Informatique théorique

Nhan LE THANH
DUT informatique
janvier 2007

Plan du cours

1- Structures usuelles
2- Raisonnement par récurrence et dénombrement
3- Raisonnement par induction et suites récurrentes
4- Logique proportionnelle et prédicative
5- fonction récursives et Machine Turing
6- Calculabilité et complexité
7- Classe NP

Informatique et mathématiques discrètes

- Circuits imprimés
- Réseaux
- Bases de données
- Compilation des langages de programmation
- compression de données
- Preuves de programmes
- Efficacité des algorithmes:
- Classes de problèmes
- Programmation fonctionnelle
- Sécurité
- Images
- Systèmes distribués
- Calcul booléen
- Théorie des graphes
- Logique
- Théorie des langages et
- des automates
- Induction
- Complexité
- Calculabilité
- l-calcul
- Cryptographie
- Géométrie algorithmique
- p-calcul

Séance 1: Structures usuelles

Plan

1- Ensembles et ensembles finis cardinalité
2- Relations
3- Fonctions et Applications
4- Mots, Langage
5- Chemins, Graphes, Arbres

I.1 Ensembles et éléments

- Un ensemble d'éléments est une collection d'objets distincts
- Un ensemble est défini par les éléments qu'il contient et qui lui appartiennent
- La relation d'appartenance d'un élément à un ensemble est notée $\in$ (ex: $x \in E$)
  - L'unique ensemble qui ne contient aucun élément est l'ensemble vide, noté $\varnothing$.
  - Les éléments d'un ensemble ne sont pas ordonnés entre eux.
  - Un ensemble peut être élément d'un autre ensemble, *mais pas de lui-même !* (voir le paradoxe de Russell)
  - La relation d'inclusion entre deux ensembles est notée $\subseteq$:
  - $A \subseteq B$ ssi $(x \in A \Rightarrow x \in B)$; donc $A=B$ ssi $(A \subseteq B$ et $B \subseteq A)$

I.1 Calcul ensembliste

- Opérations sur les ensembles :
  - Union : $x \in A \cup B$ ssi $(x \in A$ ou $x \in B)$
  - Intersection : $x \in A \text{Ç} B$ ssi $(x \in A$ et $x \in B)$
  - Complémentaire : $x \in \bar{A} = \mathbf{C}(A)$ ssi $x \notin A$
  - Différence : $B \backslash A = B \cap \bar{A}$
  - Différence symétrique : $A \Delta B = A \backslash B \cup B \backslash A$
  - Produit cartésien: $(x, y) \in A \times B$ ssi $x \in A$ et $y \in B$
- Note : deux ensembles sont dits disjoints si leur intersection est vide.

I.1- Propriétés des opérateurs

Soient A, B et C trois ensembles.

- Commutativité :
  - $A \cup B = B \cup A$
  - $A \cap B = B \cap A$
- Associativité :
  - $(A \cup B) \cup C = A \cup (B \cup C)$
  - $(A \cap B) \cap C = A \cap (B \cap C)$
- Distributivité :
  - De l'union par rapport à l'intersection (à gauche et à droite)
    - $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
    - $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$
  - De l'intersection par rapport à l'union (à gauche et à droite)
    - $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
    - $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
- Lois de Morgan :
  - $\overline{A \cup B} = \overline{B} \cap \overline{A}$
  - $\overline{A \cap B} = \overline{B} \cup \overline{A}$

I.1 Définition d'un ensemble

Un ensemble peut être défini :

- **en extension** : par la liste exhaustive de ses éléments
- **en compréhension** : par une propriété vérifiée par ses éléments (en général déjà définis dans un (sur-)ensemble donné)
- ex: les entiers naturels pairs (dans le sur-ensemble IN)
- *Remarque : Une propriété ne suffit pas toujours à définir un ensemble !*

I.1 Paradoxe de Russell

- Considérons la définition
  - E = { F tel que F est un ensemble }
  - E apparaît ainsi défini comme l'ensemble de tous les ensembles.
- Remarques :
  - « F est un ensemble » est bien une propriété mais ici le sur-ensemble serait E lui-même...
  - si E était un ensemble bien défini alors on aurait $E \in E$!
  - Par l'absurde, si E est bien un ensemble alors on peut aussi définir l'ensemble
    - $A = \{G \in E \text{ tel que } G \notin G\}$
  - De deux choses l'une :
    - soit $A \in A$ et alors $A \notin A \Rightarrow$ contradiction
    - soit $A \notin A$: et alors $A \in A \Rightarrow$ contradiction aussi
  - D'où le paradoxe. ($\Rightarrow$ pas d'ensemble de tous les ensembles)

I.1- Ensembles finis et cardinalité

- Le cardinal d'un ensemble est un concept permettant de se représenter son «nombre d'éléments ».
- On distingue notamment :
  - Les ensembles finis (dénombrables au sens large), dont les cardinaux sont les entiers naturels.
  - Les ensembles (infinis) dénombrables, en bijection avec IN, de cardinal noté $\aleph_0$.
  - Les ensembles (infinis) non-dénombrables, impossibles à mettre en bijection avec IN.

I.1- Cardinaux finis

- On note |E |le cardinal d'un ensemble E fini :
  - Si E et F sont deux ensembles disjoints alors
    $|E \cup F| = |E| + |F|$
  - Si (E_i) est une partition d'un ensemble E alors
    $|E| = \Sigma_i |E_i|$
  - Si E et F sont deux ensembles quelconques alors
    $|E \cup F| = |E| + |F| - |E \cap F|$
  - Formule du Crible *(généralisation à n ensembles* (E_i) $1 \leq i \leq n)$

$$|\cup_{1\leq i\leq n} E_i| = \Sigma_{1 \leq k \leq n} (-1)^{k+1} \Sigma_{1\leq i1<i2<\ldots<ik\leq n} |E_{i1} \cap E_{i2} \cap \ldots \cap E_{ik}|$$

I.1 Cardinaux finis (2)

- Soient E, F et les $(E_i)_{1\leq i\leq n}$ des ensembles finis.
- Principe multiplicatif
  - $|E \times F| = |E| * |F|$
  - Généralisation à n ensembles :
    $|E_1 \times E_2 \times \ldots \times E_n| = P_{1\leq i\leq n} |E_i|$
- Principe d'égalité
- Il existe une bijection entre E et F ssi
  $|E| = |F|$

I.1 Cardinaux finis (3)

Soient E et F deux ensembles finis.

- Principe d'inégalité ou *principe des tiroirs* : (*pigeon-hole principle*)
  - Si |E |> |F |alors il n'existe pas d'injection de E dans F.
  - *Interprétation équivalente :*

    Si n objets sont dans m tiroirs et si n>m, alors il existe au moins un tiroir qui contient au moins deux objets.

I.1 Dénobrabilité

Un ensemble infini est dénombrable :

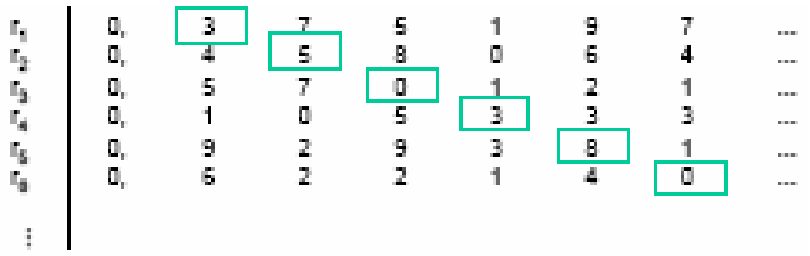
- s'il est en bijection avec IN
- s'il peut être injecté dans IN
- si ses éléments peuvent être « numérotés »
- Exemples :
  - IN2 est dénombrable : La fonction bijective $f(x,y) = y+(x+y)(x+y+1)/2 = y+(1+2+\ldots+(x+y))$ permet de numéroter IN^2.
- IN^n est en fait dénombrable pour tout n.
- Remarque : IR n'est pas dénombrable (voir la diagonale de Cantor).
- Propriétés : Est dénombrable :
  - Toute partie infinie d'un ensemble dénombrable
  - Tout produit cartésien fini d'ensembles dénombrables
  - Toute union dénombrable d'ensembles dénombrables

I.1 Parties d'un ensemble

Soit E' un sous-ensemble (ou partie) d'un ensemble E.

- La fonction caractéristique de E' est : f : E → { 0, 1 } telle que
  - pour tout e ∈ E', f(e) = 1
  - pour tout e ∉ E', f(e) = 0
- L'ensemble des parties de E se note P(E). Donc A∈P(E) ssi A⊆E
  - on a toujours Ø ∈ P(E) et E ∈ P(E)
  - si E est fini |P(E) |= 2 |E|
    (c'est le nombre d'applications de E dans {0,1})
  - si E est infini dénombrable, P(E) n'est pas dénombrable

I.1 Diagonale de Cantor



- Supposons que les r_i forment une liste des réels de l'intervalle]0,1[.
- Considérons un réel x = 0,461275$d_7d_8d_9d_{10}d_{11}$… tel que di n'est pas égal à la ième décimale de r_i
- x ne figure pas dans notre liste donc elle est incomplète.
- ⇒ **l'ensemble IR des réels n'est donc pas dénombrable.**

I.2. Relations, Ordres

- Un **couple** est une **paire ordonnée** d'éléments.
- ex: les points (x,y)du plan de IN2 ou de IR2, les nom et prix d'un produit, les instances d'un objet en Java (à 2 attributs).
  - Le **produit cartésien** de E par F est un ensemble de couples: **E × F = { (e,f) /e ∈ E et f ∈ F }**
  - Une **relation binaire** R de E dans F est une partie de E × F. **R ⊆ E × F**
- Si E = F, on parle de relation sur E (ex: l'inclusion sur E=P(IN)).
  - Pour tout couple (e,f) de R, e est dit en relation avec f.
- (e,f) ∈ R se note également e R f
  - La relation binaire « vide » correspond au sous-ensemble ∅ de E × F.

I.2. Relations et ordres

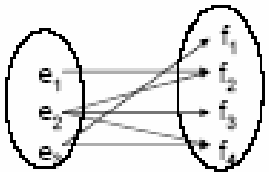
- la droite y=2x+1
  {(0,1),…, (1/2,2),…, (1,3),…,(2,5),…,(3,7),…,(4,9),…}
- la relation *« est parent de »* dans une famille
  {(Alice, Bob), (Alice, Chloé), (Dan, Elsa), (Dan, Bob), (Dan, Chloé), (Jules, Alice)}
- ordre strict ou non sur les entiers
  (0,1) est noté 0 < 1 ou (0,1) est noté 0 =\1
- ordre alphabétique : (a,z) est noté a <alph z
- fonction successeur sur les entiers : (7,8) est noté succ(7)=8
- relation d'égalité : (1,1) est noté 1 = 1
- relation de divisibilité : (12,132) est noté 12 | 132
- relation d'inclusion sur les parties d'un ensemble: ∅ ⊆ {a} {a} ⊆ {a,b,c}

I.2 Représentation d'une relation binaire

- **matrices binaires**

	f_1	f_2	f_3	f_4
e_1	0	1	0	0
e_2	0	1	1	1
e_3	1	0	0	1

- **diagrammes sagittaux**



- **diagrammes cartésiens**
  - tableaux à deux entrées, similaires aux matrices
- **graphes orientés**
  - seulement dans le cas où la relation est sur un ensemble.
  - on utilise un graphe non-orienté si la relation est symétrique.

I.2 Propriétés

- **réflexivité : pour tout x, x R x**
- **irréflexivité : pour tout x, x R x**
- **symétrie : x R y ⇒ y R x**
- **antisymétrie : (x R y et y R x)⇒ x = y**
- **transitivité : (x R y et y R z)⇒ x R z**
- *Remarque :*
  - Une relation irréflexive et transitive est toujours antisymétrique,
  - comme par exemple l'ordre strict < sur IR.

I.2 Relations d'équivalence

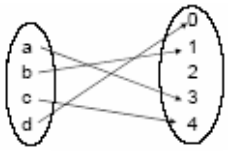
- Une **relation d'équivalence** est une relation sur un ensemble E: **Réflexive (R) - Symétrique (S) - Transitive (T)**
- Une relation d'équivalence R induit une partition de cet ensemble en **classes d'équivalence**.
- L'ensemble des classes est l'ensemble quotient E/R. Deux éléments en relation sont dans la même classe.
- *Exemples :*
  - l'égalité : que sont les différentes classes d'équivalence ?
  - les « cohortes » utilisées en démographie : la population française est partagée en classes d'individus tous nés la même année
  - les congruences : la congruence modulo 3 par exemple

I.3. Fonctions et Applications

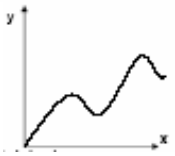
- Une **fonction** f d'un ensemble E dans un ensemble F est une relation R qui vérifie la propriété (x R y et x R z) $\Rightarrow$ y = z
- On note habituellement f(x) = y au lieu de x R y
- *La propriété précédente devient*
  (f(x) = y et f(x) = z) $\Rightarrow$ y = z
- Autrement dit, à tout élément on associe au plus une image.
- Une **application** est une fonction qui associe une image à tout élément de l'ensemble de départ.

I.2 Représentation d'une fonction

- **diagrammes sagittaux** (cas fini)



- **graphes** (cas infini pour les fonctions sur les entiers, les réels)



- **matrices** (pour les applications linéaires). Par exemple, le point (x,y,z) a pour image

(**x' = 2y ; y'= y + 3z ; z' = 2x+4z**)

$$\begin{pmatrix} 0 & 2 & 0 \\ 0 & 1 & 3 \\ 2 & 0 & 4 \end{pmatrix}$$

I.3 Propriétés d'une fonction

- Une application f de E dans F est :
  - **Injective** quand tout élément de F a **au plus un antécédent**
  - **Surjective** quand tout élément de F a **au moins un antécédent**
  - **Bijective**
    - quand tout élément de F a **exactement un antécédent**
    - quand elle est à la fois injective et surjective
- Théorème : Si |E |= |F | alors ces 3 propriétés sont équivalentes

I.2. Currification

- Soit f: $E_1 \times E_2 \rightarrow F$ une fonction à 2 variables. On appelle **currification** de f la fonction $f^c : E_1 \rightarrow [E_2 \rightarrow F]$ telle que pour tout $x_1 \in E_1$ et tout $x_2 \in E_2$ on a

  $$[f^c(x_1)]\ (x_2) = f(x_1,x_2)$$

- *Explication :*
  - Pour x_1 fixé, on note $f_{x1} : E_2 \rightarrow F :\ x_2 \rightarrow f(x1,x2)$
  - On a alors $f^c : E_1 \rightarrow [E_2 \rightarrow F] :\ x_1 \rightarrow f_{x1}$
- On dit qu'une fonction est « d'ordre supérieur » quand ses valeurs sont des fonctions.
- En programmation fonctionnelle (Lisp, Scheme…), on utilise ce mécanisme pour éviter les fonctions à plusieurs variables.

I.3 Relations d'ordre

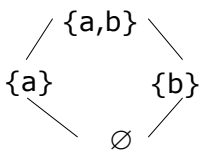
- Une relation d'**ordre large**, dit poset (partially ordered set), sur un ensemble est une relation
  Réflexive (R) - Antisymétique (A) - Transitive (T)
  Par défaut, on note $\leq$ un poset.
- Une relation d'**ordre strict** sur un ensemble est une relation **Irréflexive (I) - Transitive (T)**
- Une telle relation est donc aussi antisymétrique et s'appelle un quasi-ordre.
- Un ordre $\leq$ est dit **total** quand on a :
  $x \neq y \Rightarrow (x \leq y$ ou $y \leq x)$
- Un ordre est donc total quand il n'admet pas d'éléments incomparables. Sinon, il est **partiel**.

I.3 Relations d'ordre (2)

- Ordres partiels stricts
  - la relation *« est l'aïeul de »* dans une famille
- Ordres partiels
  - relation de divisibilité
  - inclusion sur les ensembles
- Ordres totaux stricts
  - ordre < sur les nombres
  - ordre alphabétique
- Ordres totaux
  - ordre ≤ sur les nombres
  - ordre lexicographique sur les mots

I.3 Représentation d'une relation d'ordre

- **Graphes orientés**
  - On peut choisir de faire figurer ou non les relations de réflexivité.
  - On aura des graphes «avec boucles» ou «sans boucle».
- **Diagrammes de Hasse**
  - Par définition, on ne fait pas figurer les relations de réflexivité ni celles déduites de la transitivité. On ne fait pas non plus figurer le sens dela relation par des arcs mais on oriente le diagramme dans son entier (les éléments les « plus petits » en bas ou à gauche)
  - Ex : (P({a,b}) ,⊆)



(IN, ≤) 0 — 1 — 2 — 3 — 4 — 5 — 6 — 7 ------

I.3 Linéarisation d'un ordre

Tout ordre partiel peut être étendu à un ordre total.

- **Algorithme du tri topologique**
  - Il s'applique au graphe d'un quasi-ordre.
  - Tant que le graphe contient un sommet :
    - 1) on choisit un sommet de degré entrant nul ;
    - 2) on le retire du graphe, ce qui décrémente le degré entrant de tous ses successeurs
  - La liste ordonnée obtenue est une **linéarisation** de l'ordre initial.
  - Elle définit un ordre total compatible avec l'ordre partiel de départ, c'est-à-dire qu'il le contient.

- *Remarque :* Plusieurs linéarisations peuvent être obtenues, selon les choix faits aux étapes 1) de l'algorithme.

I.3 Relations n-aire

- Les relations *n*-aires correspondent à des tableaux à *n* entrées, que l'on représente sur plusieurs colonnes.
- Considérons *n* ensembles $(E_i)_{1=vi=vn}$
  - Une relation R est un sous-ensemble de $E_1 \acute{} E_2 \times \ldots \times E_n$.
  - C'est donc un ensemble de ***n*-uplets** $(e_1, e_2, \ldots e_n)$.
  - *n* correspond à l'**arité** de la relation.

I.3 Relations n-aire (2)

- Les points (x,y,z) de IN_3, ceux du plan ax+by+cz+d=0 dans IR_3
- les instances d'un objet en Java
- le quadruplet (voir l'exemple de base de données qui suit) :

  (En avant, Les Alizés, danse, Opéra)

  indique la relation entre ces 4 informations.

I.3 Bases de données relationnelles

Spectacle	Compagnie	Type	Lieu
En avant	Les Alizés	danse	Opéra
De l'art	Cie ABC	théâtre	Gd-théâtre
Ciao	Alice H.	musique	Gd-théâtre
Fou rire	Vis Comica	théâtre	Opéra
Encre	B.T.J.	danse	Opéra
De passage	Bob Jr	musique	Gd-théâtre

Dates	oct	nov	déc
En avant	X		X
De l'art	X		X
Ciao	X	X	
Fou rire	X		X
Encre			
De passage		X	X

I.3 Bases de données relationnelles (2)

- En informatique, une **base de données relationnelles** permet de gérer un ensemble de relations.
- Les relations traduisent des informations diverses et variées.
- La base doit être organisée pour que l'on ait un accès rapide à chaque information.
- Une fois la base constituée, on lui adresse des requêtes (SGBD en langage SQL par exemple)
- Là encore, l'exécution des opérations nécessaires au traitement d'une requête doit être rapide.

I.3 Bases de données relationnelles (3)

- Voici celles appliquées temporairement aux relations d'une base, suite à une requête: **somme, produit, union, intersection, produit cartésien, complément**
- **la sélection** σ
  - Ex : quels sont les représentations de danse ? (2 lignes-réponses)
- **la projection** π
  - Ex : quels sont les types de spectacles joués à l'Opéra ?
  - On sélectionne les spectacles ayant lieu à l'Opéra (3 lignes-réponses).
  - On applique π à la réponse précédente pour ne garder que le type des spectacles (2 cases-réponses).
- **la jointure**
  - Ex : C'est le fait de fusionner en une relation deux relations existantes, avec des champs en commun.
  - Quels sont les noms des compagnies qui se produisent en novembre ?

I.4 Mots et langages

- un symbole ou une lettre est un caractère a, 0, 1, a, +, …
- un alphabet S est un ensemble dénombrable de symboles. En pratique, un alphabet est fréquemment fini
  - Exemple : {0,1}, {0,…,9}, {xi,i>0}, caractères Unicode
- un mot est une suite finie de symboles
  - Exemple : 010111001100, (5+7)/3,
  - class Hello { public static void main(String [] args) {…} }
  - le source d'une page HTML
- le mot vide ε, on appelle ainsi l'unique mot de longueur 0 (à distinguer du caractère ε). Il a surtout une utilité théorique
- l'ensemble des mots sur l'alphabet S se note S*
  - Exemple : si S={0,1}, S ={e,0,1,00,01,10,11,000,...}
- un sous-ensemble de S* s'appelle un langage (formel)
  - Exemple : l'ensemble des programmes écrits en C
  - les représentations binaires des int de Java
  - le langage de Dyck

I.4 Utilisation

- spécification de langages de programmation
- compilation
- recherche de motifs dans les éditeurs de texte, dans les programmes, dans une base de données, sur le web
- compression de textes
- preuves de programmes
- codage et décodage
- cryptographie
- décodage du génome
- calculabilité
- et aussi : linguistique, sciences cognitives, ...

I.4 Vocabulaire

- la longueur |m| d'un mot m est son nombre de lettres
  - Exemples $|\varepsilon| = 0$ $|010| = 3$
  - représentation binaire bn de n : $|b_n| = [\log_2 (n)] + 1$
- le nombre d'occurrences d'une lettre a dans un mot m est notée $|m|_a$
  - Exemple : $|010|_0 = 2$ $|010|_1 = 1$
- le ième caractère d'un mot m se note m[i], $1 \leq i \leq |m|$
- u est un facteur de m s'il existe v et w tels que m = v u w.
  - si $v = \varepsilon$, u est préfixe de m, si $w = \varepsilon$, u est suffixe de m.
  - Si $v \neq \varepsilon$ et $w \neq \varepsilon$, u est facteur propre, un préfixe ou un suffixe u est propre si $u \neq \varepsilon$ et $u \neq m$.
- on obtient un sous-mot d'un mot donné m en effaçant une ou plusieurs de ses lettres.
- le miroir d'un mot m est le mot obtenu en inversant le sens de lecture
  - Les palindromes sont les mots identiques à leur miroir (ex: elle, bob…).

I.4 Ordre sur les mots

- Soit $(A, <_{alph})$ un alphabet totalement et strictement ordonné,
  - $\leq_{alph}$ est l'ordre non strict associé à $<_{alph}$.
- Voici la définition de quelques ordres :
  - ordre préfixe : si un mot est préfixe d'un autre
  - (B) $\varepsilon \leq \varepsilon$, pour tout a de A : $\varepsilon \leq a$
  - (I) si $u \leq v$ alors, pour tout a de A : $u \leq v\,a$ et $a\,u \leq a\,v$
- ordre lexicographique (c'est celui du dictionnaire)
  - (B) $\varepsilon \leq_{lex} \varepsilon$, pour tout $(a,a') \subseteq A^2$ tel que $a \leq_{alph} a'$: $\varepsilon \leq_{lex} a$ et $a \leq_{lex} a'$
  - (I) si $u \leq_{lex} v$ et $|v| \leq |u|$ alors $u\,a \leq_{lex} v$ pour tout a de A
  - si $u \leq_{lex} v$ alors $a\,u \leq_{lex} a\,v$ et $u \leq_{lex} v\,a$ pour tout a de A
- ordre hiérarchique (aussi appelé militaire)
  - tri des mots sur la longueur puis à longueur donnée, tri lexicographique

I.4 Codage

Soit A et B deux alphabets.

- On appelle codage de A dans B toute application injective de A+ dans B+.
- Exemples
  - le codage d'un entier naturel en binaire : à chaque mot sur l'alphabet {0,1,2,...,9}, on associe sa représentation binaire sur {0,1}
  - le code Morse est un codage de l'alphabet latin sur l'alphabet {-,.}
  - le codage en b-aire, codage en b-adique : à chaque mot sur l'alphabet {0,1,...,9}, on associe sa représentation b-aire sur {0,1,…,b-1}, sa représentation b-adique sur {1,…,b}.

I.4 Opérations sur les langages

- Soit L et M deux langages donnés. On dispose des opérations ensemblistes (union, intersection, différences, complémentation, produit cartésien, ...)
  - définition du produit de concaténation de L et M :
    $L.M = \{w = uv / u \in L, v \in M\}$
  - définition récursive de la puissance de L :
    (B) $L^0 = \{ e \}$
    (I) pour tout i>0, $L^i = L.L^{i-1}$
  - définition de la fermeture de Kleene, notée *
    $L^* = \cup_{i \geq 0} L^i$
- Remarque :
  - on note $L^+ = \cup_{i>0} Li$.
  - si $L=\emptyset$, $L^0 = \{ \varepsilon \}$
  - L'ensemble des mots S* est bien la fermeture de Kleene de l'alphabet S.

I.4 Opérations sur les langages (2)

- L = {0,1,2,3,4}
- M = {5,6,7,8,9}
- $L \cup M$ = {0,1,2,3,4,5,6,7,8,9}
- L^2 = {00,01,02,03,04,10,11,12,13,14,20,21,22, 23,24,30,31,32,33,34, 40,41,42,43,44}
- L.M={05,06,07,08,09,15,16,17,18,19,25,26,27, 28,29,35,36,37,38,39,...}
- $(L \cup M)^+ = (L \cup M)^* \backslash \{\varepsilon\}$ est l'ensemble de toutes les représentations décimales, y compris celles avec des 0 inutiles en tête.

I.4 Propriétés

- Si K, L et M sont des langages, on a :
  - $(K^*)^* = K^*$
  - $L \cup \emptyset = \emptyset \cup L = L$
  - $K \cup L = L \cup K$
  - $K \cup (L \cup M) = (K \cup L) \cup M$
  - $L.\emptyset = \emptyset.L = \emptyset$
  - $L.\{\varepsilon\} = \{\varepsilon\}.L = L$
  - $K.(L \cup M) = K.L \cup K.M$
  - $(K \cup L).M = K.M \cup L.M$
  - $(K.L).M = K.(L.M)$
  - $(K \cup L)^*.K = (K^*.L)^* K^+$

I.4 Dénombrabilité

- Soit A = { a_0, a_1, a_2, a_3, ... } un alphabet ordonné.
  - L'ensemble des mots sur A est dénombrable
  - Preuve
    - on trie les éléments de A* par ordre hiérarchique
    - triés ainsi, on les indice par les entiers naturels
    - on a ainsi mis A* en bijection avec IN et donc prouvé que A* est dénombrable.
  - L'ensemble des langages sur A n'est pas dénombrable
  - Preuve (cf. transparent suivant)
  - supposons par l'absurde que l'ensemble des langages est dénombrable; cela revient à les indicer par les entiers naturels
  - on associe à chaque langage sa fonction caractéristique sur A*
  - l'argument de la diagonale de Cantor nous assure de l'existence d'un langage absent de la liste
  - on en déduit que l'ensemble des langages sur A* n'est pas dénombrable.

I.4 Dénombrabilité (2)

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	...
L_0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	...
L_1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	...
L_2	0	1	0	0	1	1	1	0	0	0	0	1	0	0	0	...
L_3	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	...
L_4	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	...
L_5	1	0	1	0	1	1	0	0	0	0	0	1	0	1	0	...
?																
⋮																

- On en déduit le langage L tel que : $a_i \in L$ ssi $a_i \in L_i$ ici L = {a_1, a_2, a_4, ... }
- Il ne figure pas dans la liste donc elle est incomplète
- Pour n'importe quelle liste, on pourrait exhiber un langage qui n'y appartient pas.
- Conclusion : l'ensemble des langages sur A n'est pas dénombrable.

I.5 Graphes, arbres et informatique

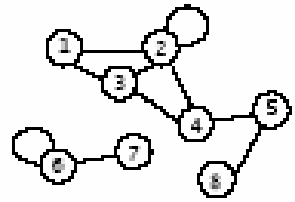
- Modélisation par les graphes :
  - relations, fonctions
  - réseaux (conception et routage)
  - circuits VLSI
  - automates
- Problèmes classiques sur les graphes :
  - recherche de plus courts chemins
  - ordonnancement de tâches
  - recherche opérationnelle
  - problèmes de flots
  - problème de l'arbre recouvrant
- Graphes comme structures de données

I.5 Graphes

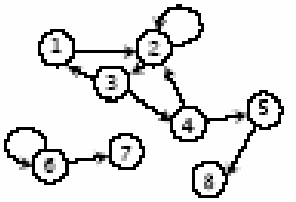
- Un graphe non orienté G = (S,A) est la donnée de 2 ensembles:
  - un ensemble fini S de sommets
  - un ensemble A d'arêtes, qui sont des paires de sommets (de la forme {s,t})
- Un graphe orienté G = (S,A) est la donnée de 2 ensembles:
  - un ensemble fini S de sommets
  - un ensemble A d'arcs, qui sont des couples de sommets (de la forme (s,t)) : $A \subseteq S \times S$
- Une arête {s,s} et un arc (s,s) sont appelés une boucle.
- On peut associer à G une valuation v : $A \rightarrow V$, V quelconque.

I.5 Graphe (2)

- Graphe G = (S,A) avec S = {1, 2, 3, 4, 5, 6, 7, 8} et
  A = {{ 1,2}, { 2,2}, { 2,3}, { 3,1}, { 3,4}, { 4,2}, { 4,5}, { 5,8}, { 6,6}, { 6,7}}
  puis avec A = {(1,2), (2,2), (2,3), (3,1), (3,4), (4,2), (4,5), (5,8), (6,6), (6,7)}
- Version non orientée



Version orientée



I.5 Représentation informatique de graphe

- liste des arêtes ou des arcs
  - ex: ((1, 2), (2, 2), (2, 3), (3, 1), (3, 4), (4, 2), (4, 5), (5, 8), (6, 6), (6, 7))
- matrice d'adjacence $M_G = (m_{ij})_{1 \le i,j \le |S|}$ $m_{ij} = 1$ si $(i,j) \in A$, $m_{ij} = 0$ sinon

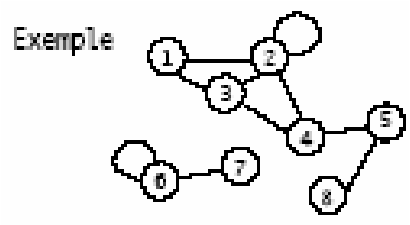
  0 1 0 0 0 0 0 0
  0 1 1 0 0 0 0 0
  1 0 0 1 0 0 0 0
  0 1 0 0 1 0 0 0
  0 0 0 0 0 0 0 1
  0 0 0 0 0 1 1 0
  0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0
- G valué, les coefficients sont $m_{ij} = v(i,j)$ pour tout $(i,j) \in A$, $m_{ij} = 0$ sinon.
- matrice d'incidence $N_G = (n_{ij})_{1 \le i \le |S|; 1 \le i \le |A|}$ n_{ij} est le nombre de fois où l'arc j est incident au sommet i.
- liste d'adjacence : liste des sommets et pour chacun, liste de ses successeurs.
  ((1, (2)), (2, (2, 3)), (3, (1, 4)), (4, (2, 5)), (5, (8)), (6, (6, 7)), (7, ( )), (8, ( )))

I.5 Vocbulaire

- Le nombre de sommets d'un graphe est appelé son ordre du graphe.
- Son nombre d'arêtes ou d'arcs est appelé sa taille.
- Le nombre d'arêtes ou d'arcs incidents à un sommet est son degré, noté d(s) .
- Pour une arête {s,t} donnée, s et t en sont les extrémités. Pour un arc (s,t) donné, s est l'extrémité initiale et t l'extrémité finale ; s est alors un prédécesseur de t, t un successeur de s.
- Une chaîne est une suite non vide d'arêtes adjacentes qui relient un sommet s à un sommet t. Un cycle est une chaîne reliant un sommet à lui-même.
- Un chemin est une suite non vide d'arcs consécutifs qui relient un sommet s à un sommet t. Un circuit est à un chemin ce qu'un cycle est à une chaîne.
- Un sous-graphe d'un graphe G est obtenu en retirant des arêtes/arcs à G. Un graphe partiel du graphe G est obtenu en retirant des sommets ainsi que leurs arêtes/arcs incidents.

I.5 Connexité et acyclicité

- Un graphe est connexe s'il existe une chaîne reliant toute paire de sommets.
- On peut partitionner l'ensemble des sommets en p classes $C_1,\ldots, C_p$ de telle sorte que les p graphes partiels $G_i = (C_i, A_i)_{1\le i\le p}$ soient connexes.
- Les $(G_i)_{1\le i\le p}$ sont appelées les composantes connexes de G.



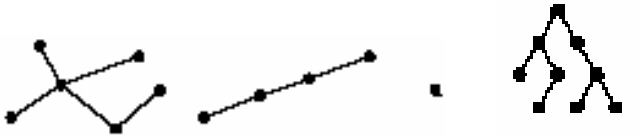
Ce graphe a
2 composantes
connexes.

I.5 Connexité et acyclicité (2)

- Théorème 1.5.1 : Tout graphe connexe d'ordre n>0 a au moins n-1 arêtes.
  - Démonstration par récurrence généralisée
    - si n=1 c'est vrai.
    - on suppose que la propriété est vraie aux ordres 1<k<n.
    - soit un graphe G = (S, A) à n sommets. s est un sommet de G. Considérons le graphe partiel G' dont l'ensemble des sommets est S\{s}.
    - Partitionnons G' en composantes connexes Gi=(Si, Ai)$_{1\le i\le p}$. G connexe donc
      p ≤ d(s)
    - On a |S| = 1 + $\Sigma_{1\le i\le p}$ |S| et |A|= $\Sigma_{1\le i\le p}$ |Ai|+ d(s).
    - Par hypothèse de récurrence, pour tout i, |Si|- 1≤ |Ai|
    - $\Sigma_{1\le i\le p}$ |Si| - p ≤ $\Sigma_{1\le i\le p}$ |Ai|
    - |S|-1 -p ≤ |A|- d(s)
    - |S|-1 ≤ |A|
- Théorème 1.5.2 (admis) : Tout graphe sans cycle d'ordre n>0 a au plus n-1 arêtes.

I.5 Arbre et propriétés

- Un arbre est un graphe connexe sans cycle.
- Théorème 1.5.3 : Soit un graphe G=(S, A) d'ordre n (n≥2). Les propriétés suivantes sont équivalentes:
  1. G est un arbre
  2. tout couple de sommets est relié par une chaîne unique
  3. G est sans cycle et en ajoutant une arête on en crée un
  4. G est connexe et si on supprime une arête il ne l'est plus
  5. G est sans cycle et admet n-1 arêtes
  6. G est connexe et admet n-1 arêtes
- exemples d'arbres :



I.5 Arbre et propriétés (2)

- Preuve
  - (1→2)
    - G est connexe donc tout couple de sommets est relié par une chaîne.
    - Elle est unique car sinon, on a un cycle.
  - (2→3)
    - Si il y avait un cycle, deux sommets ne seraient pas connectés par une chaîne unique. Si on ajoute une arête entre deux sommets qui n'étaient pas reliés, elle forme un cycle avec la chaîne déjà existante entre les deux sommets.
  - (3→1)
    - G est sans cycle. Prenons deux sommets dans G. Si il existe une arête entre ces deux sommets, a fortiori il existe une chaîne entre les deux. Supposons qu'il n'existe pas d'arête entre les deux sommets. En ajoutant une arête entre les deux sommets, on crée un cycle. C'est qu'il y avait une déjà une chaîne entre ces deux sommets. Donc G est bien connexe. G est connexe sans cycle, c'est donc un arbre.

I.5 Arbre et propriétés (3)

- Preuve (suite)
  - (1← 4)
    - G est un arbre donc G est connexe. Si on enlève une arête, on coupe l'unique chaîne qui reliait ses deux extrémités, le graphe n'est plus connexe.
    - supposons la propriété 4 vraie et aussi que G a un cycle. En supprimant une arête du cycle, G reste connexe, d'où la contradiction.
    - Donc G n'a pas cycle. Comme il est connexe, c'est un arbre.
  - (1← 5)
    - G est un arbre d'ordre n alors il est sans cycle et a n-1 arêtes.
    - G est sans cycle et a n-1 arêtes. Si on ajoute une arête, le graphe a n arêtes et donc on a créé un cycle (Théorème 1.5.2). La propriété 3 est vraie, elle implique la 1.
  - (4← 6)
    - G est un arbre d'ordre n alors il est connexe et a n-1 arêtes.
    - G est connexe et a n-1 arêtes. Si on enlève une arête, le graphe a n-2 arêtes. Il n'est plus connexe (Théorème 1.5.2). G vérifie la propriété 4.

I.5 Arbre et propriétés (4)

Propriété 1.5.4 : Un arbre d'ordre n>1 a au moins deux feuilles.
Preuve

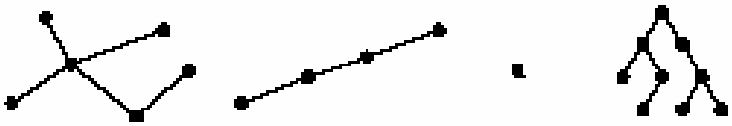
- On considère une chaîne de longueur maximale et d'extrémités s et t.
- Il existe un x tel que l'arête (s, x) existe.
- Il n'existe pas d'arête (s, y) pour un autre sommet y de la chaîne car sinon, on aurait un cycle.
- Il n'existe pas non plus d'arête (s, z) pour un sommet z hors de la chaîne car sinon, la longueur de la chaîne ne serait pas maximale.

Propriété 1.5.5 : Un arbre d'ordre n>0 a n-1 arêtes.

- Preuve:
  - Par récurrence sur l'ordre. Un arbre à 1 sommet a 0 arête.
  - Supposons qu'un arbre a n sommets, n>1, ait n-1 arêtes.
  - Considérons un graphe à n+1 sommets.
  - Il a au moins une feuille f. Le reste du graphe a n sommets donc, par hyp. de réc. il a n-1 arêtes. En ajoutant l'unique arête reliant f au graphe, il a n arêtes : cqfd.

I.5 Vocabulaire

- Forêt : Une forêt est un graphe dont chaque composante connexe est un arbre. Autrement dit, une forêt est un graphe sans cycle :



- Tout sommet d'un arbre est appelé un nœud.
- On appelle feuille un sommet d'un arbre adjacent à une seule arête.
- On peut distinguer un sommet particulier d'un arbre. On l'appelle racine et l'arbre sera dit enraciné.
- Une chaîne reliant la racine d'un arbre à une feuille est appelée une branche.

I.5 Arborescence

- Une arborescence k-aire est un arbre enraciné orienté dont chaque sommet a au plus k successeurs. Quand k=2, l'arborescence est binaire.
- Un arbre k-aire est complet si tous ses nœuds ont 0 ou k successeurs.
- Un niveau est un ensemble des nœuds qui ont en commun d'être équidistants de la racine. On les numérote à partir de 0.
- La hauteur d'un arbre binaire est le nombre d'arcs sur un chemin de longueur maximale. Sa profondeur est son nombre de niveaux.
- Un arbre complet est saturé si, pour une hauteur donnée, il a un nombre maximal de nœuds.

I.5 Propriétés des arbres binaires

- On considère un arbre binaire saturé à n nœuds.
  1. Le niveau i, i≥0, contient 2^i nœuds.
  2. Le nombre de feuilles est donc égal à 2^h (h est nombre de niveaux depuis la racine)
  3. Le nombre de nœuds est égal à $n = 2^{h+1}-1$
  4. La hauteur h est égale à $\log_2(n+1)-1$
  5. Le nombre de feuilles est égal à $(n+1)/2$.

I.5 Représentation informatique des arbres

- Représentation séquentielles des arbres binaires saturés
  - on prend un tableau à $n = 2^{h+1}-1$ cases, h étant la hauteur
  - la case n°0 contient la racine de l'arbre
  - la valeur du fils gauche du nœud i, $0<i\leq n$, est dans la case 2i
  - la valeur du fils droit du nœud i, 0<i!n, est dans la case 2i+1
- Représentation récursive des arbres binaires
  - un arbre est un pointeur sur un nœud
  - un nœud est une valeur et deux pointeurs
  - un sur l'arbre fils gauche
  - un sur l'arbre fils droit
- Représentation récursive des arbres k-aires
  - un arbre est un pointeur sur un nœud
  - un nœud a une valeur et deux pointeurs
  - un sur l'arbre appelé premier fils
  - un sur l'arbre appelé la liste des frères