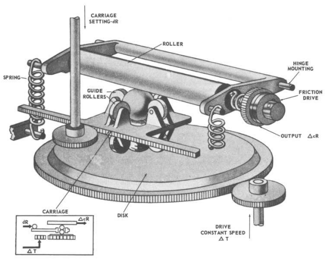
Fondements de l'informatique



Olivier Bournez
bournez@lix.polytechnique.fr
Ecole Polytechnique

INF412

CSC_INF41012_EP

Plan du cours :

- Logique : 1,2,3,4
  - Calcul propositionnel.
  - Calcul des prédicats.
- Modèles : 4,5,6,7
  - Machines de Turing.
  - Thèse de Church-Turing.
  - Calculabilité.
- Calculs : 7,8,9,10
  - Complexité en temps :
    - la question $\mathrm{P} = \mathrm{NP}$?
    - NP-complétude.
  - Complexité en espace.
  - Gérer la complexité.

Organisation du cours

- 10 blocs, soit 10 mardis/mercredis.
  - Le mardi après midi, amphi Cauchy, de 15h15 à 16h45.
  - Le mercredi matin, PC, de 8h00 à 10h00 ou de 10h15 à 12h15.

Samuel Mimram,
Gr1-3,
PC 42

Noam Zeilberger,
Gr2-4,
PC 44

- Page du cours SUR MOODLE

  https://moodle.polytechnique.fr/course/view.php?id=19276

  et vos questions à `Olivier.Bournez@polytechnique.fr` et/ou aux enseignants de l'équipe.
- Évaluation.
  - Une PC notée : la PC 6, **le 2 Octobre 2024**.
  - Contrôle à la fin.
  - Note de module : $\max(CC, \frac{3}{4}CC + \frac{1}{4}PC)$.

Délégués

Les noms de deux délégué(e)s ?

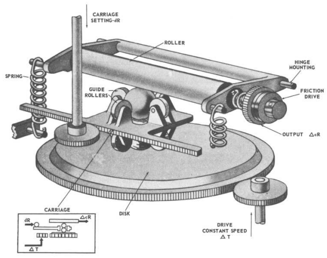
Exprimez vous.

Page du cours.

Commentaires, avis
sur les cours et les PCs.

- Page du cours:
  https://moodle.polytechnique.fr/course/view.php?id=19276.
- Commentaires, avis sur les cours et les PCs.
  www.enseignement.polytechnique.fr/informatique/INF412/AVIS.

Cours 1 : Motivation.
Récursivité et induction.
Calcul Propositionnel



Olivier Bournez
bournez@lix.polytechnique.fr INF412 Ecole Polytechnique
CSC_INF41012_EP

Fondements des mathématiques

Cantor

Russel

Hilbert

- Fin du 19ème siècle :
  - Théorie des ensembles de Cantor.
- Début du 20ème siècle :
  - Des paradoxes apparaissent :
    - Russel : $y = \{x | x \not\in x\}$ est-il un ensemble ?
  - Hilbert propose de **fonder les mathématiques** : c'est-à-dire, de proposer un langage et des axiomes permettant d'exprimer toutes les mathématiques, avec
    - **Complétude** : une preuve que tout énoncé mathématique vrai peut se prouver dans le formalisme ;
    - **Cohérence** : une preuve qu'aucune contradiction ne peut être obtenue ;
    - **Décidabilité** : décider si un énoncé est vrai ou faux doit s'obtenir par une certaine **méthode (algorithme)**.

Fondements des mathématiques
Fondements de l'informatique

Gödel

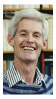
Church

Turing

- 1931 : Gödel prouve **les théorèmes d'incomplétude**.
  - toute théorie cohérente suffisamment expressive pour parler des entiers et de la multiplication des entiers ne peut pas prouver sa propre cohérence.
- Années 1930 :
  - On cherche alors à formaliser la notion **d'algorithme.**
    - Fonctions récursives : Gödel.
    - Lambda-calcul : Church, 1936.
    - Machines de Turing : Turing, 1936.
  - **Thèse de Church-Turing :** tous ces modèles sont équivalents.

Fondements de l'informatique


Turing


Cook


Levin

- Depuis 1936 : Théorie de la **calculabilité.**
  - Plusieurs problèmes naturels s'avèrent ne pas être solvables. Par exemple :
    - vérification, 10ème problème de Hilbert, simplification en calcul formel, ...
- 1970 : Théorie de la **complexité** (moderne).
  - Notion de NP-complétude.
  - Question P = NP ?
- 21ème siècle :
  - La question P = NP ? est parmi les 4-questions pour le millénaire.
  - Mise à prix à 1.000.000 de dollars.
  - Tous les modèles restent d'actualité, et sont à la base de nos modèles informatiques actuels.

- Un cours sur trois domaines centraux en informatique :
  1. **la logique,**
  2. **les modèles de calculs,**
  3. **la complexité**,
- reliés par la question suivante :

  Quelles sont les capacités et les limites des ordinateurs ?

Question principale du cours

Question principale : **Comprendre qu'est-ce qui rend certains problèmes difficiles, et d'autres faciles ?**

- Comprendre pourquoi
  - certains problèmes se résolvent très vite :

    exemple : un téléphone portable moderne peut trier un répertoire de plus d'un million d'entrées en quelques minutes.
  - d'autres sont difficiles à résoudre :

    exemple : résoudre un problème d'emploi du temps peut prendre des siècles à résoudre avec seulement un millier de données en entrées, même avec les ordinateurs les plus puissants actuels.

Pourquoi s'intéresser aux problèmes difficiles ?

Quelques histoires récentes :

- 370 millions de dollars :

- $\geq$ 475 millions de dollars :

  $\frac{4195835}{3145727} =$ 1.333739068902037589

- Parce que des problèmes très simples et concrets, et aux enjeux économiques considérables, en font partie.
- Pour comprendre comment simplifier ou modifier les problèmes pour qu'ils puissent être résolus.
- Pour concevoir des systèmes :
  - en évitant les difficultés :
    - conception, vérification, ...
  - en utilisant les difficultés :
    - cryptographie, ...

Introduction

- La **logique propositionnelle** permet essentiellement de discuter des connecteurs grammaticaux comme la négation ($\neg$), la conjonction ($\wedge$) et la disjonction ($\vee$), en composant des propositions à partir de propositions données.

p	q	$\neg p$	$p \vee q$	$p \wedge q$	$p \Rightarrow q$	$p \Leftrightarrow q$
0	0	1	0	0	1	1
0	1	1	1	0	1	0
1	0	0	1	0	0	0
1	1	0	1	1	1	1

- Elle permet essentiellement de parler de **fonctions booléennes**, c'est-à-dire de fonctions de $\{0,1\}^n \to \{0,1\}$. En effet, les variables, c'est-à-dire **les propositions**, ne peuvent prendre que deux valeurs, **vrai** (codé par 1) ou **faux** (codé par 0).

Syntaxe	Sémantique
Expressions arithmétiques : ▸ $((1+2)*3)$: ok. ▸ $+4($: pas ok.	▸ 9
Formules propositionnelles : ▸ $((p \vee q) \wedge r)$: ok. ▸ $((q \wedge p) \Rightarrow q)$: ok. ▸ $\vee \Rightarrow \wedge p$: pas ok.	▸ Une fonction de $\{0,1\}^3$ dans $\{0,1\}$: vaut par exemple 1 pour $(p,q,r) = (0,1,1)$. ▸ Une fonction de $\{0,1\}^2$ dans $\{0,1\}$

- Il est utile[1] de lire **le polycopié**.
  - en particulier, le chapitre 1 rappelle les notions d'alphabet, mot, langage, graphe, arbres, ...,
  - qui doivent/devraient être connues.

1. voire nécessaire

Exemple : mots

- Un **alphabet** Σ est un ensemble fini, dont les éléments sont appelés des **lettres**.
- Un **mot** w sur Σ est une suite finie d'éléments de Σ.
  - w s'écrit $w = a_1 a_2 \cdots a_n$, avec $a_i \in \Sigma$.
  - $n \geq 0$ est **la longueur du mot**.
  - Le mot vide, noté ϵ, est (l'unique) mot de longueur 0.
- L'ensemble des mots sur Σ est noté Σ^*
- Un **langage** sur Σ est un ensemble de mots sur Σ (une partie de Σ^*).

Quelques opérations sur les mots

- Etant donnés deux mots m_1 et m_2, la concaténation de m_1 et de m_2, notée $m_1.m_2$, ou notée $m_1 m_2$, est le mot obtenu en mettant m_2 à la fin de m_1.
  - Exemple : $aba.ba$ est le mot $ababa$ sur $\Sigma = \{a, b\}$.
- La concaténation est une opération associative, avec un élément neutre, le mot vide ϵ.

Exemples de langages

- $\Sigma_{bin} = \{0, 1\}$:
  - $101101 \in \Sigma_{bin}^*$;
- $\Sigma_{nombre} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$:
  - $1794 \in \Sigma_{nombre}^*$, $007 \in \Sigma_{nombre}^*$ et $101101 \in \Sigma_{nombre}^*$;
  - $\mathcal{N}$ l'ensemble des nombres entiers non-nuls écrits en base 10 ;
    - $1794 \in \mathcal{N}$, $101101 \in \mathcal{N}$, $007 \notin \mathcal{N}$.
- $\Sigma_{latin} = \{a, b, \cdots, z, A, B, \cdots, Z\}$:
  - vocalise $\in \Sigma_{latin}^*$ et vczeerrztx $\in \Sigma_{latin}^*$;
  - *Dico* le langage des mots d'un dictionnaire du français ;
    - *haricot* $\in$ *Dico*, *sphf* $\notin$ *Dico*.
- $\Sigma_{unicode}$ défini comme l'ensemble des caractères Unicode
  - génial, c'est la rentrée $\in \Sigma_{unicode}^*$:
  - *Francais* le langage des phrases valides du français.

- $\Sigma_{exp} = \{0, 1, 2, \cdots, 9, +, -, *, /, (,)\}$ l'alphabet des expressions arithmétiques :
  - ((1+3)*2) $\in \Sigma_{exp}^*$ et ((+)+/3) $\in \Sigma_{exp}^*$.
  - *Arith* le langage des expressions arithmétiques.
    - ((1+3)*2) $\in$ *Arith*, +/3- $\notin$ *Arith*.
- Fixons un ensemble $\mathscr{P}$ de variables.
  $\Sigma_{prop} = \mathscr{P} \cup \{0, 1, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ est l'alphabet du calcul propositionnel sur $\mathscr{P}$: par exemple, pour $\mathscr{P} = \{p, q, r\}$:
  - $((r \vee p) \wedge q) \in \Sigma_{prop}^*$ et $p \vee q \wedge (\in \Sigma_{prop}^*$.
  - *Prop* le langage des formules propositionnelles.
    - $((r \vee p) \wedge q) \in$ *Prop*, $\vee p \wedge \notin$ *Prop*.
- Comment définir *Arith* et *Prop* :
  - récursivement/inductivement ! ! !

L'induction

- **l'induction** est un outil formel élégant, simple et très utile en informatique :
  - qui permet de faire des définitions récursives ;
  - et qui permet une technique de preuve qui généralise la preuve par récurrence.

Trois exemples

- En JAVA : dans

```
class Liste {
  int contenu;
  Liste suivant;
  }
Liste lst;
```

  on définit la classe `Liste` en utilisant dans la définition le champ "`suivant`" dont le type est la classe `Liste`.

- L'ensemble **des entiers pairs** est le plus petit [2] ensemble qui contient 0 et tel que si n est pair, alors $n+2$ est pair.

- L'ensemble *Arith* des **expressions arithmétiques** est le plus petit ensemble qui contient les entiers, et tel que si g et d sont des expressions arithmétiques, alors $g+d$, $g-d$, $g*d$, g/d et (g) sont des expressions arithmétiques.

2. "Plus petit" : pour l'inclusion dans tout ce qui suit.

Définition inductives

- Une **définition inductive** d'une partie X d'un ensemble consiste à se donner :
  - la donnée explicite de certains éléments de X (**base**) ;
  - la donnée de règles de construction de nouveaux éléments de X à partir d'éléments déjà construits (**étapes inductives**).

- Exemple. Une définition inductive des entiers pairs :
  (B) $0 \in P$;
  (I) $n \in P \Rightarrow n+2 \in P$.

- Exemple. L'ensemble $\mathcal{N}$ des nombres entiers non-nuls écrits en base 10 est la partie de $\Sigma_{exp}{}^*$, définie inductivement par
  (B) $a \in \mathcal{N}$ pour chaque $a \in \{1,2,\ldots,9\}$;
  (I) $g \in \mathcal{N} \Rightarrow ga \in \mathcal{N}$, pour chaque $a \in \{0,1,2,\ldots,9\}$.

Théorème du point fixe

- Une **définition inductive** d'une partie X d'un ensemble E consiste à se donner :
  - un sous ensemble non vide B de E ;
  - et un ensemble de **règles** R : chaque règle $r_i \in R$ est une fonction (qui peut être partielle) r_i de $E^{n_i} \to E$ pour un certain entier n_i.

Théorème
A une définition inductive correspond un plus petit ensemble X qui vérifie les propriétés suivantes :
(B) *il contient B : $B \subseteq X$;*
(I) *il est stable par les règles de R : pour chaque règle $r_i \in R$, pour tout $x_1, \cdots, x_{n_i} \in X$, on a $r_i(x_1, \cdots, x_{n_i}) \in X$.*

- On dit que cet ensemble est alors **défini inductivement**.

Démonstration du théorème :

- On considère
$$X = \bigcap_{Y \in \mathcal{F}} Y, \qquad (1)$$
  où
$$\mathcal{F} = \{Y \subseteq E | Y \text{ satisfait } (B) \text{ et } (I)\}.$$

- X est l'ensemble recherché.

Noter une définition inductive

- On note souvent une définition inductive
  - aussi sous la forme de **règles de déduction :**

$$\frac{}{B \subseteq X} \qquad \frac{x_1 \in X \quad \dots \quad x_{n_i} \in X}{r_i(x_1, \dots, x_{n_i}) \in X}$$

- Exemples :
  - Définition inductive des entiers pairs précédente.
  - L'ensemble défini par

$$\frac{}{0 \in X} \qquad \frac{n \in X}{n+1 \in X}$$

n'est autre que $\mathbb{N}$ tout entier.

Preuves par induction

Théorème
*Soit $X \subseteq E$ défini inductivement à partir de B et R, et soit $\mathscr{P}(x)$ un **prédicat** exprimant une propriété de $x \in E$.*

Si les conditions suivantes sont vérifiées :
(*B*) *$\mathscr{P}(x)$ est vraie pour chaque $x \in B$;*
(*I*) *Pour chaque règle $r_i \in R$, pour chaque $x_1, \cdots, x_{n_i} \in E$, $\mathscr{P}(x_1), \cdots, \mathscr{P}(x_{n_i})$ vraies impliquent $\mathscr{P}(x)$ vraie en $x = r(x_1, \cdots, x_{n_i})$.*
Alors $\mathscr{P}(x)$ est vraie pour chaque élément $x \in X$.

Démonstration du théorème :

- On considère l'ensemble Y des éléments $x \in E$ qui vérifient le prédicat $\mathscr{P}(x)$.
- Il contient B, il est stable par les règles de R, et donc il contient X.

Quelques applications

- Montrer que tout mot de *Arith* possède autant de parenthèses fermantes qu'ouvrantes.
- On considère le sous-ensemble *ABS* des arbres binaires stricts défini inductivement par :
  (*B*) $(\emptyset, a, \emptyset) \in ABS$, pour chaque $a \in A$.
  (*I*) $g, d \in ABS \Rightarrow (g, a, d) \in ABS$, pour chaque $a \in A$.
  - Montrer qu'un élément de *ABS* est toujours non-vide et sans sommet avec un seul fils non-vide.

Définition explicite d'un ensemble défini inductivement

Théorème

Chaque ensemble X défini inductivement à partir de l'ensemble de base B et des règles R s'écrit aussi

$$X = \bigcup_{n \in \mathbb{N}} X_n,$$

où $(X_n)_{n\in\mathbb{N}}$ est la famille de parties de E définie par récurrence par

- $X_0 = B$
- $X_{n+1} = X_n \cup \{r_i(x_1, \cdots, x_{n_i}) | x_1, \cdots, x_{n_i} \in X_n \text{ et } r_i \in R\}$.

Autrement dit, tout élément de X est obtenu en partant d'éléments de B et en appliquant un nombre fini de fois les règles de R pour obtenir des nouveaux éléments.

Arbre de dérivations

- La caractérisation précédente **par le bas** de X invite à chercher à garder la trace de comment chaque élément est obtenu, en partant de X et en appliquant les règles de R.
  - Exemple, $1+2+3 \in Arith$ car

$$\dfrac{\dfrac{\dfrac{}{1 \in Arith} \qquad \dfrac{}{2 \in Arith}}{1+2 \in Arith} \qquad \dfrac{}{3 \in Arith}}{1+2+3 \in Arith}$$

- On dit qu'une définition inductive de X est **non ambiguë** s'il n'existe qu'une unique façon de construire chaque élément de X.

Exemples

- La définition suivante de $\mathbb{N}^2$ est ambiguë.
  - (B) $(0,0) \in \mathbb{N}^2$;
  - (I) $(n,m) \in \mathbb{N}^2 \Rightarrow (n+1,m) \in \mathbb{N}^2$;
  - (I) $(n,m) \in \mathbb{N}^2 \Rightarrow (n,m+1) \in \mathbb{N}^2$.
- La définition précédente de *Arith* est ambiguë : Le mot $1+2+3$ correspond à la dérivation

$$\dfrac{\dfrac{\dfrac{}{1 \in Arith} \qquad \dfrac{}{2 \in Arith}}{1+2 \in Arith} \qquad \dfrac{}{3 \in Arith}}{1+2+3 \in Arith}$$

Ce n'est pas la seule possible. En effet, on peut aussi écrire

$$\dfrac{\dfrac{}{1 \in Arith} \qquad \dfrac{\dfrac{}{2 \in Arith} \qquad \dfrac{}{3 \in Arith}}{2+3 \in Arith}}{1+2+3 \in Arith}$$

Fonctions définies inductivement

Théorème

Soit $X \subseteq E$ un ensemble défini à partir de B et R de façon non-ambiguë. Soit Y un ensemble.

Pour qu'une application f de X dans Y soit parfaitement définie, il suffit de se donner :

- (B) *la valeur de $f(x)$ pour chacun des éléments $x \in B$;*
- (I) *pour chaque règle $r_i \in R$, la valeur de $f(x)$ pour $x = r_i(x_1, \cdots, x_{n_i})$ en fonction de la valeur de $x_1, \ldots, x_{n_i}$, $f(x_1), \ldots,$ et $f(x_{n_i})$.*

Exemples

- Exemple : La fonction factorielle *Fact* de $\mathbb{N}$ dans $\mathbb{N}$ se définit inductivement par :

  (*B*) $Fact(0) = 1$;
  (*I*) $Fact(n+1) = (n+1) * Fact(n)$.

- Exemple : La valeur h d'un mot de $\mathcal{N}$ se définit par :

  (*B*) $h(a) = a$ pour $a \in \{1,2,\ldots,9\}$.
  (*I*) $h(ga) = 10 * h(g) + a$.

- Les fonctions primitive récursives :
  - voir PC demain.

Syntaxe

Soit $\mathscr{P}$ un ensemble de variables.

Définition (Formules propositionnelles)

L'ensemble des formules propositionnelles $\mathscr{F}$ sur $\mathscr{P}$ est le langage sur l'alphabet $\mathscr{P} \cup \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (,)\}$ défini inductivement par les règles suivantes :

(*B*) *il contient $\mathscr{P}$: toute variable propositionnelle est une formule propositionnelle ;*

(*I*) *si $F \in \mathscr{F}$ alors $\neg F \in \mathscr{F}$;*

(*I*) *si $F, G \in \mathscr{F}$ alors $(F \wedge G) \in \mathscr{F}$, $(F \vee G) \in \mathscr{F}$, $(F \Rightarrow G) \in \mathscr{F}$, et $(F \Leftrightarrow G) \in \mathscr{F}$.*

Décomposition/Lecture unique

La définition inductive est non ambiguë :

Proposition (Décomposition / Lecture unique)

Soit F une formule propositionnelle. Alors F est d'une, et exactement d'une, des formes suivantes

1. *une variable propositionnelle $p \in \mathscr{P}$;*
2. *$\neg G$, où G est une formule propositionnelle ;*
3. *$(G \wedge H)$ où G et H sont des formules propositionnelles ;*
4. *$(G \vee H)$ où G et H sont des formules propositionnelles ;*
5. *$(G \Rightarrow H)$ où G et H sont des formules propositionnelles ;*
6. *$(G \Leftrightarrow H)$ où G et H sont des formules propositionnelles.*

De plus dans les cas 2., 3., 4., 5. et 6., il y a unicité de la formule G et de la formule H avec cette propriété.

Valeur de vérité d'une formule

Une **valuation** est une une fonction de $\mathscr{P}$ vers $\{0,1\}$;

Proposition

Soit v une valuation.
Par le théorème précédent (fonction définie inductivement), il existe une unique fonction définie sur tout $\mathscr{F}$ qui vérifie les conditions suivantes

(*B*) *chaque variable propositionnelle p s'interprète par $v(p)$;*

(*I*) *la négation s'interprète par la négation logique ;*

(*I*) *$\wedge$ s'interprète comme le et logique ;*

(*I*) *$\vee$ s'interprète comme le ou logique ;*

(*I*) *$\Rightarrow$ s'interprète comme l'implication logique ;*

(*I*) *$\Leftrightarrow$ s'interprète comme l'équivalence logique.*

Lorsque F s'évalue en 1, on dit que F est **satisfaite** en v, et que v est un **modèle** de F.

Tautologies, formules équivalentes

- Une **tautologie** est une formule F qui est satisfaite en toute valuation.
- Deux formules F et G sont dites **équivalentes** si elles s'évaluent en 1 (et donc aussi en 0) pour exactement les mêmes valuations.
  - On écrit dans ce cas $F \equiv G$.
- Exemples :
  - La formule $p \vee \neg p$ est une tautologie ;
  - Les formules p et $\neg\neg p$ sont équivalentes.

Quelques équivalences

- Pour toutes formules F et G, les formules suivantes sont des tautologies :

$$(F \Rightarrow F),$$
$$(F \Rightarrow (G \Rightarrow F)),$$
$$(F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))).$$

- Idempotence : pour toute formule F

$$(F \vee F) \equiv F,$$
$$(F \wedge F) \equiv F.$$

- Associativité : pour toutes formules F, G, H

$$(F \wedge (G \wedge H)) \equiv ((F \wedge G) \wedge H),$$
$$(F \vee (G \vee H)) \equiv ((F \vee G) \vee H).$$

Quelques équivalences

- Commutativité : pour toutes formules F et G

$$(F \wedge G) \equiv (G \wedge F),$$
$$(F \vee G) \equiv (G \vee F).$$

- Distributivité : pour toutes formules F, G, H

$$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H)),$$
$$(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H)).$$

- Lois de De Morgan : pour toutes formules F et G

$$\neg(F \wedge G) \equiv (\neg F \vee \neg G),$$
$$\neg(F \vee G) \equiv (\neg F \wedge \neg G).$$

- Absorption : pour toutes formules F et G

$$(F \wedge (F \vee G)) \equiv F$$
$$(F \vee (F \wedge G)) \equiv F.$$

La suite ...

- Demain :
  - Appréhender ce que l'on arrive à définir par induction sur les entiers.
- La semaine prochaine :
  - Calcul propositionnel, Logique du premier ordre.
  - Preuve. Validité. Complétude.

Exprimez vous.

Page du cours.

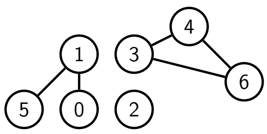
Commentaires, avis
sur les cours et les PCs.

- Page du cours:
  https://moodle.polytechnique.fr/course/view.php?id=19276.
- Commentaires, avis sur les cours et les PCs.
  www.enseignement.polytechnique.fr/informatique/INF412/AVIS.

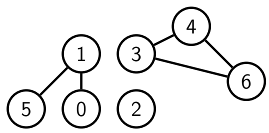
ANNEXES

Un graphe

- Un **graphe** (non orienté) est donné par un couple $G = (V, E)$, où :
  - V est un ensemble ;
  - E est un ensemble de paires $\{u, v\}$ avec $u, v \in V$.
- On convient de représenter une paire $\{u, v\}$ par (u, v) ou (v, u). Autrement dit, (u, v) et (v, u) dénotent la même arête.
- Exemple :
  - $V = \{0, 1, \ldots, 6\}$
  - $E = \{(0,1), (3,4), (5,1), (6,3), (6,4)\}$.

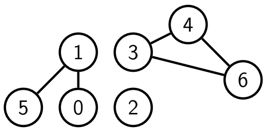


Vocabulaire



- Les éléments e de E sont appelés des **arêtes**. Si $e = (u, v)$, u et v sont appelés les **extrémités** de e.
- u et v sont dits **voisins** s'il y a une arête entre u et v.
- Le degré de u est le nombre de voisins de u.

Vocabulaire



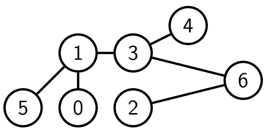
- Un **chemin du sommet** s **vers le sommet** t est une suite $e_0, e_1, \cdots, e_n$ de sommets telle que $e_0 = s$, $e_n = t$, $(e_{i-1}, e_i) \in E$, pour tout $1 \leq i \leq n$.
  - n est appelé la **longueur** du chemin, et on dit que t **est joignable** à partir de s.
  - Le chemin est dit **simple** si les e_i sont distincts deux-à-deux.
  - Un **cycle** est un chemin de longueur non-nulle avec $e_0 = e_n$.
- Le sommet s est dit **à distance** n de t s'il existe un chemin de longueur n entre s et t, mais aucun chemin de longueur inférieure.

Les graphes sont partout !

- Beaucoup de problèmes se modélisent par des objets et des relations entre objets.
- Exemples :
  - Le graphe routier.
  - Les réseaux informatiques.
  - Le graphe du web.
- Beaucoup de problèmes se ramènent à des problèmes sur les graphes.
- Les graphes sont omniprésents en informatique.
- Théorie des graphes :
  - Euler, Hamilton, Kirchhoff, König, Edmonds, Berge, Lovász, Seymour,. . .

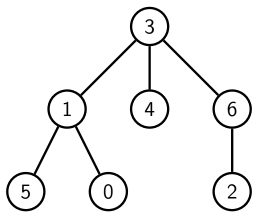
Les arbres

- Un graphe **connexe** est un graphe tels que les sommets sont joignables deux à deux.
- Un graphe connexe sans cycle est appelé un **arbre** (libre).
- Un graphe sans-cycle est appelé une **forêt**.

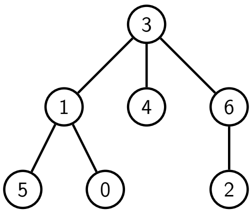


Les arbres poussent de haut en bas en informatique

- On distingue souvent un sommet que l'on appelle **sa racine**.
- On dessine un arbre
  - en plaçant la racine r tout en haut ;
  - puis en plaçant les sommets à distance i de r à la ligne i.
- Exemple : pour l'arbre libre précédent, en prenant le sommet d'étiquette 3 comme racine.

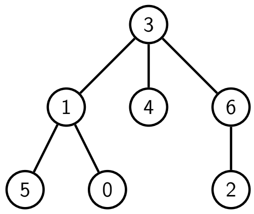


Encore du vocabulaire



- Pour tout sommet u, il existe un unique chemin (simple) entre la racine r et u.
- Tout sommet a sur ce chemin est appelé **un ancêtre** de u. u est dit un **descendant** de a.
- L'avant dernier sommet p sur ce chemin est appelé **le père** de u. u est appelé **un fils** de p.
- Le **sous-arbre** de racine u est l'arbre induit par les descendants de u.

Encore du vocabulaire



- **L'arité** d'un sommet est le nombre de ses fils.
- Un sommet qui n'a pas de fils est appelé **une feuille**.
- La **hauteur d'un sommet** est sa distance à la racine r.
- La **hauteur d'un arbre** est la hauteur de la feuille la plus haute.