Graphes et outils logiques

Thibaut Balabonski @ Université Paris-Sud

Première partie, 15 janvier 2020

Dans ce cours nous allons étudier des structures permettant de modéliser et résoudre des problèmes informatiques, et apprendre à raisonner sur ces structures. Le cours mélange des aspects mathématiques et algorithmiques, et vise à la fois à vous faire découvrir des structures de données et algorithmes fondamentaux en informatique et à affûter vos capacités de raisonnement.

Ce cours doit beaucoup au cours « Mathématiques pour l'informatique » de Chistine Paulin

Table des matières

0 Révisions : logique, raisonnement, ensembles

0.1 Langage logique

Objectifs

— Clarifier le discours en soulignant certaines articulations
— Fixer une interprétation commune du langage
— Décrire, justifier, argumenter

Ce qu'on écrit Une ***formule*** est construite avec les éléments suivants :

— des constantes : $\top$ (formule vraie) et $\bot$ (contradiction)
— des variables propositionnelles, désignant des faits ou formules indéterminés : A, B, C...
— des connecteurs : $\wedge$ (et), $\vee$ (ou), $\neg$ (non), $\Longrightarrow$ (implique), $\Longleftrightarrow$ (équivalent à), $\forall$ (pour tout), $\exists$ (il existe)...

Ce que cela signifie On fixe deux ***valeurs de vérité*** possibles : {Vrai, Faux} (ou $\{1,0\}$, ou $\{V,F\}$). Une ***interprétation*** ρ est une fonction associant une valeur de vérité à chaque variable propositionnelle.

On a deux niveaux extrèmes de validité/invalidité pour une formule :

— une formule est ***valide*** si elle est vraie pour toute interprétation, c'est-à-dire si elle est vraie par sa seule forme indépendamment de ce que représentent ses différentes variables (on dit aussi que cette formule est une tautologie),
— une formule est ***contradictoire*** si elle n'est vraie pour aucune interprétation.

Les négations de ces deux critères donnent des niveaux intermédiaires :

— une formule est ***satisfiable*** (ou non-contradictoire) si elle est vraie pour au moins une interprétation,
— une formule est ***invalide*** si elle est fausse pour au moins une interprétation.

Pour visualiser toutes les interprétations d'une formule, on peut utiliser une ***table de vérité*** : un tableau donnant, pour chaque combinaison de statut vrai ou faux de chaque variable propositionnelle, la véracité de la formule complète. Exemple :

A	B	C	$(A \wedge B) \vee (C \wedge B)$
F	F	F	F
F	F	V	F
F	V	F	F
F	V	V	V
V	F	F	F
V	F	V	F
V	V	F	V
V	V	V	V

Note : il y a 2^n interprétations différentes d'un ensemble de n variables, la table de vérité d'une formule à n variables propositionnelle a donc 2^n lignes.

Deux formules A et B sont ***équivalentes*** si elles sont respectivement vraies et fausses pour les mêmes interprétations exactement. On note alors $A \equiv B$.

0.2 Raisonnement : les règles de la justification

Principe Pour ***justifier*** qu'un fait donné est vrai, on déduit sa véracité à l'aide de règles de raisonnement en partant de certains faits de base supposés vrais.

On a donc toujours dans ce contexte un ensemble de formules (appelées ***hypothèses***) dont on suppose qu'elles sont vraies, et à partir desquelles on veut ***déduire*** qu'une certaine formule cible (la ***conclusion***) est vraie également.

Chaque articulation logique est associée à des règles de déduction de base, indiquant notamment :

— comment justifier une conclusion présentant cette articulation (règle d'introduction)

— comment utiliser une hypothèse basée sur cette articulation (règle d'élimination)

On a en plus un certain nombre de grandes techniques : raisonnement par l'absurde, tiers exclu, contradiction, récurrence...

À noter que l'on ne cherche jamais à justifier que les hypothèses sont elles-mêmes vraies. On veut simplement justifier qu'elles ne peuvent être vraies sans que la conclusion ne le soit elle aussi. D'ailleurs, on verra ci-dessous que certaines techniques de raisonnement consistent au contraire à montrer que les hypothèses ne peuvent pas être vraies.

0.3 Connecteurs de la logique propositionnelle

Conjonction : A **ET** B

Notation $A \wedge B$

Validité Formule vraie lorsque A, B sont toutes deux vraies.

Table de vérité

A	B	$A \wedge B$
F	F	F
F	V	F
V	F	F
V	V	V

Équivalences

$A \wedge B$	$\equiv$	$B \wedge A$	commutativité
$A \wedge (B \wedge C)$	$\equiv$	$(A \wedge B) \wedge C$	associativité
$A \wedge \top$	$\equiv$	A	$\top$ élément neutre
$A \wedge \bot$	$\equiv$	$\bot$	$\bot$ élément absorbant
$A \wedge (B \vee C)$	$\equiv$	$(A \wedge B) \vee (A \wedge C)$	distributivité de $\wedge$ sur $\vee$
$\neg(A \wedge B)$	$\equiv$	$\neg A \vee \neg B$	loi de De Morgan

Raisonnement

— Pour justifier $A \wedge B$: justifier les deux formules

— Pour réfuter $A \wedge B$: réfuter l'une des deux formules

— Une hypothèse $A \wedge B$ permet de justifier A

— Une hypothèse $A \wedge B$ permet de justifier B

Disjonction : A **OU** B

Notation $A \vee B$

Validité Formule vraie lorsqu'au moins une parmi A, B est vraie.

Table de vérité

A	B	$A \vee B$
F	F	F
F	V	V
V	F	V
V	V	V

Équivalences

$A \vee B$	$\equiv$	$B \vee A$	commutativité
$A \vee (B \vee C)$	$\equiv$	$(A \vee B) \vee C$	associativité
$A \vee \top$	$\equiv$	$\top$	$\top$ élément absorbant
$A \vee \bot$	$\equiv$	A	$\bot$ élément neutre
$A \vee (B \wedge C)$	$\equiv$	$(A \vee B) \wedge (A \vee C)$	distributivité de $\vee$ sur $\wedge$
$\neg(A \vee B)$	$\equiv$	$\neg A \wedge \neg B$	loi de De Morgan

Raisonnement

— Pour justifier $A \vee B$: justifier l'une des deux formules
— Pour réfuter $A \vee B$: réfuter les deux formules
— Si d'une part A permet de justifier C et si d'autre part B permet aussi de justifier C, alors $A \vee B$ permet de justifier C (raisonnement par cas)

Négation : NON A

Notation $\neg A$

Validité Formule vraie lorsque A est fausse.

Table de vérité

A	$\neg A$
F	V
V	F

Équivalences

$\neg\neg A$	$\equiv$	A	involutivité
$A \vee \neg A$	$\equiv$	$\top$	tiers exclu
$A \wedge \neg A$	$\equiv$	$\bot$	contradiction
$\neg A$	$\equiv$	$A \Longrightarrow \bot$	

Raisonnement

— Pour justifier $\neg A$: supposer l'hypothèse A et obtenir une contradiction
— Une hypothèse $\neg A$ peut être combinée à A pour produire une contradiction

Raisonnement par contradiction L'hypothèse $\bot$ permet de justifier n'importe quelle conclusion. Autrement dit, un ensemble d'hypothèses permettant de déduire la contradiction $\bot$ justifie peut justifier n'importe quelle formule.

Raisonnement par l'absurde Pour justifier une conclusion A, on peut prendre comme hypothèse $\neg A$ et en déduire la contradiction $\bot$.

Raisonnement par tiers exclu Pour justifier une conclusion C, on peut raisonner par cas sur la disjonction $A \vee \neg A$ pour une formule A de notre choix (cette disjonction est une tautologie, c'est-à-dire une formule tout le temps vraie, quelle que soit la formule A).
D'où : choisir une formule A puis,

— sous l'hypothèse A, justifier C
— sous l'hypothèse $\neg A$, justifier C également

Implication : SI A ALORS B

Notation $A \Longrightarrow B$

Validité Formule valide si toute interprétation rendant A vraie rend aussi B vraie (remarque : aucune contrainte sur les interprétations rendant A fausse).

Table de vérité

A	B	$A \Longrightarrow B$
F	F	V
F	V	V
V	F	F
V	V	V

Équivalences

$$\begin{array}{rcll} A \Longrightarrow B & \equiv & \neg A \vee B & \\ A \Longrightarrow B & \equiv & \neg B \Longrightarrow \neg A & \text{contraposée} \\ A \Longrightarrow (B \Longrightarrow C) & \equiv & (A \wedge B) \Longrightarrow C & \text{curryfication} \end{array}$$

Raisonnement

— Pour justifier $A \Longrightarrow B$: supposer l'hypothèse A et justifier B
— Pour réfuter $A \Longrightarrow B$: trouver une interprétation rendant A vraie et B fausse
— Une hypothèse $A \Longrightarrow B$ peut être combinée à A pour justifier B

Équivalence : A SI ET SEULEMENT SI B

Notation $A \Longleftrightarrow B$

Validité Formule vraie si A et B vraies sur les mêmes interprétations.

Définition à partir de l'implication

$$A \Longleftrightarrow B \quad \equiv \quad (A \Longrightarrow B) \wedge (B \Longrightarrow A)$$

0.4 Logique des prédicats

On ne regarde plus des formules logiques pures, mais des énoncés « à propos de certains objets ».

Contexte (ou ***théorie***)

— Le ***domaine*** désigne les objets dont on parle.
exemples : nombres entiers, ensembles...

— On a des ***opérations*** combinant les objets du domaines.
exemples : opérations arithmétiques, opérations ensemblistes

— Des ***prédicats*** définissent les faits logiques de base à propos des objets.
exemples : égalité, comparaison, appartenance, inclusion...

— On a un certain nombre de formules supposées vraies, les ***axiomes***, qui traduisent des propriétés de base des objets du domaine

Une ***expression*** t désigne un objet du domaine exprimé sous la forme d'une combinaison d'objets concrets et de variables (x, y, z...) par des opérations.
Exemples : $2 * (1 + x)$, $x \cap y$

Formules Elles sont construites avec :

— les constantes $\top$ et $\bot$

— les connecteurs habituels

— des prédicats appliqués à des expressions
exemple : $f(x) \in (Y \cup Z)$

— les quantificateurs $\forall$ et $\exists$

On peut expliciter le fait qu'une formule φ dépend des variables x et y en notant cette formule $\varphi(x, y)$. On note alors $\varphi(t, u)$ la formule φ dans laquelle chaque occurrence de la variable x est remplacée par l'expression t, et chaque occurrence de la variable y est remplacée par l'expression u.

Les quantificateurs s'utilisent de la manière suivante :

— $\forall x,\ \varphi(x)$

— $\exists x,\ \varphi(x)$

La variable utilisée avec un quantificateur est une variable n'existant que dans le fragment de formule considéré. Son nom n'importe pas et pourrait être remplacé par un autre nom. Les formules $\forall x,\ \varphi(x)$ et $\forall y,\ \varphi(y)$ sont égales.

0.5 Quantificateurs

Quantification universelle : POUR TOUT OBJET x, $\varphi(x)$

Notation $\forall x,\ \varphi(x)$

Validité Formule vraie si tous les objets du domaine vérifient la formule φ.

Équivalences

$$\begin{aligned} \forall x,\ \forall y,\ \varphi(x,y) &\equiv \forall y,\ \forall x,\ \varphi(x,y) \\ \neg \forall x,\ \varphi(x) &\equiv \exists x,\ \neg \varphi(x) \end{aligned}$$

Raisonnement

— Pour justifier $\forall x,\ \varphi(x)$, fixer une variable y désignant un objet arbitraire (c'est-à-dire un objet dont on ne sait rien, sur lequel on n'a aucune hypothèse), et justifier $\varphi(y)$

— Pour réfuter $\forall x,\ \varphi(x)$, donner explicitement un objet concret t pour lequel la formule $\varphi(t)$ est fausse

— Une hypothèse $\forall x,\ \varphi(x)$ permet de justifier $\varphi(t)$ pour n'importe quel objet t, qu'il soit donné directement par un objet concret ou par une expression dépendant d'une ou plusieurs variables

Quantification existentielle : IL EXISTE UN OBJET x TEL QUE $\varphi(x)$

Notation $\exists x,\ \varphi(x)$

Validité Formule vraie si au moins un objet du domaine vérifie la formule φ.

Équivalences

$$\begin{aligned} \exists x, \exists y,\ \varphi(x,y) &\equiv \exists y, \exists x,\ \varphi(x,y) \\ \neg\exists x,\ \varphi(x) &\equiv \forall x,\ \neg\varphi(x) \end{aligned}$$

Raisonnement

— Pour justifier $\exists x,\ \varphi(x)$, donner explicitement un objet concret t pour lequel la formule $\varphi(t)$ est vraie
— Pour réfuter $\exists x,\ \varphi(x)$, fixer une variable y désignant un objet arbitraire et réfuter $\varphi(y)$
— Une hypothèse $\exists x,\ \varphi(x)$ permet d'introduire un objet désigné par une nouvelle variable y et dont on sait qu'il vérifie la formule $\varphi(y)$ (on ne sait en revanche rien d'autre sur cet objet)

Variantes : quantification bornée

Les deux formes de quantifications peuvent être restreintes à un ensemble d'objets E donné.

Quantification universelle bornée

Notation $\forall x \in E,\ \varphi(x)$

Validité Formule vraie si tous les objets de l'ensemble E vérifient la formule φ.

Définition à partir de la quantification universelle

$$\forall x \in E,\ \varphi(x) \quad\equiv\quad \forall x,\ x \in E \Longrightarrow \varphi(x)$$

Quantification existentielle bornée

Notation $\exists x \in E,\ \varphi(x)$

Validité Formule vraie si au moins un objet de l'ensemble E vérifie la formule φ.

Définition à partir de la quantification existentielle

$$\exists x \in E,\ \varphi(x) \quad\equiv\quad \exists x,\ x \in E \wedge \varphi(x)$$

Attention : on a ici une conjonction ($\wedge$) alors que la formule équivalente pour la quantification universelle est construite avec une implication ($\Longrightarrow$).

0.6 Théorie arithmétique : récurrence

On prend comme domaine les nombres entiers, ainsi que les opérations et prédicats habituels.

En plus des règles de raisonnement de base, on a un principe de raisonnement par récurrence pour justifier des énoncés de la forme $\forall n \in \mathbb{N}, \varphi(n)$.

Si on a justifié les deux formules

— $\varphi(0)$

— $\forall n \in \mathbb{N}, \varphi(n) \Longrightarrow \varphi(n+1)$

alors on peut justifier

— $\forall n \in \mathbb{N}, \varphi(n)$

Variante : récurrence forte

Si on a justifié la formule

— $\forall n \in \mathbb{N}, (\forall k \in \mathbb{N}, k < n \Longrightarrow \varphi(k)) \Longrightarrow \varphi(n)$

alors on peut justifier

— $\forall n \in \mathbb{N}, \varphi(n)$

0.7 Théorie naïve des ensembles

Prédicat de base : $x \in A$ (appartenance d'un élément x à un ensemble A)

Autre prédicats pouvant être définis à partir de l'appartenance

— non appartenance : $x \notin A \equiv \neg(x \in A)$

— inclusion : $A \subseteq B \equiv \forall x, x \in A \Longrightarrow x \in B$

— égalité : $A = B \equiv A \subseteq B \wedge B \subseteq A \quad (\equiv \forall x, x \in A \Longleftrightarrow x \in B)$

Quand $A \subseteq B$ on dit aussi que A est une partie de B

Quelques ensembles et constructions

— $\emptyset$: ensemble vide

— $\mathbb{N}$: ensemble des entiers naturels

— $\{x_1, ..., x_n\}$: définition en extension

— $\{x \mid \varphi(x)\}$: définition en compréhension

— $\{x \in E \mid \varphi(x)\}$: définition en compréhension (version bornée)

— $[0,n[= \{0,1,\ldots,n-1\} = \{x \in \mathbb{N} \mid x < n\}$: segment initial des entiers

Opérations

$A \cup B$	$\equiv$	$\{x \mid x \in A \vee x \in B\}$	union
$A \cap B$	$\equiv$	$\{x \mid x \in A \wedge x \in B\}$	intersection
$A \setminus B$	$\equiv$	$\{x \mid x \in A \wedge x \notin B\}$	différence ensembliste
$A \times B$	$\equiv$	$\{(a,b) \mid a \in A \wedge b \in B\}$	produit cartésien
A^0	$\equiv$	$\{\emptyset\}$	puissance
A^{n+1}	$\equiv$	$A^n \times A$	
$\mathscr{P}(A)$	$\equiv$	$\{X \mid X \subseteq A\}$	ensemble des parties

Ensembles finis L'ensemble A est ***fini*** s'il existe

— un entier $n \in \mathbb{N}$, et

— une fonction f injective de A dans $[0,n[$.

Si f est bijective, on appelle n le ***cardinal*** de A (notation $|A|$).

0.8 Relations binaires

Une ***relation binaire*** $\mathscr{R}$ entre deux ensembles A et B est un sous-ensemble du produit cartésien $A \times B$. Dans ce contexte, on note couramment $a\mathscr{R}b$ ou $\mathscr{R}(a,b)$ pour $(a,b) \in \mathscr{R}$.

Opérations

- $\mathscr{R}^{-1} \equiv \{(b,a) \in B \times A \mid (a,b) \in \mathscr{R}\}$
  par exemple $\leq^{-1} \equiv \geq$
- $\neg\mathscr{R} \equiv \{(a,b) \in A \times B \mid (a,b) \notin \mathscr{R}\}$
  par exemple $\neg \leq \equiv >$
- $\mathscr{R}_1 \cup \mathscr{R}_2$, $\mathscr{R}_1 \cap \mathscr{R}_2$, $\mathscr{R}_1 \setminus \mathscr{R}_2$ avec les définitions usuelles
  par exemple $\leq \equiv < \cup =, et < \equiv \leq \cap \neq$

Relations particulières

- La relation $\mathscr{R}$ est ***totale*** si pour tout élément du domaine d'entrée A il existe une sortie associée :

$$\forall a \in A, \exists b \in B, a\mathscr{R}b$$

- La relation $\mathscr{R}$ est ***fonctionnelle*** si chaque élément de A est en relation avec au plus un élément de B :

$$\forall a \in A, \forall b_1 \in B, \forall b_2 \in B, a\mathscr{R}b_1 \wedge a\mathscr{R}b_2 \Longrightarrow b_1 = b_2$$

0.9 Fonctions

- Une relation binaire fonctionnelle entre A et B est aussi appelée une ***fonction*** de A dans B.
- Une relation binaire fonctionnelle et totale entre A et B est aussi appelée une ***application*** de A dans B.

Variante de vocabulaire : les applications sont aussi appelées fonctions totales, et les fonctions qui ne sont pas des applications des fonctions partielles.

Images et antécédents

- Étant donnée une fonction $f : A \longrightarrow B$ (autrement dit une relation fonctionnelle entre A et B) on note $f(a)$ pour l'unique b tel que $(a,b) \in f$, en supposant qu'un tel b existe. On appelle b l'***image*** de a, et a un ***antécédent*** de b.
- Étant donné un ensemble $A_0 \subseteq A$, on définit

$$f(A_0) = \{b \in B \mid \exists a, \in A_0 . f(a) = b\}$$

- Étant donné un ensemble $B_0 \subseteq B$, on définit

$$f^{-1}(B_0) = \{a \in A \mid f(a) \in B_0\}$$

- Étant donné un ensemble $A_0 \subseteq A$, la ***restriction*** de f à A_0 est la fonction $f' : A_0 \longrightarrow B$ telle que pour tout $a \in A_0$, $f'(a) = f(a)$.

Fonctions particulières

- Une fonction ***injective*** est une fonction $f : A \longrightarrow B$ telle que chaque élément du domaine d'arrivée B a au plus un antécédent :

$$\forall b \in B, \forall a_1, a_2 \in A, f(a_1) = b \wedge f(a_2) = b \Longrightarrow a_1 = a_2$$

— Une fonction ***surjective*** est une fonction $f : A \longrightarrow B$ telle que chaque élément du domaine d'arrivée B a au moins un antécédent :

$$\forall b \in B, \exists a \in A, f(a) = b$$

— Une fonction ***bijective*** est une fonction à la fois injective et surjective.

1 Points et traits (23/01)

Un ***graphe*** est formé par :
— un ensemble de points (les ***sommets***),
— reliés par des traits (les ***arêtes***).

Dans le cas d'un ***graphe orienté***, chaque arête a un sommet de ***départ*** et un sommet d'***arrivée***. Lorsque l'on ne distingue pas les deux extrémités d'une arête on parle d'un ***graphe non orienté***. Un ***chemin*** dans un graphe est une succession contiguë d'arêtes.

1.1 Graphes orientés

Un ***graphe orienté*** est donné par un triplet (S, A, ϵ) où
— S est un ensemble dont les éléments sont appelés ***sommets*** (ou nœuds),
— A est un ensemble dont les éléments sont appelés ***arêtes*** (ou arcs, ou flèches),
— ϵ est une application de A dans $S \times S$ donnant les ***extrémités*** de chaque arête.

Pour $a \in A$, $s_1 \in S$ et $s_2 \in S$ avec $\epsilon(a) = (s_1, s_2)$, on appelle s_1 le ***départ*** (ou la source) de a, et s_2 l'***arrivée*** (ou la cible) de a, et on dit que l'arête a va de s_1 vers s_2. On pourra aussi noter $\sigma(a) = s_1$ et $\tau(a) = s_2$. Deux sommets s_1 et s_2 tels qu'il existe une arête de s_1 vers s_2 ou de s_2 vers s_1 sont dits ***adjacents***. Une arête a telle que $\sigma(a) = s$ ou $\tau(a) = s$ est dite ***incidente*** à s.

Sous-graphes un ***sous-graphe induit*** d'un graphe (S, A, ϵ) est donné par
— S' une partie arbitraire de S ($S' \subseteq S$)
— A' l'ensemble des arêtes de A dont les extrémités sont dans S' ($A' \equiv \{a \in A \mid \epsilon(a) \in S' \times S'\}$)
— ϵ' la restriction de ϵ à A'

Arêtes particulières Une arête a ayant mêmes sommets de départ et d'arrivée est appelée une ***boucle***. Deux arêtes distinctes a_1 et a_2 telles que $\epsilon(a_1) = \epsilon(a_2)$ sont appelées ***arêtes parallèles***. On appelle ***graphe simple*** un graphe sans boucles ni arêtes parallèles.

Graphes et relations binaires Un graphe simple (S, A, ϵ) définit une relation binaire $\mathscr{R}$ sur $S \times S$ par la condition « $s_1 \mathscr{R} s_2$ s'il existe une arête $a \in A$ de s_1 vers s_2 ». Formellement :

$$s_1 R s_2 \equiv \exists a \in A, \epsilon(a) = (s_1, s_2)$$

Réciproquement, toute relation binaire $\mathscr{R}$ sur un ensemble E définit un graphe simple ayant E pour ensemble de sommets et ayant une arête de e_1 vers e_2 si et seulement si $e_1 \mathscr{R} e_2$.

Représentation graphique Un graphe peut être dessiné en représentant chaque sommet par un point du plan et chaque arête comme une flèche allant de son sommet de départ à son sommet d'arrivée.

Un graphe qui peut être dessiné dans le plan en deux dimensions sans que les arêtes ne se croisent est appelé ***graphe planaire***.

Exercices

— nombre maximal d'arêtes pour un graphe simple à n sommets : $n(n-1)/2$
— nombre de graphes simples à n sommets : $2^{(n(n-1)/2)}$

Exemples

— clique (K_n) : n sommets, toutes les arêtes (simples)
— bipartite complet ($K_{n,m}$) : deux ensembles de n et m sommets, chaque sommet d'un ensemble relié à tous ceux de l'autre ensemble
— ligne (L_n) : sommets $[0,n[$, arêtes de i à $i+1$
— cycle (C_n) : ligne L_n à laquelle on ajoute une arête de $n-1$ à 0
— grille ($G_{n,m}$) : sommets $[0,n] \times [0,m]$, arêtes entre deux sommets égaux sur une coordonnée et ayant une différence de 1 sur l'autre
— n-cube (H_n) : sommets mots binaires de longueur n, arêtes entre deux sommets égaux sur toutes les lettres sauf une

Exercices

— K_4 est planaire, mais pas K_5
— $K_{2,2}$ et $K_{2,3}$ sont planaires, mais pas $K_{3,3}$
— H_3 est planaire, mais pas H_4

1.2 Graphes non orientés

Un ***graphe non orienté*** est un graphe $G = (S, A, \epsilon)$ dans lequel on ne distingue pas le départ de l'arrivée des arêtes. La fonction extrémités ϵ est alors une application de A dans $\mathscr{P}(S)$ qui renvoie un ensemble de cardinal 1 ou 2.

1.3 Isomorphismes de graphes

Informellement, deux graphes sont identiques s'ils peuvent être représentés par le même dessin, éventuellement à renommage des sommets près.

Cette idée est formalisée par la notion d'isomorphisme de graphe : un ***isomorphisme*** entre deux graphes (S_1, A_1, ϵ_1) et (S_2, A_2, ϵ_2) est une paire de fonctions (f_s, f_a) telles que

— $f_s : S_1 \longrightarrow S_2$ et $f_a : A_1 \longrightarrow A_2$ sont des applications bijectives
— pour toute arête $a \in A_1$ avec $\epsilon(a) = (s,t)$ on a $\epsilon(f_a(a)) = (f_s(s), f_s(t))$ (notation alternative : $\sigma(f_a(a)) = f_s(\sigma(a))$ et $\tau(f_a(a)) = f_s(\tau(a))$)

Autrement dit, les fonctions de l'isomorphisme préservent les relations de départ et d'arrivée entre sommets et arêtes

1.4 Degrés

Étant donnés un graphe (S, A, ϵ) et un sommet $s \in S$:

— le ***degré entrant*** $\delta^+(s)$ de s est le nombre d'arêtes dont s est l'arrivée

$$\delta^+(s) = |\{a \in A \mid \tau(a) = s\}|$$

— le ***degré sortant*** $\delta^-(s)$ de s est le nombre d'arêtes dont s est le départ

$$\delta^-(s) = |\{a \in A \mid \sigma(a) = s\}|$$

— le ***degré total*** $\delta(s)$ est la somme des degrés entrant et sortant $\delta^+(s)$ et $\delta^-(s)$.

Note : pour un graphe non orienté, on a une unique notion de degré $\delta(s)$, correspondant au nombre d'extrémités d'arêtes touchant s (une boucle sur s contribue pour 2 à $\delta(s)$, une arête incidente à s qui n'est pas une boucle contribue pour 1).

Exercices

— La somme des degrés des sommets d'un graphe est deux fois son nombre d'arêtes.

— Les isomorphismes préservent les degrés.

1.5 Chemins

Un ***chemin*** de longueur n dans un graphe orienté $G = (S, A, \epsilon)$ est une liste $\rho = (s_0, a_1, s_1, a_2, s_2, \ldots, a_n, s_n)$ alternant $n+1$ sommets et n arêtes, de sorte que chaque arête soit précédée de son sommet de départ et suivie de son sommet d'arrivée :

$$\forall i \in [1, n],\ s_{i-1} = \sigma(a_i) \wedge s_i = \tau(a_i)$$

On peut noter qu'en conséquence, chaque arête part du sommet où la précédente s'arrête :

$$\forall i \in [1, n-1],\ \sigma(a_{i+1}) = \tau(a_i)$$

Le ***départ*** $\sigma(\rho)$ du chemin ρ est le sommet s_0 (départ de la première arête). L'***arrivée*** $\tau(\rho)$ du chemin ρ est le sommet s_n (arrivée de la dernière arête). Un chemin peut n'avoir aucune arête (on parle de ***chemin vide***). Ainsi, pour tout sommet s on a un chemin de longueur 0 défini par la séquence (s), que le graphe soit orienté ou non.

Un ***cycle*** (ou circuit) est un chemin non vide dont le sommet de départ est aussi le sommet d'arrivée. Un ***graphe acyclique*** est un graphe dont aucun chemin n'est un cycle.

Deux chemins $\rho_a = (s_0, a_1, s_1, \ldots, a_n, s_n)$ et $\rho_b = (t_0, b_1, t_1, \ldots, b_m, t_m)$ tels que $s_n = t_0$ peuvent être regroupés en un chemin

$$\rho_a \cdot \rho_b = (s_0, a_1, s_1, \ldots, a_n, s_n, b_1, t_1, \ldots, b_m, t_m)$$

appelé la ***concaténation*** de ρ_1 et ρ_2.

Graphes non orientés Un ***chemin*** de longueur n dans un graphe non orienté est une liste $\rho = (s_0, a_1, s_1, a_2, s_2, \ldots, a_n, s_n)$ alternant sommets et arêtes, de sorte que chaque arête ait pour extrémités les deux sommets l'encadrant dans la séquence :

$$\forall i \in [1, n],\ \epsilon(a_i) = \{s_{i-1}, s_i\}$$

Le ***départ*** d'un tel chemin est le sommet s_0, et son ***arrivée*** le sommet s_n.

Exercices

- — Pour un graphe orienté la séquence des arêtes donne à elle seule une définition non ambiguë du chemin.
- — Pour un graphe non orienté la séquence des arêtes ne suffit pas à donner une définition non ambiguë du chemin.
- — Dans certains cas (lesquels ?) la séquence des sommets donne une définition non ambiguë du chemin.
- — Dans un graphe non orienté, s'il existe un chemin de s à s' alors il existe aussi un chemin de s' à s (mais cela n'est pas nécessairement vrai pour les graphes orientés).

Graphes pondérés Un ***graphe pondéré*** est un graphe (S, A, ϵ) associé à une fonction $p : A \longrightarrow \mathbb{R}$ donnant à chaque arête un ***poids***. Le ***poids d'un chemin*** $\rho = (s_0, a_1, s_1, a_2, s_2, \ldots, a_n, s_n)$ est alors la somme des poids de ses arêtes $\sum_i p(a_i)$.

1.6 Représentations des graphes simples

Il y a plusieurs manières de représenter les graphes.

Matrice d'adjacence (graphes sans arêtes parallèles)
Un graphe G à n sommet $\{s_1, s_2, \ldots, s_n\}$ et sans arêtes parallèles peut être représenté par une matrice M, carrée, d'ordre n et telle que

- — $M[i, j] = 1$ s'il existe une arête de s_i vers s_j, et
- — $M[i, j] = 0$ sinon.

On appelle M la matrice d'adjacence de G.

Si les sommets de G sont représentés par les entiers de 0 à $n-1$, on peut représenter en machine une matrice d'adjacence comme un simple tableau à deux dimensions.

Pour un graphe non orienté on peut utiliser cette même définition, qui donnera une matrice symétrique. Cette représentation peut également être généralisée à un graphe avec arêtes parallèles en définissant $M[i, j]$ comme le nombre d'arêtes de s_i vers s_j.

Liste d'adjacence (graphes sans arêtes parallèles)
Un graphe $G = (S, A, \epsilon)$ peut être représenté par une fonction $\mathrm{adj} : S \longrightarrow \mathscr{P}(S)$ qui à un sommet s associe l'ensemble des sommets cibles d'arêtes partant de s. Autrement dit

$$\mathrm{adj}(s) = \{s' \in S \mid \exists a \in A,\ \epsilon(a) = (s, s')\}$$

Une bonne manière de représenter cette structure en machine consiste à utiliser une table de hachage, qui à chaque nom de sommet associe une liste ou un tableau de sommets voisins. On parle parfois dans ce cas de ***dictionnaire d'adjacence***. Lorsque les sommets s sont les entiers de 0 à $n-1$, on peut aussi se contenter d'un tableau de longueur n, contenant des listes ou tableaux de voisins.

Cette représentation peut être généralisée à un graphe avec arêtes parallèles en gardant dans $\mathrm{adj}(s)$ le nombre d'occurrences de chaque sommet cible.

Une autre variante adaptée aux graphes avec arêtes parallèles consiste à inclure les arêtes dans le résultat, avec une fonction $\mathrm{out} : S \longrightarrow \mathscr{P}(A \times S)$ qui à un sommet s associe l'ensemble des arêtes partant de s et des sommets cibles de ces arêtes. Autrement dit

$$\mathrm{out}(s) = \{(a, s') \in A \times S \mid \epsilon(a) = (s, s')\}$$

Exercices

- Donner des algorithmes permettant de calculer les degrés (entrant, sortant, total) pour chacune de ces représentations.
- Donner des algorithmes permettant de passer d'un représentation à l'autre.
- Écrire des programmes réalisant vos algorithmes.

1.7 Coloration de graphe

Étant donnés un graphe $G = (S, A, \epsilon)$ et un ensemble Σ d'éléments appelés « couleurs », une coloration du graphe G est une application $c : S \longrightarrow \Sigma$ donnant une couleur à chaque sommet de sorte que les sommets adjacents aient des couleurs différentes. Autrement dit :

$$\forall s_1, s_2 \in S, (\exists a \in A, \epsilon(a) = (s_1, s_2)) \Longrightarrow c(s_1) \neq c(s_2)$$

Une question classique consiste, étant donné un graphe, à colorer tous ses sommets avec un nombre minimal de couleurs (alternativement : étant donnés un graphe et un nombre de couleurs disponibles, dire s'il est possible de colorer tous les sommets du graphe).

Cette question modélise des problèmes d'allocation de ressources : étant donnés

- un ensemble de ressources (par exemple des créneaux horaires), et
- un ensemble d'acteurs à qui affecter ces ressources (par exemples des cours)

et sachant que

- certaines paires d'acteurs peuvent se partager une même ressource (par exemple des cours impliquant des ensembles de personnes disjoints, qui pourraient avoir lieu simultanément dans des salles différentes), alors que
- d'autres ne le peuvent pas (par exemple deux cours encadrés par le même enseignant),

on cherche à savoir s'il est possible d'allouer une ressource à chaque acteur (c'est-à-dire de fixer un créneau horaire pour chaque cours).

Lien entre le problème d'origine et la coloration de graphes :

- l'ensemble des acteurs définit un ensemble de sommets,
- les paires d'acteurs ne pouvant se partager une ressource sont reliés par une arête.

Ceci définit un graphe appelé graphe d'interférence, qu'on cherche à colorer avec un ensemble de couleurs défini par l'ensemble des ressources. Ainsi les colorations du graphe correspondent exactement aux allocations valides de ressources.

Théorème classique : tout graphe planaire peut être coloré avec 4 couleurs.

Exercices

- K_5 ne peut pas être coloré avec 4 couleurs.
- Un graphe bipartite peut être coloré avec 2 couleurs.
- Il faut parfois 3 couleurs pour colorer un cycle.
- Un graphe non orienté dont tous les sommets ont un degré $\delta \leq \Delta$ peut être coloré avec $\Delta + 1$ couleurs (indice : récurrence sur le nombre de sommets) (remarque : de la preuve par récurrence on peut déduire un algorithme)

1.8 Tri topologique

Étant donné un graphe orienté $G = (S, A, \epsilon)$ à N sommets, un tri topologique de G consiste à former avec l'ensemble des sommets une séquence telle que s'il existe une arête de s_1 vers s_2, alors s_1 apparaît avant s_2 dans la séquence. Autrement dit, on cherche une application $\varphi : S \longrightarrow [0, N[$ associant à chaque sommet un entier, qui soit bijective et telle que

$$\forall a, \in A. \varphi(\sigma(a)) < \varphi(\tau(a))$$

Cette question modélise des problèmes d'ordonnancement : étant donnés

— un ensemble de tâches à réaliser (par exemple des concepts à présenter) et

— des contraintes imposant que certaines tâches soient réalisée avant certaines autres (par exemple, un concept ne peut être introduit qu'une fois que tous les autres dont il dépend ont déjà été traités),

on cherche à définir un ordre respectant toutes les contraintes.

Lien entre le problème d'origine et le tri topologique :

— l'ensemble des tâches à ordonner définit un ensemble de sommets,

— si une tâche T_1 doit nécessairement être traitée avant une tâche T_2, alors on place une arête de T_1 vers T_2.

Ceci définit un graphe appelé graphe de dépendance, dont on cherche à faire un tri topologique.

Remarque sur ce problème : le sommet recevant le plus petit numéro ne peut pas être la cible d'une arête.

Tentative d'algorithme :

— prendre un sommet de degré entrant 0, lui donner le plus petit numéro pas encore affecté, et le retirer du graphe,

— continuer jusqu'à avoir numéroté tous les sommets.

Réalisation en Python, le graphe étant donné sous la forme d'un dictionnaire d'adjacence.

```
def choisit_sommet_libre(G):
  for s in G:
    if len(G[s]) == 0:
      return s

def supprime_sommet(s, G):
  del G[s]
  for x in G:
    if s in G[x]:
      G[x] = G[x] - {s}

def tri_topologique(G):
  ordre = []
  G = [ G[s] for s in G ]
  while len(G) != 0:
    sommet_courant = choisit_sommet_libre(G)
    supprime_sommet(G, sommet_courant)
    ordre.append(sommet_courant)
  return ordre
```

Mais...

— y a-t-il nécessairement un sommet de degré 0 dans le graphe d'origine ? [Non]

— dans le cas où il y a un sommet de degré 0 dans le graphe d'origine, en trouve-t-on encore à coup sûr à chaque étape suivante ? [Non]

— en observant la forme d'un graphe, peut-on savoir à l'avance si cet algorithme fonctionnera ? [Oui, c'est la partie suivante]

1.9 Acyclicité et tri topologique

Théorème Un graphe orienté acyclique admet un tri topologique.

On démontre au préalable un lemme : tout graphe acyclique non vide admet un sommet de degré entrant 0.
Preuve par l'absurde. Soit G un graphe acyclique non vide tel que tous les sommets de G ont un degré entrant strictement positif. On va démontrer par récurrence que $\forall n \in \mathbb{N}$, G admet un chemin de longueur n :

— *Cas de base :* G étant non vide il contient un sommet s, et un chemin de longueur 0 de s à s.
— *Itération :* Soit n tel que G admette un chemin de longueur n, et soit ρ un tel chemin de longueur n dans G. Le sommet de départ s de ρ a un degré entrant non nul, il existe donc une arête a ayant s pour arrivée. En notant s' le départ de a on en déduit un chemin (s', a, ρ) de longueur $n+1$ dans G.

Donc $\forall n \in \mathbb{N}$, G admet un chemin de longueur n.

En particulier, en notant N le nombre de sommets de G, G admet un chemin ρ de longueur $N+1$. Par le principe des tiroirs il existe un sommet s de G tel que ρ passe deux fois par s. On extrait de ρ la séquence comprise entre les premières et deuxième occurrences de s pour obtenir un chemin de s à s, c'est-à-dire un cycle. Contradiction.

Donc G admet un sommet de degré entrant 0.

Retour au théorème : Tout graphe orienté acyclique admet un tri topologique.
Preuve par récurrence sur le nombre de sommets du graphe.
On note $P(n)$ la propriété : tous les graphes acycliques à n sommets admettent un tri topologique.

— Preuve de $P(0)$: un graphe vide est trié topologiquement par la séquence vide.
— Preuve de $\forall n \in \mathbb{N}, P(n) \Longrightarrow P(n+1)$: Soit n tel que $P(n)$ soit vraie, et soit G un graphe acyclique à $n+1$ sommets.
Par notre lemme, G admet un sommet s de degré entrant 0. Le sous-graphe $G' = (S \setminus \{s\}, A', \epsilon')$ obtenu en retirant de G ce sommet s et ses arêtes incidentes a n sommets, et est de plus acyclique (supposons qu'il existe un cycle dans G', alors ce cycle existerait également dans G ; or G est acyclique : contradiction).
On peut donc appliquer l'hypothèse de récurrence à G' pour obtenir un tri topologique de G', c'est-à-dire une bijection $\varphi' : S \setminus \{s\} \longrightarrow [0, n[$ telle que

$$\forall a \in A',\ \varphi'(\sigma(a)) < \varphi'(\tau(a))$$

On définit $\varphi : S \longrightarrow [0, n+1[$ par :
— $\varphi(s) = 0$
— $\forall s' \in S \setminus \{s\},\ \varphi(s') = \varphi'(s') + 1$
(cela revient à placer s en tête de la séquence obtenue pour G').
Montrons que l'application φ est un tri topologique de G :
— φ est une bijection car il s'agit d'une application surjective entre deux ensembles de même cardinal (justification de la surjectivité : soit $k \in [0, n+1[$, si $k = 0$ alors $\varphi(s) = k$, et sinon $k - 1 \in [0, n[$, d'où par surjectivité de φ' il existe s' tel que $\varphi'(s') = k - 1$

et $\varphi(s) = \varphi'(s') + 1 = k$) (si on voulait aussi prouver l'injectivité à la main : soient s_1, s_2 deux sommets distincts, si $s_1 = s$ alors $\varphi(s_1) = 0$ et $\varphi(s_2) = \varphi'(s_2) + 1 (> 0)$ donc $\varphi(s_1) \neq \varphi(s_2)$, si $s_2 = s$ on conclut de même, et si $s_1 \neq s$ et $s_2 \neq s$ alors $\varphi(s_1) = \varphi'(s_1) + 1$ et $\varphi(s_2) = \varphi'(s_2) + 1$; si $\varphi(s_1) = \varphi(s_2)$ on aurait alors $\varphi'(s_1) = \varphi'(s_2)$, ce qui est impossible par injectivité de φ').

— φ respecte les contraintes d'ordre : soit $a \in A$
  — si $a \in A'$, alors par hypothèse $\varphi'(\sigma(a)) < \varphi'(\tau(a))$, avec de plus σ (a) et τ (a) différents de s ; donc $\varphi(\sigma(a)) = \varphi'(\sigma(a)) + 1$, $\varphi(\tau(a)) = \varphi'(\tau(a)) + 1$ et comme $\varphi'(\sigma(a)) + 1 < \varphi'(\tau(a)) + 1$ on conclut $\varphi(\sigma(a)) < \varphi(\tau(a))$.
  — sinon, au moins l'une des extrémités de a est s ; or s a un degré entrant 0, donc $\tau(a) \neq s$ et par conséquent $\sigma(a) = s$; on a donc $\varphi(\sigma(a)) = \varphi(s) = 0$ et $\varphi(\tau(a)) = \varphi'(\tau(a)) + 1 > 0$, d'où $\varphi(\sigma(a)) < \varphi(\tau(a))$.

2 Chemins dans un graphe et connexité (30/01)

2.1 Connexité

Un graphe est dit ***fortement connexe*** si toute paire de sommets est reliée par au moins un chemin :

$$\forall s, s' \in S,\ \exists \rho \in \text{chemins}(G),\ \sigma(\rho) = s \wedge \tau(\rho) = s'$$

Note : cette définition vaut à la fois pour les graphes orientés et non orientés.

Un graphe orienté est dit ***connexe*** si le graphe non orienté sous-jacent est fortement connexe (et un graphe non orienté fortement connexe est aussi simplement appelé un graphe connexe).

Définition alternative : un graphe est dit fortement connexe si tous deux sommets sont reliés par au moins un chemin dans chaque sens :

$$\forall s, s' \in S,\ (\exists \rho \in \text{chemins}(G),\ \sigma(\rho) = s \wedge \tau(\rho) = s') \wedge (\exists \rho' \in \text{chemins}(G),\ \sigma(\rho') = s' \wedge \tau(\rho') = s)$$

Exercices

— Donner un graphe connexe mais pas fortement connexe.
— Les deux définitions de forte connexité sont équivalentes (même pour des graphes orientés).
— Un graphe admettant un cycle hamiltonien est fortement connexe.
— Un cycle eulérien dans un graphe connexe passe au moins une fois par chaque sommet.

Distances On se place dans le cas d'un graphe non orienté.

La ***distance*** $\text{dist}(s, s')$ entre deux sommets s et s' est la longueur minimale d'un chemin entre s et s'. Autrement dit :

— s'il existe au moins un chemin entre s et s', alors il existe un chemin ρ dont la longueur est inférieure ou égale aux longueurs de tous les autres chemins entre s et s' (exercice : le justifier), et la distance entre s et s' est la longueur de ρ,
— s'il n'existe pas de chemin entre s et s', alors la distance entre s et s' est l'infini.

Dans un graphe connexe, les distances sont donc toujours finies.

Le ***diamètre*** d'un graphe est la distance maximale entre deux sommets du graphe. Autrement dit :

— si le graphe n'est pas connexe alors il existe deux sommets à distance infinie et le diamètre est infini,
— si le graphe est connexe alors il existe deux sommets s, s' avec $\text{dist}(s,s')$ supérieure ou égale aux distances entre toutes les autres paires de sommets, et le diamètre du graphe est $\text{dist}(s,s')$.

Exercices
— Donner les diamètres des graphes K_n, $K_{n,m}$, L_n, C_n, $G_{n,m}$ et H_n cités dans le chapitre précédent.
— Donner un cas dans lequel les distances $\text{dist}(s,s')$ et $\text{dist}(s',s)$ ne sont pas égales.

2.2 Composantes connexes

La ***composante fortement connexe*** d'un sommet s d'un graphe G est l'ensemble des sommets s' de G tels qu'il existe un chemin de s vers s' et un chemin de s' vers s.

Les composantes fortement connexes d'un graphes sont les composantes fortement connexes de ses sommets. Ces composantes définissent une partition des sommets du graphe (voir prochain chapitre). En conséquence, si un sommet s' appartient à la composante fortement connexe d'un sommet s, alors ces deux sommets ont la même composante fortement connexe.

Caractérisation alternative Une ***composante fortement connexe*** d'un graphe G est un sous-graphe fortement connexe maximal de G.

Précisons qu'on parle ici de maximalité vis-à-vis de l'inclusion des sommets. Ainsi un sous-graphe $G' = (S', A', \epsilon')$ fortement connexe de G est une composante fortement connexe de G si aucun sous-graphe $G'' = (S'', A'', \epsilon'')$ de G tel que $S' \subset S''$ (et $S' \neq S''$) n'est fortement connexe.

Exercices
— Deux composantes fortement connexes d'un graphe sont soit disjointes soit égales.

2.3 Produit matriciel et chemins

Considérons un graphe G à N sommets représenté par sa matrice d'adjacence M. La matrice M^n dénombre les chemins de longueur n entre chaque paire de sommets (fonctionne aussi pour la généralisation de la matrice d'adjence pour des graphes avec arêtes parallèles).

Preuve par récurrence sur n. On note $P(n)$ la propriété : « pour tous i,j dans $[1,N]$, la valeur $M^n[i,j]$ est égale au nombre de chemins de i à j dans G.

— Cas de base : M^0 est la matrice identité, or pour tout i on a bien un unique chemin de longueur 0 de i à i (le chemin vide), et pour tous $i \neq j$ il n'existe pas de chemin de longueur 0 de i à j.
— Itération : soit n tel que $P(n)$, soient i,j dans $[0,N[$. Par définition

$$M^{n+1}[i,j] = M^n[i,1]M[1,j] + M^n[i,2]M[2,j] + \ldots + M^n[i,N]M[N,j]$$

Par hypothèse, pour tout $k \in [1,N]$ la valeur $M[i,k]$ est égale au nombre de chemins de longueur n de i à k ; chaque combinaison de l'un de ces chemins avec une arête de k à j donne un chemin de longueur $n+1$ de i à j (et tous les chemins ainsi obtenus sont

différents), on a donc $M^n[i,k]M[k,j]$ chemins de longueur $n+1$ de i à j pour lesquels l'avant dernier sommet visité est k.
Comme tout chemin de i à j de longueur $n+1$ admet un avant-dernier sommet défini de manière unique (cela fonctionne même pour $n = 0$, où l'avant-dernier sommet est aussi le premier sommet), le nombre total de chemins de i à j de longueur $n+1$ est bien donné par la somme $M^n[i,1]M[1,j] + \ldots + M^n[i,N]M[N,j]$.

Par principe de récurrence, la propriété est justifiée pour tout $n \in \mathbb{N}$.

Exercices

— Donner une expression dénombrant le nombre de chemins de i à j de longueur inférieure ou égale à n.

— En s'inspirant du résultat de cette section, proposer une technique pour déterminer entre quelles paires de sommet il existe un chemin (sans contraintes sur les longueurs).

2.4 Calcul d'accessibilité : algorithme de Roy-Warshall

Prenons la matrice d'adjacence M d'un graphe G à n sommets, où les sommets sont les nombres de l'intervalle $[1,n]$. On peut considérer que cette matrice contient des booléens en interprétant 0 comme Faux et 1 comme Vrai (et dans la version généralisée en interprétant tout nombre non nul comme Vrai).

On cherche à obtenir une matrice d'accessibilité pour G, c'est-à-dire une matrice C carrée d'ordre n telle que $C[i,j]$ vaut Vrai si et seulement s'il existe un chemin de i à j dans le graphe G (pour simplifier, on ne s'intéressera qu'aux chemins de longueur au moins 1).

L'***algorithme de Roy-Warshall*** répond à ce problème en calculant une séquence de matrices C_k pour k entre 0 et n, de sorte que $C_k[i,j]$ vaut Vrai si et seulement s'il existe dans le graphe G un chemin de i à j n'utilisant que des sommets de l'intervalle $[1,k]$ (hormis éventuellement i et j sur lesquels on n'a pas de contraintes).

Premières remarques :

— C_0 est égale à la matrice d'adjacence M. En effet, un chemin de i à j qui n'utilise que des sommets de l'intervalle $[1,0]$ (qui est l'intervalle vide) est un chemin sans sommets intermédiaires, c'est-à-dire une arête de i à j.

— C_n est la matrice C cherchée. En effet, un chemin de i à j qui peut utiliser tout sommet de l'intervalle $[1,n]$ est un chemin arbitraire de G.

Le cœur de l'algorithme (le calcul des matrices C_k successives) est ensuite basé sur cette unique réflexion : un chemin de i à j n'utilisant que des sommets de l'intervalle $[1,k+1]$ soit ne contient pas le sommet $k+1$ (et donc ne contient que des sommets de l'intervalle $[1,k]$), soit contient le sommet $k+1$. Dans ce dernier cas, on peut justifier qu'il existe un chemin de i à j n'utilisant que des sommets de l'intervalle $[1,k+1]$ qui est élémentaire (reprendre la preuve selon laquelle s'il existe un chemin ρ entre deux sommets s_1, s_2 alors il existe un chemin élémentaire ρ' entre s_1 et s_2 et montrer qu'il est de plus toujours possible de choisir le chemin ρ' de sorte qu'il ne passe que par des sommets par lesquels passait déjà ρ), et donc qui ne contient qu'une fois le sommet $k+1$. Ce chemin est donc la concaténation d'un chemin de i à $k+1$ et d'un chemin de $k+1$ à j, ces deux chemins prenant leurs sommets intermédiaires dans l'intervalle $[1,k]$.

On en déduit la formule :

$$C_{k+1}[i,j] \equiv C_k[i,j] \vee (C_k[i,k+1] \wedge C_k[k+1,j])$$

Il suffit ensuite, partant de $C_0 = M$, de calculer successivement les C_k.

Réalistion en Python

```
def roy_warshall(M):
  n = len(M)
  # C est un tableau de n+1 matrices carrées d'ordre n
  C = [[[ 0 for j in range(n) ] for i in range(n) ] for k in range(n+1) ]
  # C_0 est egale a M
  C[0] = M
  for k in range(n):
    # Définition de C_{k+1} en fonction de C_k
    for i in range(n):
      for j in range(n):
        # Pour chaque case on applique la formule
        C[k+1][i][j] = C[k][i][j] | (C[k][i][k] & C[k][k][j])
  # Le résultat est C_n
  return C[n]
```

Exercice

— Plutôt que de définir $n+1$ matrices successives, on propose de modifier en place la matrice d'origine, ce qu'on peut réaliser par exemple avec le code suivant :

```
def roy_warshall(M):
  n = len(M)
  for k in range(n):
    # On modifie n fois la matrice M
    for i in range(n):
      for j in range(n):
        # Pour chaque case on applique la formule
        M[i][j] = M[i][j] | (M[i][k] & M[k][j])
  return M
```

En quoi cet algorithme est-il différent ? Peut-on justifier que c'est encore correct ?

2.5 Calcul des plus courts chemins : algorithme de Roy-Floyd-Wharshall

L'algorithme de Roy-Warshall peut être adapté pour ne pas seulement indiquer quels sommets peuvent être reliés par un chemin, mais préciser également la longueur du chemin le plus court permettant de les relier.

Considérons un graphe G à n sommets, où les sommets sont les nombres de l'intervalle $[1,n]$. On représente ce graphe par une variante M de la matrice d'adjacence dans laquelle $M[i,j]$ vaut 1 s'il existe un arc du sommet i au sommet j, et ∞ sinon. On construit alors comme dans l'algorithme de Roy-Warshall une séquence C_k de matrices, qui sont cette fois telles que $C_k[i,j]$ contient la longueur du plus court chemin de i à j n'empruntant que des sommets intermédiaires de l'intervalle $[1,k]$ s'il en existe un, et ∞ s'il n'existe pas de tel chemin.

Le même raisonnement aboutit à la formule

$$C_{k+1}[i,j] \equiv \min(C_k[i,j],\ C_k[i,k+1] + C_k[k+1,j])$$

Note : l'algorithme de ***Roy-Floyd-Warshall*** proprement dit applique cette même formule à un graphe pondéré pour calculer le poids minimum d'un chemin entre les deux sommets. Cette version complète fonctionne tant qu'il n'existe pas de cycle de poids négatif (dans ce cas la notion de chemin le plus court elle-même peut ne plus avoir de sens).

2.6 Chemins particuliers

Un ***chemin simple*** est un chemin ne comportant pas deux fois la même arête. Un ***chemin élémentaire*** est un chemin ne comportant pas deux fois le même sommet.

Un ***chemin eulérien*** est un chemin simple passant par toutes les arêtes (c'est-à-dire un chemin comportant exactement une fois par chaque arête). Un ***cycle eulérien*** est un chemin eulérien qui est aussi un cycle.

Un ***chemin hamiltonien*** est un chemin élémentaire passant par tous les sommets (c'est-à-dire un chemin comportant exactement une fois par chaque sommet). Un ***cycle hamiltonien*** est un cycle comportant exactement une fois par chaque sommet, excepté sont sommet de départ qui est présent deux fois (au départ et à l'arrivée).

Exercices

- S'il existe un chemin entre deux sommets s et s', alors il existe un chemin sans boucles entre s et s'.
- S'il existe un chemin entre deux sommets s et s', alors il existe un chemin élémentaire entre s et s'.
- Un chemin hamiltonien n'est pas forcément eulérien.
- Un chemin eulérien n'est pas forcément hamiltonien.
- Un graphe possédant un cycle eulérien n'a que des sommets dont le degré entrant est égal au degré sortant (ou un degré pair dans le cas d'un graphe non orienté).

2.7 Un critère d'euléricité

Tout graphe non orienté connexe dont les sommets ont des degrés pairs admet un cycle eulérien.

Démonstration par récurrence forte sur le nombre d'arêtes du graphe. On note $P(n)$ la propriété « tout graphe non orienté connexe à n arêtes et dont les sommets ont des degrés pairs admet un cycle eulérien ».

- $P(0)$: un graphe à 0 arêtes comporte un cycle eulérien (le chemin vide).
- Soit $n \in \mathbb{N}$, $n > 0$ tel que pour tout $k < n$, $P(k)$ soit vraie, et soit $G = (S, A, \epsilon)$ un graphe non orienté connexe à n arêtes dont les sommets ont des degrés pairs.
  - Si G contient une boucle a autour d'un sommet s, alors le graphe $G' = (S, A \setminus \{a\}, \epsilon')$ défini en retirant la boucle a à G est un graphe non orienté à $n-1$ arêtes tel que :
    - G' est connexe (soit une paire s_1, s_2 de sommets de G', alors par connexité de G il existe un chemin ρ dans G entre s_1 et s_2, donc par une propriété énoncée plus haut il existe un chemin sans boucle ρ' dans G entre s_1 et s_2, et comme ρ' est sans boucle en particulier il n'utilise pas a et existe donc encore dans G'), et
    - les sommets de G' ont des degrés pairs (soit un sommet s' de G', si $s \neq s'$ alors s' a le même degré dans G' et dans G et ce degré est donc pair, et si $s = s'$ alors le degré de s' dans

G' est inférieur de 2 à son degré dans G, ce qui préserve sa parité).

L'hypothèse de récurrence $P(n-1)$ s'applique donc à G' et assure qu'il existe un cycle eulérien ρ' dans G'. Or tout cycle eulérien dans un graphe connexe passe au moins une fois par chaque sommet, donc ρ' passe par s et on peut obtenir un cycle eulérien ρ de G en ajoutant la boucle a dans ρ' après la première occurrence de s.

— Sinon, G possède une arête a_1 entre deux sommets s_0 et s_1 avec $s_0 \neq s_1$. Comme $\delta(s_1)$ est pair et $\delta(s_1) \geq 1$, on déduit que $\delta(s_1) \geq 2$ et donc qu'il existe aussi une arête $a_2 \neq a_1$ incidente à s_1. On note s_2 l'autre extrémité de a_2 (par absence de boucles on a $s_2 \neq s_1$, mais on n'a pas nécessairement $s_2 \neq s_0$). Soit $G' = (S, A \setminus \{a_1, a_2\} \cup \{a\}, \epsilon')$ obtenu en retirant à G les arêtes a_1 et a_2 et en ajoutant une nouvelle arête a entre s_0 et s_2 (cette arête pouvant être une boucle). Alors :
  — G' est un graphe non orienté à $n-1$ arêtes dont les sommets ont des degrés pairs (soit un sommet s' de G', si $s \notin \{s_0, s_1, s_2\}$ alors ce sommet est incident aux mêmes arêtes dans G et G' et son degré est donc inchangé et pair, sinon, si $s = s_1$ alors s a dans G' deux arêtes incidentes en moins par rapport à sa situation dans G et la parité du degré est donc inchangée, sinon (et dans ce cas $s \in \{s_0, s_2\}$), soit $s_0 \neq s_2$ et s a le même nombre d'arêtes incidentes (on a retiré a_1 ou a_2 et on a ajouté a) soit $s_0 = s_2$ et s a deux arêtes incidentes en moins (a_1 et a_2) mais une boucle en plus (a), dans les deux derniers cas le degré est inchangé et la parité du degré est donc encore préservée), et
  — G' n'est pas nécessairement connexe car il n'y a pas forcément de lien entre les composantes connexes de s_0 et de s_1 dans G'. En revanche, on peut quand même s'assurer que soit G' est connexe, soit G' est constitué d'exactement deux composantes connexes (celle de s_0 et celle de s_1). On conclut alors avec un raisonnement par cas :
    — Si G' est connexe alors par hypothèse de récurrence $P(n-1)$, G' admet un cycle eulérien ρ'. Le cycle ρ' étant eulérien il passe exactement une fois par chaque arête, et on peut remplacer l'unique occurrence de a dans ρ' par la séquence (a_1, s_1, a_2) ou la séquence (a_2, s_1, a_1) (selon le sens dans lequel l'arête a est empruntée) pour obtenir un cycle eulérien de G.
    — Si G' est formé de deux composantes connexes C_0 (celle de s_0) et C_1 (celle de s_1) alors chacune de ces composantes est un graphe non orienté comportant au plus $n-1$ arêtes et dont les sommets ont des degrés pairs. On peut donc appliquer l'hypothèse de récurrence pour obtenir ρ_0 un cycle eulérien de C_0 et ρ_1 un cycle eulérien de C_1. On combine alors $\rho_0 \setminus \{a\}$, ρ_1, a_1 et a_2 pour créer un cycle eulérien de G (à partir de là, la meilleure conclusion se fait par un dessin).