices booléennes carrées (implication "terme à terme").
- Equivalence entre 2 matrices booléennes carrées (équivalence "terme à terme").
- Transposée d'une matrice booléenne.
- Multiplication de 2 matrices booléennes carrées de même dimension.

B) Une relation binaire étant un ensemble de couples, nous aurons aussi besoin de méthodes "outils" issues de la théorie des ensembles:

- Appartenance d'un couple donné à une relation binaire.
- Union de deux relations binaires.
- Intersection de deux relations binaires.
- Complémentaire d'une relation binaire.
- Différence de deux relations binaires.
- Inclusion d'une relation binaire dans une autre.
- Egalité de deux relations binaires.

C) Une relation binaire étant entièrement représentée par un graphe simple orienté nous aurons besoin de méthodes issues de la théorie des graphes (que vous aborderez au second semestre):

- Ensembles des successeurs d'un sommet du graphe.
- Ensembles des prédecesseurs d'un sommet du graphe.
- Ensemble des descendants d'un sommet du graphe (cette méthode sera définie dans l'extension 1).

D) Avec toutes les méthodes "outils" précédentes nous pourrons programmer les méthodes relatives aux relations binaires:

- Réflexif (cette méthode renvoie vrai si la relation est réflexive, faux sinon).
- Antiréflexif.
- symétrique.
- Antisymétrique.
- transitif.
- Relation d'ordre (cette méthode renvoie vrai si la relation est une relation d'ordre, faux sinon)
- Hasse (cette méthode renvoie une relation binaire correspondant au diagramme de Hasse d'une relation d'ordre).
- Fermeture transitive.

3) Quelques compléments mathématiques:

Dans cette partie nous allons vous donner quelques définitions mathématiques que vous n'avez pas vues en cours. Les autres définitions sont dans le cours de maths discrètes (pour l'algèbre de Boole, les calculs ensemblistes, les relations binaires) ou celui d'algèbre (pour le calcul matriciel).

A) définitions concernant les méthodes "outils" relatives aux matrices et à la logique:

On considère dans cette partie deux matrices carrées $n \times n$, $A = (a_{i,j})$ et $B = (b_{i,j})$ dont les coefficients sont des booléens (0 ou 1).

- **l'addition booléenne** $A + B$ (ou, si on préfère le langage de la logique, la disjonction: $A \vee B$) renvoie une matrice C booléenne $n \times n$ définie par $C = (c_{i,j} = a_{i,j} + b_{i,j}$) où le signe + dans l'expression $a_{i,j} + b_{i,j}$ est celui de l'addition booléenne (on peut donc aussi écrire en logique $a_{i,j} \vee b_{i,j}$).
- On définit de même **la multiplication booléenne** $A \times B$ (ou la conjonction $A \wedge B$).
- **Le complément d'une matrice booléenne** A notée $\overline{A}$ (ou la négation notée $\neg A$ si on préfère le langage de la logique) est une matrice booléenne définie par $\overline{A} = (\overline{a_{i,j}})$ ("on remplace tous les 1 par des 0 et tous les 0 par des 1").
- **L'implication** $A \Rightarrow B$ est une matrice carrée $n \times n$ définie par $C = (c_{i,j} = (a_{i,j} \Rightarrow b_{i,j}))$où le signe $\Rightarrow$ dans l'expression $a_{i,j} \Rightarrow b_{i,j}$ est celui du cours de logique:

$a_{i,j}$	$b_{i,j}$	$a_{i,j} \Rightarrow b_{i,j}$
1	1	1
1	0	0
0	1	1
0	0	1

- **L'équivalence** $A \Leftrightarrow B$ est une matrice carrée $n \times n$ définie par $C = (c_{i,j} = (a_{i,j} \Leftrightarrow b_{i,j}))$où le signe $\Leftrightarrow$ dans l'expression $a_{i,j} \Leftrightarrow b_{i,j}$ est celui du cours de logique:

$a_{i,j}$	$b_{i,j}$	$a_{i,j} \Leftrightarrow b_{i,j}$
1	1	1
1	0	0
0	1	0
0	0	1

- **La transposée** d'une matrice $A = (a_{i,j})$ notée A^t est la matrice définie par $A^t = \left(a'_{i,j} = a_{j,i}\right)$ ("les lignes deviennent des colonnes").

- **Le produit de deux matrices carrées** booléennes (qu'on notera plutôt $A * B$ pour ne pas confondre avec $A \times B$) est définie exactement comme le produit de 2 matrices à coefficients réels à part que les opérations $+$ et $\times$ sont celles de l'algèbre de Boole et non pas celles des nombres réels.

Voici un exemple de ces opérations qui devrait rendre plus clair les définitions précédentes:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$A + B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$A \times B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\overline{A} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$A \Rightarrow B = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$A \Leftrightarrow B = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$A^t = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$A * B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

C) définitions concernant les méthodes "outils" relatives à la théorie des graphes:
On notera R une relation binaire. Un sommet x est un élément quelconque de l'ensemble $E = \{0, 1, \ldots, n-1\}$.

- **L'ensemble des successeurs** d'un sommet x est l'ensemble des sommets y tels que xRy. Dans l'exemple de l'introduction l'ensemble des successeurs de 0, que l'on notera succ(0) est $\{0, 1, 2\}$(c'est donc un ensemble d'entiers). De même $succ\,(2) = \{0\}$ et $succ(3) = \emptyset$.
- **L'ensemble des prédécesseurs** d'un sommet x est l'ensemble des sommets y tels que yRx. Dans l'exemple de l'introduction l'ensemble des prédécesseurs de 0, que l'on notera pred(0) est $\{0, 2\}$(c'est donc un ensemble d'entiers). De même $pred\,(2) = \{0, 1\}$ et $pred(3) = \emptyset$.
- **L'ensemble des descendants** de x est l'ensemble des sommets y pour lesquels il existe une suite finie de l sommets (avec $l \geq 1$), $y_1, y_2, \ldots, y_l = y$ tel que $xRy_1 \wedge y_1Ry_2 \wedge \ldots \wedge y_{l-1}Ry$ (graphiquement cela signifie qu'il existe un parcours de x vers y empruntant un ou plusieurs arcs). Dans l'exemple de l'introduction, l'ensemble des descendants de 2, noté desc(2) est $\{0, 1, 2\}$. Dans cet ensemble on retrouve le successeur de 2 (plus généralement $succ\,(x) \subseteq desc\,(x)$) on trouve le sommet 1 (car $2R0 \wedge 0R1$) et le sommet 2 lui-même (car $2R0 \wedge 0R1 \wedge 1R\,2$).

D) définitions concernant les méthodes relatives aux relations binaires:

- Une relation binaire R est une **relation d'ordre** ssi R est réflexive, antisymétrique et transitive.
- **Diagramme de Hasse d'une relation d'ordre R de E vers E**: il s'agit d'une représentation par un graphe simple orienté d'une relation d'ordre dans laquelle on enlève toutes les boucles et tous les arcs qui peuvent être retrouvés par transitivité. Si on note $H\,(R)$ la relation binaire correspondant au diagramme de Hasse de la relation d'ordre R, alors $H\,(R)$ est défini de manière ensembliste par:

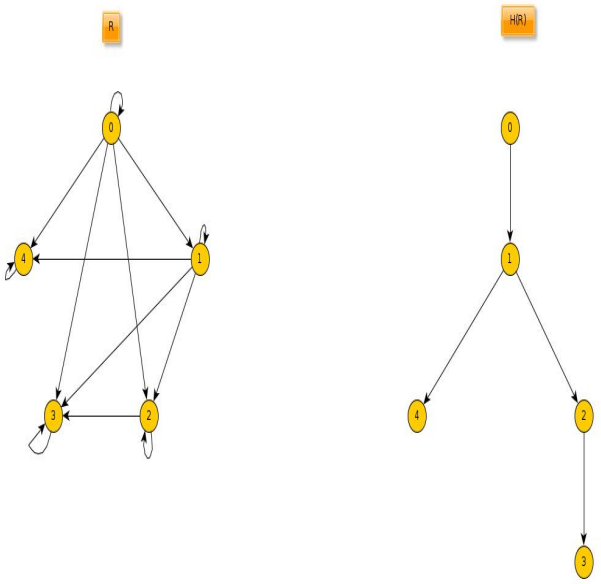
$$H\,(R) = R' - \bigcup_{x \in E} trav\,(x)$$

- avec $R' = R - \underset{x \in E}{\cup} \{(x,x)\}$(dans R' on enlève toutes les boucles de R (les arcs de la forme (x,x)).
- Pour $x \in E$, l'ensemble d'arcs $trav\,(x)$ est défini de la manière suivante:

$$trav\,(x) = pred\,(x) \times succ\,(x)$$

(c'est donc le produit cartésien de l'ensemble des prédécesseurs de x par l'ensemble des successeurs de x . Les prédécesseurs et les successeurs sont calculés dans la relation binaire R').

La figure ci-dessous donne un exemple de diagramme de Hasse:



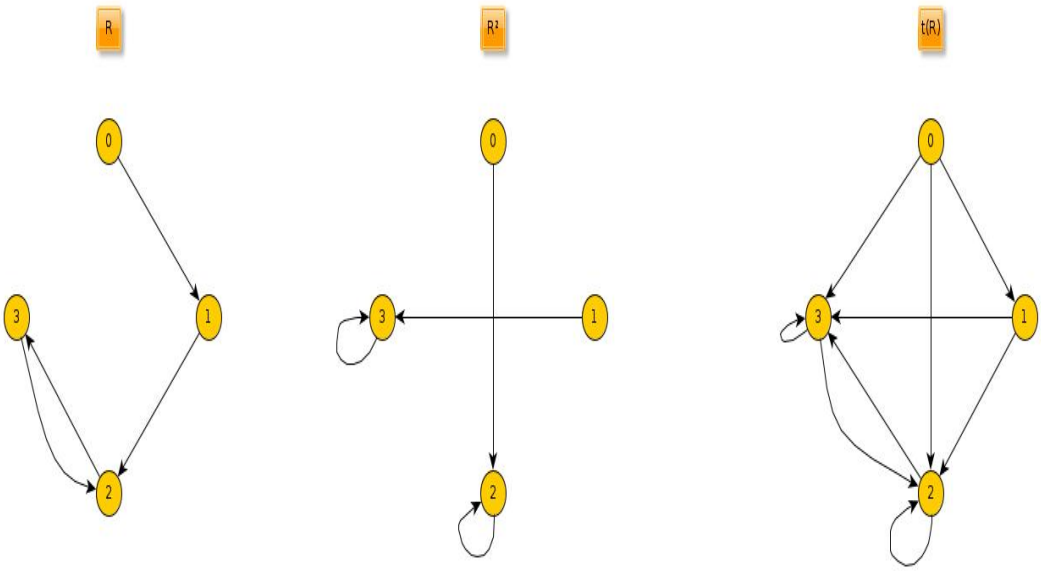
Pour plus de lisibilité si $(x,y) \in H\,(R)$ alors x est représenté par un sommet situé au-dessus du sommet représentant y. On évite, si possible, d'avoir des arcs qui se croisent.
La représentation graphique par un graphe simple et orienté de la relation $H\,(R)$ n'est pas demandée dans la version de base (elle fera l'objet de l'extension 3).

La fermeture transitive d'une relation binaire a été définie dans le cours de maths discrètes. Nous vous donnons ci-dessous deux définitions équivalentes:

- (théorème) et **définition 1**: la fermeture transitive $t\,(R)$ d'une relation binaire R est la plus petite relation binaire (du point de vue de l'inclusion) contenant R et transitive. Cela signifie que toute autre relation binaire R' contenant R et transitive vérifie $t\,(R) \subseteq R'$.
  Dans cette définition on admet l'existence d'une telle relation binaire. On pourrait le démontrer en construisant $t\,(R)$ comme l'intersection de toutes les relations binaires R' contenant R et transitive.

- **définition 2:** la fermeture transitive $t\,(R)$ d'une relation binaire R est définie par $t\,(R) = \cup_{k=1}^{n} R^k$ (où n est le cardinal de E).
  Pour comprendre cette définition il nous faut définir le produit de deux relations binaires R et S sur le même ensemble E :
  - $\forall a \in E, \forall b \in E,\ aRSb \Leftrightarrow (\exists c \in E, aRc \wedge cSb)$
  - dans la définition de $t\,(R)$ le terme R^k correspond au produit $RR....R$ où R apparaît k fois.

Dans l'exemple qui suit on a représenté successivement R, R^2 et $t\,(R)$:



4) Version de base:

Un squelette de code vous est donné qui contient la classe **RelationBinaire** ainsi que ses attributs et des noms de méthodes associées à cette classe (avec les entrées et la sortie attendue). C'est à vous de programmer chacune de ces méthodes.

Les attributs de la classe **RelationBinaire**:

- le cardinal n de l'ensemble E
- la matrice d'adjacence $matAdj$
- le cardinal m de la relation binaire (c'est à dire le nombre de couples)
- un tableau d'ensemble d'entiers $tabSucc$ de dimension n. le contenu de $tabSucc[i]$ est l'ensemble des successeurs du sommet i . Voici un exemple d'un tel tableau pour la relation de l'introduction:

$\{0,1,2\}$	$\{2\}$	$\{0\}$	$\emptyset$

Vous serez amené à programmer différents **constructeurs** en fonctions des attributs.

Voici la liste des méthodes:

méthodes	entrèes	sorties	description
toString	une relation binaire	une chaïne de caractères	matrice et couples
opBool	un connecteur et 2 matrices booléennes	une matrice booléenne	les 5 fonctions logiques
produit	deux matrices booléennes	une matrice booléenne	le produit matriciel
transposee	une matrice booléenne	une matrice booléenne	la transposée d'une matrice
estVide	une relation binaire	V ou F	
estPleine	une relation binaire	V ou F	
appartient	une relation binaire R et un couple c	V ou F	V ssi $c \in R$
ajouteCouple	une relation binaire R et un couple c	une relation binaire	$R \cup \{c\}$
enleveCouple	une relation binaire R et un couple c	une relation binaire	$R - \{c\}$
avecBoucles	une relation binaire	une relation binaire	
sansBoucles	une relation binaire	une relation binaire	
union	deux relations binaires	une relation binaire	
intersection	deux relations binaires	une relation binaire	
complementaire	une relation binaire	une relation binaire	
difference	deux relations binaires	une relation binaire	
estIncluse	deux relations binaires	V ou F	
estEgale	deux relations binaires	V ou F	
succ	une relation et un sommet	un ensemble d'entiers	successeurs
pred	une relation et un sommet	un ensemble d'entiers	prédécesseurs
estReflexive	une relation binaire	V ou F	
estAntireflexive	une relation binaire	V ou F	
estSymetrique	une relation binaire	V ou F	
estAntisymetrique	une relation binaire	V ou F	
estTransitive	une relation binaire	V ou F	
estRelOrdre	une relation binaire	V ou F	
hasse	une relation binaire	une relation binaire	
ferTrans	une relation binaire	une relation binaire	fermeture transitive
afficheDivers	une relation binaire		affichages

Remarques importantes:

- Dans cette version de base, vous devrez privilégier la performance des algorithmes en terme de temps de calcul. En particulier, vous devrez effectuer des **parcours partiels** plutôt que des parcours totaux, et pour parcourir l'ensemble des successeurs d'un sommet x, vous devrez parcourir l'ensemble $succ(x)$ et non faire un parcours total de la ligne d'indice x de la matrice d'adjacence.
  Vous n'aurez donc pas l'occasion d'utiliser les outils de logique et de calcul matriciel car ces outils effectuent des parcours totaux de matrices. L'utilisation de ces outils fera l'objet de l'extension 2.

- Vous devez obligatoirement utiliser la version de la casse EE fournie sans la modifier, de même que pour la classe Ut si vous l'utilisez. La méthode retraitUnElt a été ajoutée à la classe EE pour vous permettre de parcourir une instance de la classe EE, et la méthode ajoutPratique a été rendue publique pour vous permettre de l'utiliser à la place de ajoutElt (quand ses pré-requis sont vérifiés bien sûr !) pour une meilleure performance.

5) Extensions:

- **Extension 1: Descendants**

  Ajouter une méthode *EE descendant* $(int\, x)$ retournant l'ensemble des descendants d'un élément x de E dans la relation. Comme dans la version de base, cette méthode doit être la plus performante possible en termes de temps de calcul, et utiliser l'ensemble $succ(x)$ et non la matrice d'adjacence pour parcourir les successeurs d'un sommet x.

- **Extension 2: Variantes des méthodes**

  Définir des variantes des méthodes en privilégiant non plus la performance, mais l'utilisation des méthodes définies précédemment. Plus précisément, pour chaque méthode de estVide à descendants que vous avez écrite avec une ou plusieurs boucles, en écrire une variante ayant le même nom suivi du suffixe "Bis", ayant exactement les mêmes spécifications et n'effectuant aucune boucle, mais qui utilise des méthodes précédentes. Cas particuliers des méthodes $ferTransBis$ et $afficheDiversBis$, qui effectuent toujours des boucles : utiliser pour $ferTransBis$ celle des définitions 1 et 2 du sujet que

vous n'avez pas utilisée dans la version de base, et $afficheDiversBis$ est obtenue à partir de $afficheDivers$ en remplaçant chaque méthode par sa variante. Vous pourrez définir une méthode $RelationBinaire\ produitRel(RelationBinaire\ r)$(sans boucles !) retournant le produit des relations *this* et r, que vous placerez avant les méthodes qui l'utilisent.

- **Extension 3: Interface graphique**

  Réaliser une interface graphique pour représenter les graphes orientés associés aux relations binaires (voir UTILE/graphismeUpdate.zip sur Moodle).

- **Extension 4 : C.N. d'une relation d'ordre**

  On note$P(R)$ le prédicat : (R est une relation d'ordre) $\Rightarrow$ (la fermeture transitive de Hasse de R avec boucles est égale à R).
  Vérifier que lors de l'exécution de afficheDivers, $P(R_i)$ est vraie pour i de 0 à 4, et écrire une méthode

  $$booleanverifCNordre(intnbRel, intcardMax)$$

  retournant vrai ssi la prédicat $P(R)$ est vrai sur *nbRel* relations binaires choisies aléatoirement sur des ensembles de cardinaux aléatoires entre 1 et $cardMax$ avec des probas p aléatoires. Le but est d'appeler cette méthode avec une grande valeur de *nbRel* pour se convaincre que l'on est bien en présence d'une C.N. (Condition Nécessaire) d'une relation d'ordre.

- **Extension 5 : Fermeture ordonnée**

  La fermeture ordonnée d'une relation binaire R est la plus petite (du point de vue de l'inclusion) relation d'ordre contenant R, si elle existe .
  Contrairement à la fermeture transitive, la fermeture ordonnée n'existe pas toujours. On pourrait pourtant tenter de démontrer son existence en construisant l'intersection de toutes les relations d'ordre contenant R, mais le problème, c'est qu'il peut n'y avoir aucune relation d'ordre contenant R, alors que pour la fermeture transitive, il y a au moins la relation pleine comme relation transitive contenant R.
  Définir une méthode $RelationBinaire\ ferOrdonnee()$ retournant la fermeture ordonnée de *this* si elle existe et la relation vide sinon.
  Trouver une C.N.S. (Condition Nécessaire et Suffisante) sur une relation binaire, facilement reconnaissable sur sa représentation graphique pour qu'elle ait une fermeture ordonnée, et définir une méthode retournant vrai ssi this vérifie cette C.N.S.
  Ecrire une méthode $booleanverifCNSferOrdo(intnbRel, int\ cardMax)$ similaire à celle de l'extension 4 pour se convaincre que l'on est bien en présence d'une C.N.S. pour qu'une relation ait une fermeture ordonnée.

- **Extension 6 : niveaux**

  Dans le diagramme de Hasse ci-dessus, les sommets sont disposés par niveaux, de sorte que tous les arcs sont dirigés de haut en bas. Essayer de deviner comment ces niveaux ont été définis, puis vérifier que la définition trouvée correspond (ou pas !) à la définition ci-dessous.
  On définit la séquence des niveaux $(E_0, E_1, ...E_i, ..., E_k)$ d'une relation R dans un ensemble E par récurrence sur i :

  - E_0 est l'ensemble des éléments de E qui n'ont aucun prédécesseur pour R (ensemble des éléments placés au niveau le plus haut),

  - pour tout $i \geq 0$, les ensembles E_0, E_1, ...E_i étant définis, E_{i+1} est l'ensemble des éléments de $E - (\cup_{j=0}^{i} E_j)$ dont tous les prédécesseus pour R sont dans $\cup_{j=0}^{i} E_j$,

  - k est le plus petit entier i tel que $\cup_{j=0}^{i} E_j = E$, si cet entier existe.

  La séquence de niveaux existe ssi l'entier k existe.
  Définir une méthode $EE[]seqNiveaux()$ retournant la séquence de niveaux de *this* si elle existe et le tableau à une case contenant l'ensemble vide sinon (sachant qu'aucun élément d'une séquence de niveaux ne peut être vide).
  Trouver une C.N.S. sur une relation binaire, facilement reconnaissable sur sa représentation graphique pour qu'elle ait une séquence de niveaux, et définir une méthode retournant vrai ssi *this* vérifie cette C.N.S.
  Ecrire une méthode $boolean\ verifCNSniveaux(int\ nbRel, int\ cardMax)$ similaire à celle de l'extension 4 pour se convaincre que l'on est bien en présence d'une C.N.S. pour qu'une relation ait une séquence de niveaux.
  Si vous avez réalisé l'extension 3 (interface graphique) représenter graphiquement les relations binaires par niveaux quand c'est possible.

- **Extension 7 : Utilisation de la classe Liste**

  Réécrire la version de base en définissant l'attribut *tabSucc* comme un tableau d'instances de la classe *Liste* vue en cours et en TD.

A l'attention des amateurs de mathématiques

Le moyen le plus sûr de se convaincre du statut de C.N. ou de C.N.S. est d'en faire la démonstration mathématique. On ne vous demande pas de rendre les preuves de la C.N. de l'extension 4 ni des C.N.S. des extensions 5 et 6, mais vous pouvez essayer de les rédiger par plaisir et/ou souci de rigueur scientifique. Vous pourrez éventuellement faire vérifier vos preuves par votre enseignant de mathématiques.