Sciences orientation logicielle 2

Mathématiques Discrètes

Support du cours

Enseignant : Niklaus Eggenberg

Email : niklaus.eggenberg@hesge.ch

Support du cours : www.eswys.ch/hepia

Plan du cours

- Ch. 1 Automates et analyse de complexité
- Ch. 2 Théorie des graphes
- Ch. 3 ..

Evaluation: Moyenne arith. 3 travaux pratiques notés

Chapitre 1

Automates & Analyse de complexité

Définitions: Problème et Instance

- Un ***problème*** est une question générique
  - *Déterminer si un nombre n entier est pair ou impair.*
  - *Quel est le PDCD de deux entiers m et n?*

Le problème est indépendant de toute notion de solution, solvabilité et peut avoir plusieurs solutions.

- Une ***instance*** d'un problème est un cas particulier d'un problème
  - *Déterminer si* 10204091 *entier est pair ou impair.*
  - *Quel est le PDCD de deux entiers* 106 *et* 2076*?*

Algorithme

Un ***algorithme*** est une méthode effective de calcul ou un processus de résolution d'un problème par le calcul.

- Premières apparition en 2000 avant J.-C. à Babylone (Irak) pour le calcul des impôts.
- Premier algorithme «moderne» par Euclide (Grèce, 300 avant J.-C.): le calcul du PGCD.
- IX^{e} siècle après J.-C., Al Khuwarzimi, un mathématicien perse (Iran) publie un ouvrage dédié à ces «méthodes». Le mot «algorithme» est dérivé de son nom.
- Aux XVII^{e}, Pascal construit la pascaline, considéré comme l'une des première machine à calculer automatisée.

Décidabilité d'un problème

En 1928, Hilbert soulève la question de décidabilité (***Entscheidungsproblem***)

Etant donné un énoncé mathématique, existe-t-il un algorithme capable de décider si l'énoncé est vrai ?

Plusieurs formalismes (encodages) sont proposés dès 1930 pour tenter de démontrer le résultat, dont la ***machine de Turing*** grâce à la quelle Turing montre qu'il n'existe aucun algorithme pour l'*Entscheidungsproblem*.

Décidabilité d'un problème

- Un problème est dit ***décidable*** s'il existe un algorithme permettant de trouver une solution au problème quelle que soit son instance.

- A l'inverse, un problème est ***indécidable*** s'il n'existe aucun algorithme pour le résoudre.

Note:
Un problème de décision est une problème dont la réponse est soit «Oui» soit «Non». C'est une sous-classe de problèmes.

Exemples: Problème et Instance

- Exemples de problèmes décidables
  - Trouver le PGCD de deux nombres entiers m et n.
  - Déterminer si un nombre entier n est un nombre premier ou non.

- Exemples de problèmes indécidables
  - Soit un polynôme $p(x) = \sum_{i=1}^{n} \alpha_i x^i$, avec $\alpha_i \in \mathbb{Z}$, existe-t-il une valeur entière $x^* \in \mathbb{Z}$ tel que
    $$p(x^*) = 0.$$
  - Soient $F_1, \ldots, F_n$ des formes polygonales, peut-on paver paver le plan, sans recouvrement ni espace vide, avec des exemplaires de $F_1, \ldots, F_n$

Thèse de Church-Turing

Pour tout problème décidable, il existe un machine de Turing pour le résoudre.

Autrement dit

Faisable par un algorithme = Faisable par une machine de Turing

Note:
Il existe d'autres formalismes équivalentes à la machine de Turing (p.ex. λ-calcul, machine à compteurs, foncions récursives, fonctions à pile, automates cellulaires).

Définitions

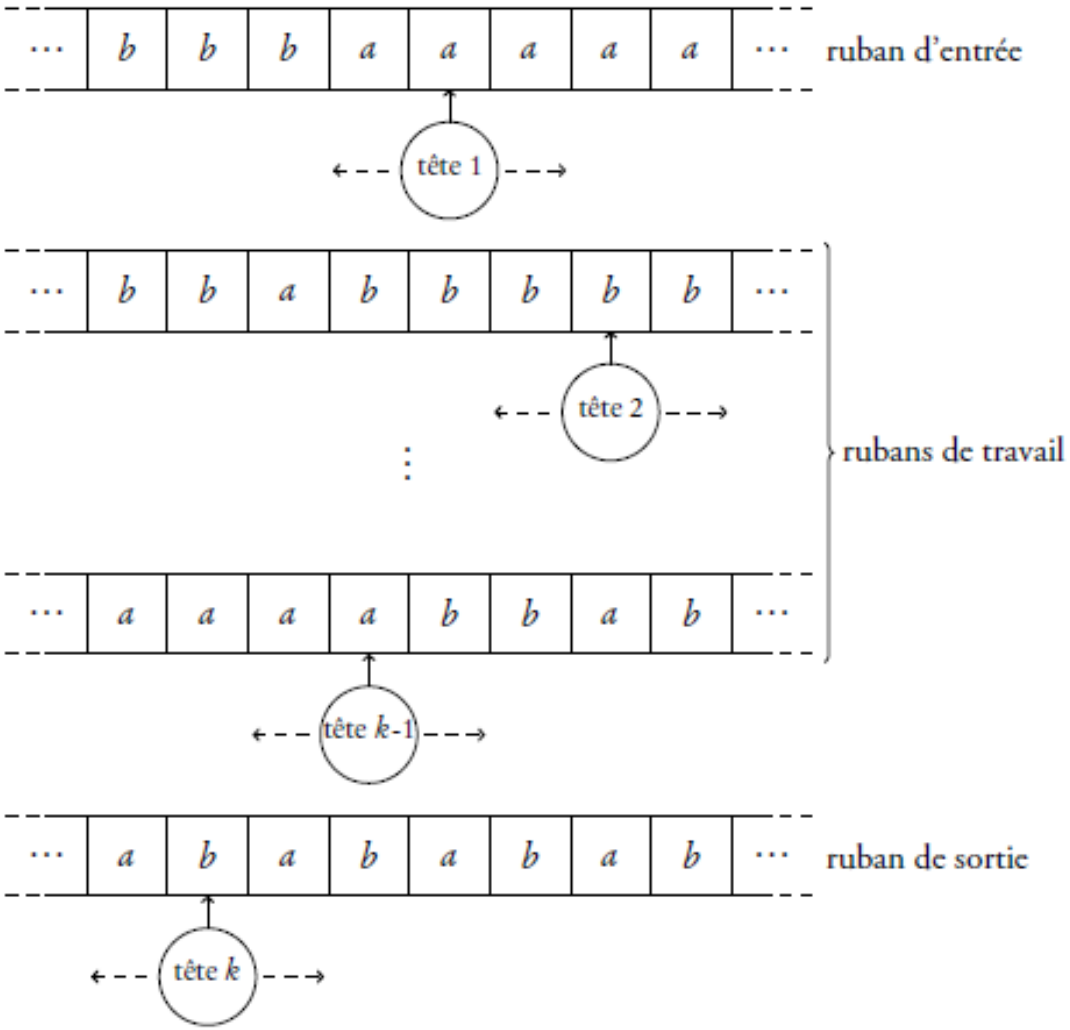
- Un ***alphabet*** est ensemble fini de ***symboles,***
- Le ***symbole vide***, représenté par ϵ,
- Un ***mot*** est une suite de symboles (ou ***concaténation***),
- La ***longueur*** d'un mot correspond au nombre de symboles dont le mot est composé,
- Un ***langage*** est un ensemble de mots,
- Un ***langage fini*** est un ensemble de mots de longueur finie,
- Une ***fonction d'encodage*** permet de traduire un problème dans un langage (fini ou non).

La machine de Turing

La machine de Turing est composée des éléments suivants :

- ***Ruban*** composé de cellules contenant des symboles,
- ***Tête de lecture*** et/ou écriture qui se trouve dans un certain état (position),
- Un **état** correspond au symbole dans la position du ruban,
- Il existe 2 états spéciaux correspondant à l'état initial et à l'état final,
- Une ou plusieurs ***fonctions de transition*** définissant les règles de transition entre les divers états.

La machine de Turing



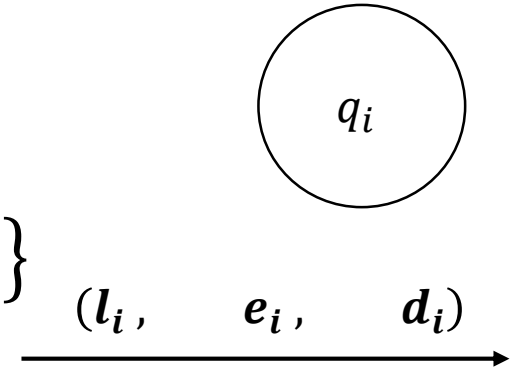
Exemple – Inverser un octet

- Alphabet$= \{null, 0, 1\}$,
- Mots $= \{(b_1, \dots, b_8), | \, b_i \in \{0,1\} \, \forall \, i = 1, \dots, 8\}$,
- **Etat** 1
  La tête se déplace à droite tant qu'elle lit $null$, et passe à l'Etat 2 dès qu'elle lit 0 ou 1,
- **Etat** 2
  Si la tête lit 0 ou 1, écrire l'inverse (1 ou 0) et se déplacer à droite
  Si la tête lit $null$, passage à l'état final (STOP)

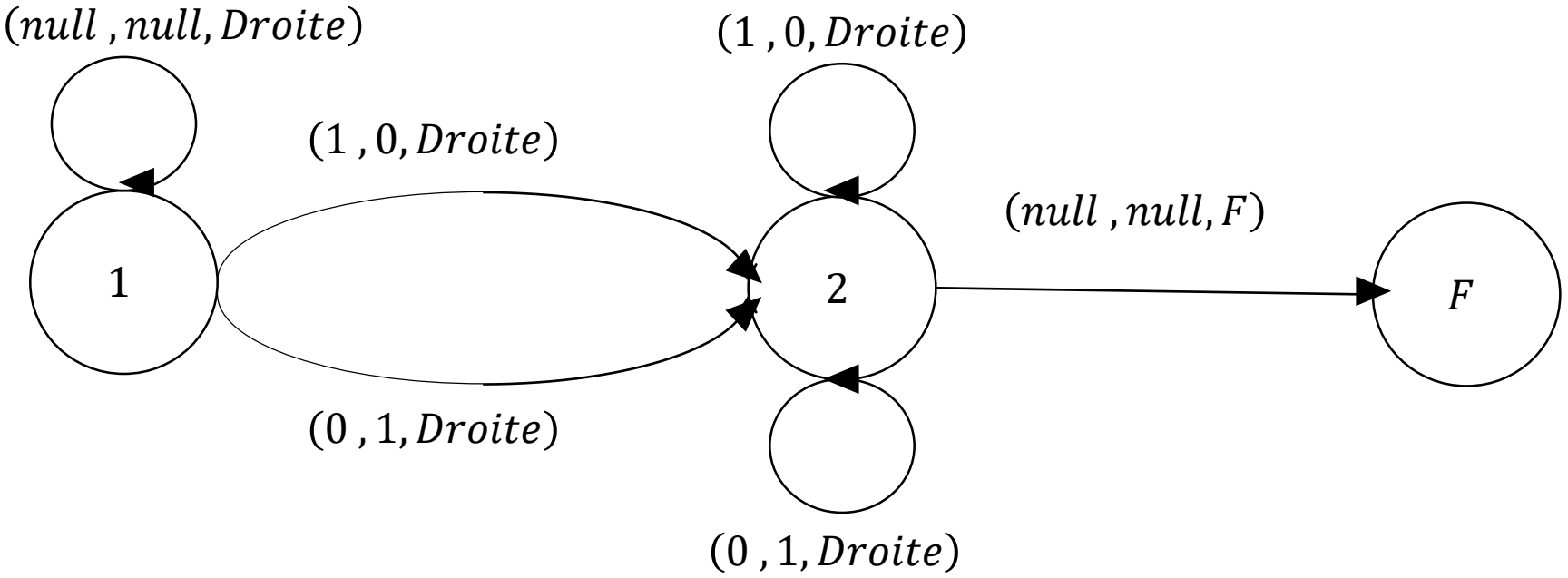
Formalisation

- Un état de la machine est défini par le triplet
  $q = (lect.,\ écr.,\ dir.)$

- La machine de Turing est définie par un ensemble d'états
  $$\mathrm{Q} = \{q_i \mid i = 0, \ldots, F\}$$
  et de transitions
  $$\mathrm{T} = \{(q_i, q_j) \mid i, j = 0, \ldots, F\}$$



Exemple – inversion du bit suite



Exemple à 3 bandes

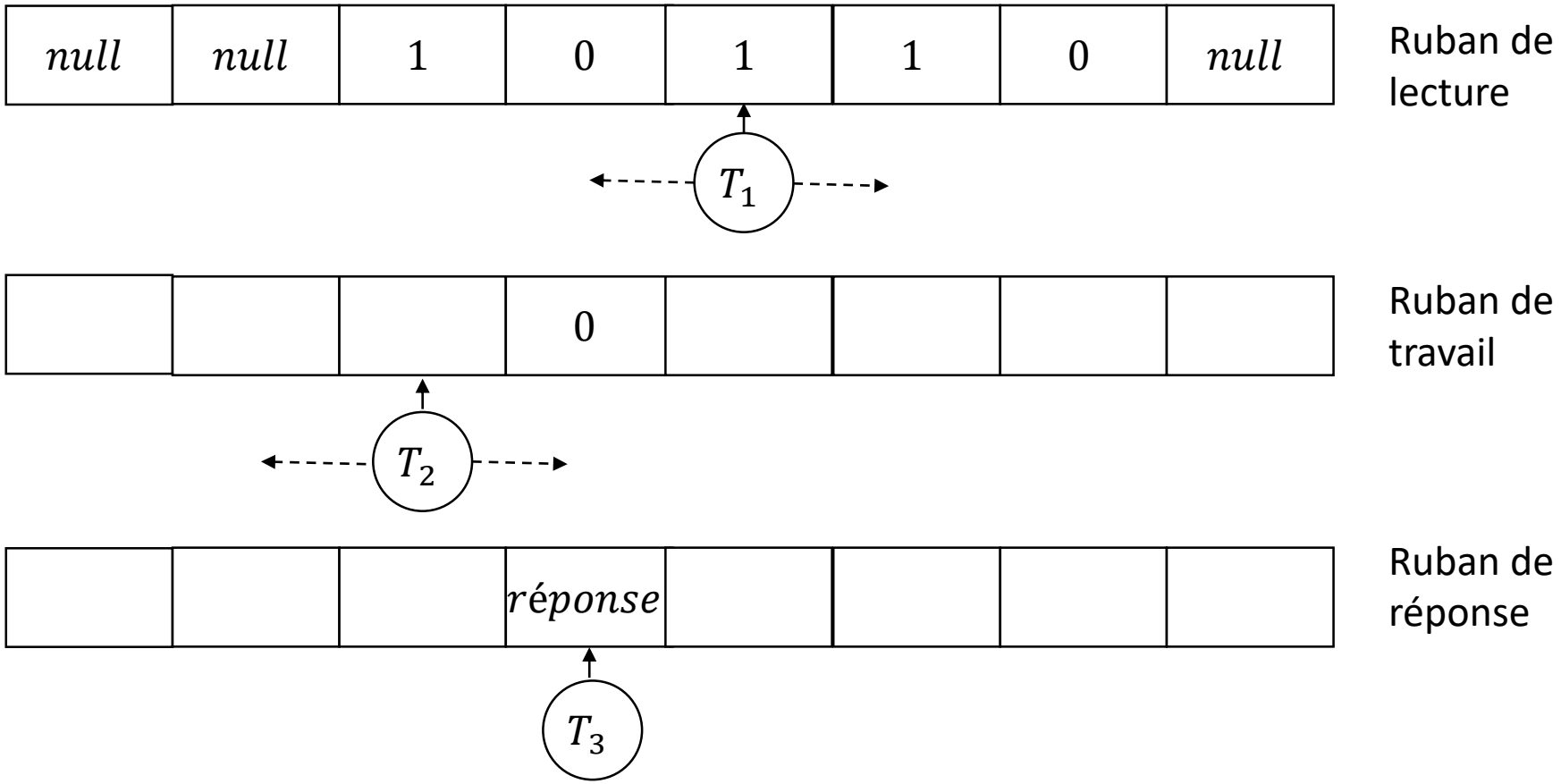
Objectif: tester si le mot en entrée a le même nombre de 0 que de 1.

Procédé:

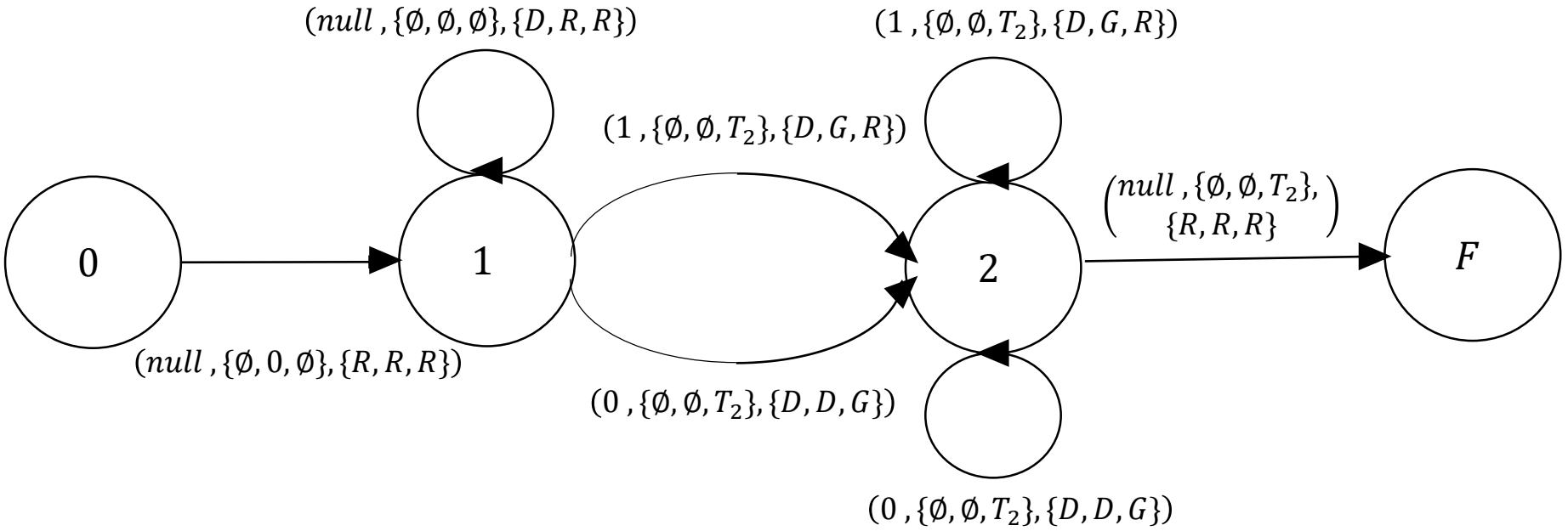
- Ecrire 0 sur le ruban d'écriture à la position 0,
- Lire le mot sur la bande de lecture de gauche à droite
- Si le symbole lu est 0, déplacer le ruban d'écriture à droite,
- Si le symbole lu est 1, déplacer le ruban d'écriture à gauche,
- Ecrire sur le ruban résultat l'état du ruban 2 (1 => «OUI», le nombre est égale, $null$ sinon).

Exemple à 3 bandes

Objectif: tester si le mot en entrée a le même nombre de 0 que de 1.



Exemple – Même nombre de 0 et 1



- R ***R***ester sur place
- D Déplacement à ***D***roite
- G Déplacement à ***G***auche
- $\emptyset$ Ne rien écrire

Exercice:

Construisez une machine de Turing permettant de lire une séquence de bits $\{0, 1\}$ et doublant le nombre de 1.

Exemple : (010100110) devient (011011001110).

Indice: utilisez 2 bandes !

Erratum: Problème et Instance

Le dixième problème de Hilbert suivant

Soit un polynôme $p(x) = \sum_{i=1}^{n} \alpha_i x^i$, avec $\alpha_i \in \mathbb{Z}$, existe-t-il une valeur entière $x^* \in \mathbb{Z}$ tel que

$$p(x^*) = 0.$$

N'est <u>PAS</u> décidable!

Erratum: Problème et Instance

Le dixième problème de Hilbert suivant

Soit un polynôme $p(x) = \sum_{i=1}^{n} \alpha_i x^i$, avec $\alpha_i \in \mathbb{Z}$, existe-t-il une valeur entière $x^* \in \mathbb{Z}$ tel que

$$p(x^*) = 0.$$

N'est PAS décidable!

En revanche, si on fixe $n < 3$, alors le problème est décidable.

Propriétés des machines de Turing

- Le ***résultat*** d'un mot correspond à tous les éléments non-vides du ruban résultat,
- Un mot est dit ***accepté*** par une machine de Turing si ladite machine se termine sur l'état final (la machine ne boucle pas et ne rejette pas le mot),
- Un langage est dit ***accepté*** par une machine de Turing si l'ensemble des mots du langage sont acceptés par ladite machine,
- Le ***temps de calcul*** d'un mot est le nombre d'étapes effectuées entre l'étape initiale et l'étape finale,
- ***L'espace*** utilisé est le nombre total de cases différentes visitées au cours du calcul par les *têtes du ruban de travail* (donc l'espace des rubans d'entrée et de sortie n'est pas compté).

Machines déterministes ou non

- Une machine est dite ***déterministe*** si, pour chaque état donné, l'action élémentaire effectuée est la même.
  - Toutes les machines encodées par une machine de Turing sont déterministes.

- Une machine ***non-déterministe*** est une machine qui peu effectuer un choix non-déterministe, ou aléatoire, de la transition.
  - Une machine non-déterministe ne peut pas être encodée par une machine de Turing standard.

Equivalences

- Une machine de Turing considère que les rubans sont infinis. Qu'ils soient infinis à gauche, à droite ou les deux est équivalent.

- Il existe un ***machine de Turing universelle*** permettant de simuler toutes les machines de Turing.
  - Cette machine a 5 rubans,
  - La description et la preuve nécessitent un formalisme qui dépasse le contenu de ce cours.

- Il est possible de construire un langage non-décidable à partir d'un langage décidable.

Preuve par la méthode diagonale

- La méthode de la preuve par le procédé diagonal fut introduite par Cantor en 1891 pour prouver que $\mathbb{R}$ n'est pas dénombrable.

- Idée par l'absurde : on suppose qu'un résultat soit vrai et, avec une liste infinie, on construit un élément qui ne fait pas partie de ladite liste.

Nous allons illustrer cette méthode avec la preuve que $\mathbb{R}$ n'est pas dénombrable.

$\mathbb{R}$ n'est pas dénombrable

- Si $[0,1)$ n'est pas dénombrable, $\mathbb{R}$ ne l'est pas non plus;
- Supposons $[0,1)$ dénombrable. Il existe donc une suite $(\mathrm{r}_1, \mathrm{r}_2, \dots)$ avec un nombre infini associant un $\mathrm{r_i}$ à tout élément dans $[0,1)$;
- $\mathrm{r_i}$ est composé d'une infinité de décimales (ou un nombre fini de décimales mais une infinité de 0);
- Ecrivons donc $\mathrm{r_i} = 0.\mathrm{r_{i1}}\ \mathrm{r_{i2}}\ r_{i3}\ \dots r_{in} \dots$ sous forme décimale;
- Construisons le réel suivant : $x = 0.\delta_{11}\ \delta_{22}\ \delta_{33} \dots \delta_{nn} \dots$ où

$$\delta_{ii} = \begin{cases} 1\ si\ r_{ii} \neq 1 \\ 2\ si\ r_{ii} = 1 \end{cases}$$

$\mathbb{R}$ n'est pas dénombrable

- Exercice : construisez $x \in [0,1)$ pour la suite suivante

$r_1 = 0, \mathbf{0}\ 1\ 0\ 5\ 1\ 1\ 0 \ldots$
$r_2 = 0, 4\ \mathbf{1}\ 3\ 2\ 0\ 4\ 3 \ldots$
$r_3 = 0, 8\ 2\ \mathbf{4}\ 5\ 0\ 2\ 6 \ldots$
$r_4 = 0, 2\ 3\ 3\ \mathbf{0}\ 1\ 2\ 6 \ldots$
$r_5 = 0, 4\ 1\ 0\ 7\ \mathbf{2}\ 4\ 6 \ldots$
$r_6 = 0, 9\ 9\ 3\ 7\ 8\ \mathbf{1}\ 8 \ldots$
$r_7 = 0, 0\ 1\ 0\ 5\ 1\ 3\ \mathbf{0} \ldots$

$\mathbb{R}$ n'est pas dénombrable

- Réponse

$$\delta_{11} = 2$$
$$\delta_{22} = 2$$
$$\delta_{33} = 1$$
$$\delta_{44} = 1$$
$$\delta_{55} = 1$$
$$\delta_{66} = 2$$
$$\delta_{77} = 1$$

$\mathbb{R}$ n'est pas dénombrable

- Posons $x = 0.\delta_{11}\delta_{22}\delta_{33}\ldots$
- Clairement, $x \in [0,1)$;
- Supposons $x \in \{r_i, i = 1, 2, 3, \ldots, n, \ldots\}$, il existe donc un élément i^* tel que $x = r_{i^*}$;
- Par définition de x, nous avons que la i^*-ème décimale de x est

$$r_{i^*i^*} = \begin{cases} 1 \; si \; r_{i^*i^*} \neq 1 \\ 2 \; si \; r_{i^*i^*} = 1 \end{cases}$$, ce qui conduit à une contradiction :

$$r_{i^*i^*} \neq r_{i^*i^*}.$$

- Nous avons donc construit un élément $x \notin \{r_i, i = 1, 2, 3, \ldots, n, \ldots\}$, qui est, par hypothèse, un ensemble dénombrable de $[0, 1)$.
- Nous concluons que l'hypothèse est fausse, et donc, que $[0, 1)$ et tous les espaces qui contiennent $[0, 1)$ (dont $\mathbb{R}$), ne sont pas dénombrables.

Langage non-décidable

- Soit $W = \{w_0, w_1, \ldots, w_n, \ldots\}$ l'ensemble infini de tous les mots finis (il en existe une infinité),
- Soit $M = \{ M_0, \ldots, M_n, \ldots\}$ l'ensemble infini de toutes les machines de Turing,
- Soit le tableau $A = \left\{ a_{ij} = \left[\begin{array}{c} 0 \; si \; M_i \; accepte \; w_j \\ 1 \; si \; M_i \; n'accepte \; pas \; w_j \end{array} \right\} \right.$,
- Alors le langage $L_0 = \{w \mid w = w_i \wedge A[i,i] == 1\}$ n'est pas décidable.

Note: $\wedge$ et l'opérateur binaire «*et*»

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Preuve – par l'absurde

- Supposons que L_0 est accepté par une machine de Turing.
- Il existe alors une machine $M_{j^*} \in M = \{ M_0, \dots, M_n, \dots \}$ qui accepte L_0, car M est l'ensemble de ***toutes*** les machines de Turing.
- Si $w_{j^*} \in L_0$, cela implique, par définition de L_0, que $A[j^*, j^*] = 1$, donc que la machine M_{j^*} n'accepte pas w_{j^*}, ce qui contredit que L_0 est accepté par M_{j^*}.
- Si $w_{j^*} \notin L_0$, cela implique, par définition de L_0, que $A[j^*, j^*] = 0$, donc que la machine M_{j^*} accepte w_{j^*}, ce qui contredit que L_0 est accepté par M_{j^*}.
- Dans les deux cas, nous avons une contradiction, prouvant que l'hypothèse est fausse : donc il n'existe pas de machine de Turing acceptant L_0.

Langage non-décidable

- Le langage L_0 n'est pas accepté par une machine de Turing et est donc non-décidable.

- Question: existe-t-il des langages acceptés par une machine de Turing mais non-décidables ?

- Réponse – OUI:

$\bar{L}_0 = \{w \mid w = w_i \wedge A[i,i] == 0\}$ est un langage accepté par une machine de Turing, mais non-décidable.

Quelques problèmes indécidables

- Déterminer si une machine de Turing s'arrête lorsqu'elle est lancée sur le mot vide est indécidable.

- Déterminer si une machine de Turing s'arrête au moins sur un mot est indécidable (problème de l'arrêt existentiel).

- Déterminer si une machine de Turing s'arrête sur tous les mots d'un langage est indécidable (problème de l'arrêt universel).

<u>Conséquence:</u>
Il n'est pas possible de construire un programme de test permettant de tester si un programme existant se termine sur toutes les instances du problème.

Théorie de la complexité

- La ***complexité*** d'un problème s'exprime en nombre d'étapes requises par un algorithme pour résoudre ***toute instance*** dudit problème;

- Elle est exprimée en fonction de la ***taille*** de l'instance;

- L'étude de la complexité s'exprime donc souvent au «*pire des cas*» et pour une taille de problème suffisamment grande (non-triviale, p.ex. taille 1 ou 2);

- La complexité ne se mesure que pour les problèmes décidables en un nombre fini d'étapes pour une instance de taille finie.

Théorie de la complexité - Formalisation

Pour un problème donné,

- Désignons par n la taille d'une instance donnée et I_n un instance de taille n;
- Pour un algorithme A donné, le temps de résolution de I_n, dénoté $\tau_{\mathrm{A}}(I_n)$, est une fonction;
- Si A est un algorithme fini (il se termine en temps fini pour une instance de taille finie), alors il existe une constante c telle que et une fonction $f : \mathbb{N} \rightarrow \mathbb{N}$ tels que

$$\tau_A(I_n) \leq c \cdot f(n), \qquad \forall\, I_n$$

- De manière générale, nous ignorons la constante c pour ne nous intéresser qu'à la fonction de complexité, que nous écrirons

$$\tau_A(I_n) \leq O_A(f(n)), \qquad \forall\, I_n$$

Exemples de complexité

Pour un problème donné,

- La complexité est donc une mesure fortement liée à l'algorithme utilisé pour la résolution;
- La complexité d'un problème est définie comme la complexité du plus efficace des algorithmes pour résoudre le problème. Nous la noterons O.
- Il est souvent difficile, voir impossible, de prouver qu'un algorithme est «le meilleur» pour un problème donné;
- En revanche, s'il existe un algorithme avec une complexité $O_A(f(n))$ donnée, alors $O \leq O_A(f(n))$.

P vs NP

Question fondamentale

- La complexité d'un problème est-elle polynômiale en fonction de sa taille ?

- Si «Oui», quel est le degré du plus grand facteur ? Est-il constant (1,2,3…) ou dépendant de la taille ?

- P est l'ensemble des langages reconnus par une ***machine déterministe*** en temps polynomial.

- NP est l'ensemble des langages reconnus par une ***machine non-déterministe*** en temps polynomial.

- Bien que cela n'ait pas été prouvé, il est supposé que $P \neq NP$.

Définitions

- Réduction ($B \leq A$)

Un problème B peut être réduit à un problème A s'il existe fonction de transition permettant de résoudre B en utilisant un algorithme permettant de résoudre A.

- Problème NP-dur (ou NP-difficile):

Un problème A est dit NP-dur pour une réduction $\leq$ si pour tout problème B dans NP, on a $B \leq A$.

- Problème NP-complet:

Un problème A est dit NP-complet pour une réduction $\leq$ si $A \in$ NP et si A est NP-dur.

Remarques

Les affirmations suivantes sont équivalentes :

- $P = NP$;
- Tout problème NP-complet est dans P;
- Il existe un problème NP-complet dans P.

Exemple de problème NP-Complet

Problème SAT (problème de *satisfaisabilité*)

Soit une formule booléenne

$$\varphi(x_1, x_2, \dots, x_n) \text{ où } x_i \in \{0,1\}, \forall i = 1, \dots n$$

Le problème SAT consiste déterminer si la formule est ***satisfaisable***, autrement dit, s'il existe une *assignation* $a_1, \dots, a_n$ telle que $\varphi(a_1, a_2, \dots, a_n) = VRAI$.

Théorème (Cook 1971, Levin 1973):
Le problème SAT est NP-complet.

NP-complétude de SAT

Idée de la preuve:

Il est facile de montrer que SAT $\in$ NP (il suffit de deviner une assignation et la vérification de celle-ci se fait en temps polynomial) et donc SAT $\in$ NP $\supseteq$ P.

Il existe une formule polynomiale pour décrire la résolution du problème SAT sur une machine non-déterministe fonctionnant en temps polynomial.

Autres problèmes NP-complets

- Dans un graphe donné, déterminer s'il existe k sommets non-reliés deux-à-deux;

- NOTE: si k est fixé, alors le problème peut être résolu en temps polynômial, car la vérification se fait en temps $O(n^k)$!

- Pour une liste d'entiers et un entier données, existe-t-il un sous-ensemble de la liste tel que leur somme égale l'entier en question ?

Introduction à la théorie des graphes

- Un ***graphe*** $G = (V, E)$ est défini par
  l'ensembles des ***sommets*** («vertices») $V = \{v_1, v_2, \dots, v_n\}$ et
  l'ensemble des ***arêtes*** («edges») $E = \{e_1, e_2, \dots, e_m\}$;

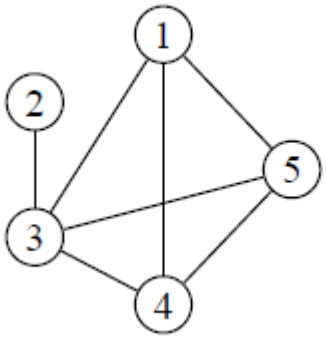
- La taille des deux ensembles et FINI : $\mathrm{n} = |V| < \infty$ et $\mathrm{m} = |E| < \infty$

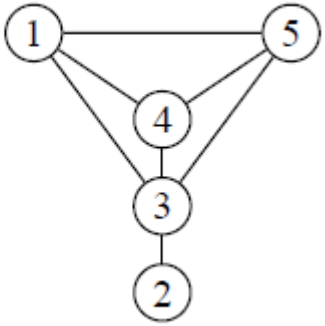
- Toute arrête est définie par une paire (non ordonnée) de sommets qu'elle relie, nous noterons ceci $e_i = (v_j, v_k)$,

- On dit que deux sommets reliés v_j et v_k sont ***adjacents***,

- Nous appelons ***l'ordre*** du graphe son nombre de sommets, soit $\mathrm{n} = |V|$

Représentation graphique





Voici deux exemples de graphes, le premier est non planaire, les second l'est.

Un graphe planaires est un graphe dont les sommets peuvent être ordonnées de façon à ce qu'aucune arrête ne se croise.

Quelques types de graphes

- Un graphe ***planaire*** est un graphe dont les sommets peuvent être arrangés de sorte à ce qu'aucune arête ne se croise (elle ne sont pas obligatoirement rectilignes !),
- Un graphe est dit ***simple*** si au plus une arête relie deux sommets,
- Si un graphe n'est pas simple, on dit que c'est un ***multigraphe***.
- Un graphe est dit ***connexe*** si, à partir d'un sommet quelconque, on peut atteindre tous les autres sommets en empruntant une suite d'arêtes,
- Si un graphe n'est pas connexe, il est alors composé de 2 ou plus ***composantes connexes***, qui sont des sous-graphes qui, eux, sont connexes.

Quelques types de graphes - suite

- Un graphe ***complet*** est un graphe simple ou toute paire de sommets distincts sont reliés par une arête,

- Un graphe est dit ***biparti*** si ses sommets peuvent être divisés en deux ensembles $X \subseteq V$ et $Y \subseteq V$, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y.

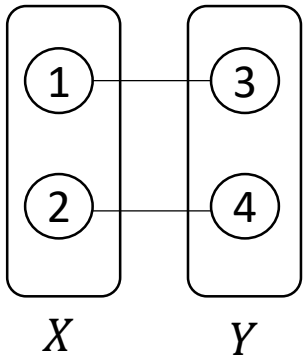
Exercices

- Un graphe ***complet*** d'ordre ≥ 3 peut-il être ***biparti*** ?
- Donnez un exemple d'un graphe biparti non connexe.
- Un graphe complet d'ordre ≥ 4 peut-il être planaire ?

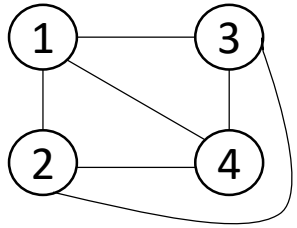
Réponses

- Si pour un graphe d'ordre ≤ 2 on peut trouver des partitions, cela n'est plus possible dès lors que nous avons 3 sommets. La preuve se fait par l'absurde

- Le graphe suivant :



- Oui, car les arêtes ne doivent pas obligatoirement être rectilignes :



Application de la théorie des graphes

- Nous avons 4 élèves devant suivre chacun son programme parmi 4 cours durant chacun 1h

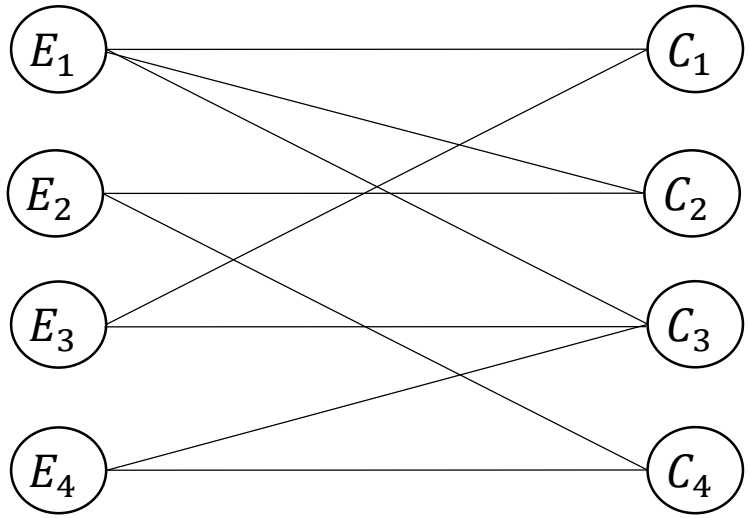
  E_1 doit suivre C_1, C_2 et C_3,
  E_2 doit suivre C_2 et C_4,
  E_3 doit suivre C_1et C_3,
  E_1 doit suivre C_3 et C_4.

Combien de plages horaires faut-il pour dispenser tous les cours une et une seule fois ?

Problème du cours – le graphe

- Attribuons un sommet à chaque élève et chaque cours.

- Relions un sommet élève avec tous les cours que celui-ci doit suivre.

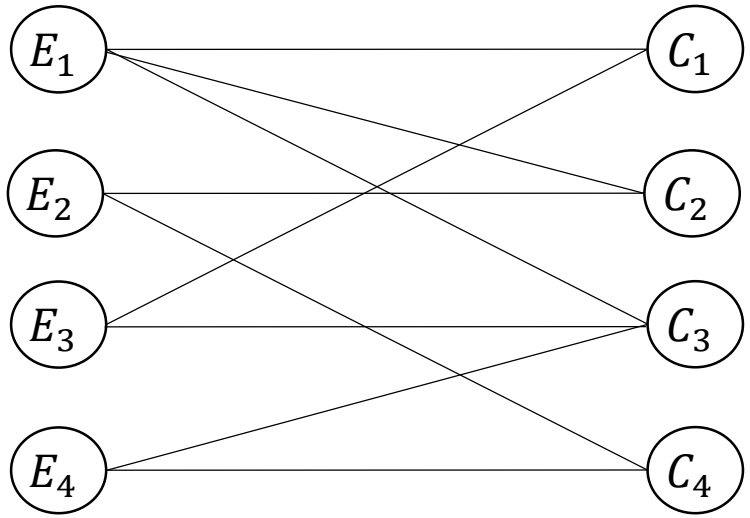


- Attribuons une couleur à chaque plage horaire. Il nous faut donc

Problème du cours – le graphe

- Attribuons un sommet à chaque élève et chaque cours.

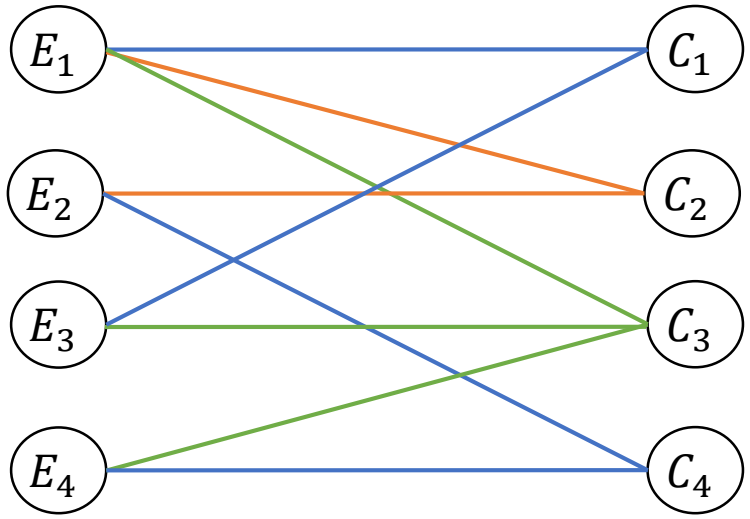
- Relions un sommet élève avec tous les cours que celui-ci doit suivre.



- Attribuons une couleur à chaque plage horaire. Un même élève ne peut avoir 2 arêtes de même couleur alors que le cours, lui, peut – Pourquoi ?

Problème du cours – le graphe

Coloration des arêtes – c'est possible avec 3 couleurs : les cours C_1 et C_4 peuvent avoir lieu simultanément !



Questions complémentaires

- De quel type de graphe s'agit-il ?

- Quelle est la condition nécessaire pour que deux cours puissent avoir lieu simultanément ?

Questions complémentaires

- C'est un graphe biparti (les élèves forment un groupe, les cours un autre),

- Pour que deux cours aient lieu simultanément, aucun élève ne doit être relié simultanément aux deux cours. Autrement dit, si on considère le ***sous-graphe*** composé de TOUS les élèves et uniquement les deux cours, ce dernier doit être non-connexe.

ATTENTION: être non-connexe ne suffit pas, SAUF si nous nous limitons au sous-graphe composé de 2 cours, et le sous-ensemble des élèves participant à l'un ou l'autre des cours (ou les deux) !

Sous-graphe et graphe partiel

- Soit $G = (V, E)$ un graphe
- Un sous-graphe $G' = (V', E')$ est un graphe tel que $V' \subseteq V$ et $E' \subseteq E$.
- Un graphe partiel $G' = (V, E')$ est un graphe avec les mêmes sommets que le graphe initial, mais un certain nombre d'arêtes en moins.

Exemples : si un graphe G est non-connexe, chaque composante connexe de G est une sous-graphe.

NOTE: tout graphe partiel est un sous-graphe !

Degré

- Le degré d'un sommet $\mathbf{v} \in V$ est noté $d(v)$. Il s'agit du nombre d'arêtes incidentes à $\mathbf{v}$.

- Le degré d'un graphe $\mathbf{d}(G)$ est le degré maximum de tous ses sommets.

ATTENTION : si une arête relie un graphe à lui-même (multigraphe), l'arête compte double !

Notez que

$$\sum_{v \in V} d(v) = 2 \times |E| = 2 \times m\,,$$

Cela vaut aussi bien pour les graphes simples que les multigraphes.

Applications

La théorie des graphes est appliquée dans de très nombreux domaines, dont les réseaux, la planification, tous les types de problèmes d'affectation…

Il y a deux mondes distincts dans la théorie des graphes : les graphes orientés et les graphes non-orientés.

Nous nous focaliserons ici sur les graphes non-orientés.

Exercice relationnel

Prouvez que dans un ensemble de 6 personnes, il existe toujours au moins 3 personnes se connaissant OU 3 personnes ne se connaissant pas mutuellement.

Est-ce toujours vrai pour un groupe de 5 personnes ?

Corrigé

Représentons le cas avec un graphe ayant 1 sommet pour chaque personne. Deux sommets sont reliés par une arête noire si les personnes se connaissent mutuellement, et une arête rouge si elles ne se connaissent pas.

Pour chaque sommet, il y a donc au moins 3 arcs incidents de la même couleur.

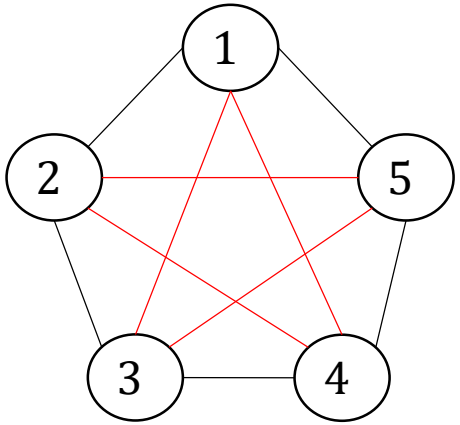
Prenons le sommet 1 et supposons qu'il ait 3 arcs noirs vers 2, 3 et 4. Or, si une des arêtes (2,3), (2,4) ou (3, 4) est noire, on a un graphe complet de taille 3 (3 personnes se connaissent mutuellement). Donc, il faut que les 3 arêtes soient rouge. Mais alors, on a 3 personnes qui ne se connaissent pas mutuellement (2, 3 et 4) !

Le même raisonnement s'applique si on suppose que le sommet 1 a 3 arcs rouges.

Corrigé

La preuve précédente est valable car nous avons choisi arbitrairement les sommes 1, 2, 3 et 4, mais sans perte de généralité pour autant.

Pour 5 personnes, voici un contre-exemple où pour chaque triplet, 2 personnes se connaissent OU 2 personnes ne se connaissent pas.



Arbres

Un ***arbre*** est un graphe connexe sans cycle.

Une ***forêt*** est un graphe non-connexe sans cycle.

Une ***feuille*** est un sommet «pendant» de degré 1.

Arbres - propriétés

Les affirmations suivantes sont équivalentes :

- G est un arbre,
- G est sans cycle et connexe,
- G est sans cycle et possède $m - 1$ arêtes,
- G est connexe et possède $m - 1$ arêtes,
- Chaque paire de sommets distincts est relié par une seule chaîne simple.

Chaîne simple

Note :

Une ***chaîne*** est une alternance de sommets et d'arêtes, commençant et se terminant par un sommet.

Une chaîne est dite simple si une arête apparaît au plus 1 fois.

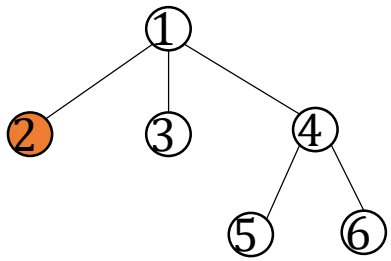
Codage de Prüfer (1896-1934)

Objectif : représenter un arbre sous forme de code – suite S de $n-2$ nombres entiers (les répétitions sont possibles).

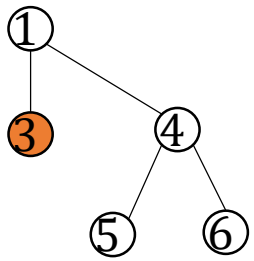
Codage : Tant que l'arbre G a plus de 2 sommets, répéter

1. Identifier la feuille v ayant le numéro minimum,
2. Ajouter à S le seul sommet adjacent à v (son parent),
3. Supprimer de G le sommet v ainsi que l'arête incidente.

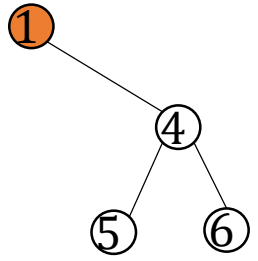
Codage de Prüfer Exemple



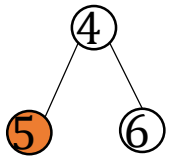
S = {}



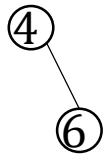
S = {1}



S = {1, 1}



S = {1, 1, 4}



S = {1, 1, 4, 4}

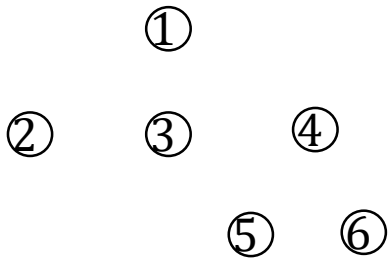
Fin (il reste 2 sommets)

Décodage

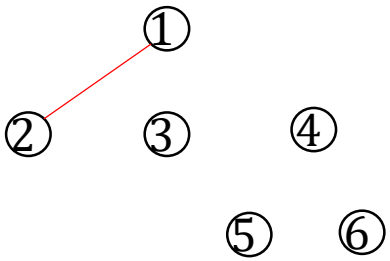
Décodage : Ayant une suite S de $n-2$ sommets provenant des sommets $I = \{1, \dots, n\}$, répéter, tant que S n'est pas vide :

1. Identifier le plus petit élément $i \in I$ n'apparaissant pas dans S,
2. Relier par une arête le sommet i avec le sommet $s \in S$ correspondant au premier élément de la suite S,
3. Retirer i de I et s de S.
4. Les deux éléments qui restent dans I à la fin représentent les extrémités de la dernière arête à ajouter au graphe G.

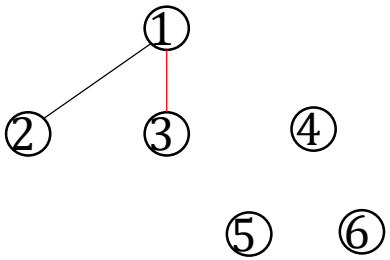
Codage de Prüfer Exemple



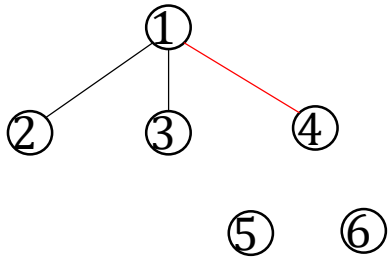
$S = \{1, 1, 4, 4\}$
$I = \{1, 2, 3, 4, 5, 6\}$



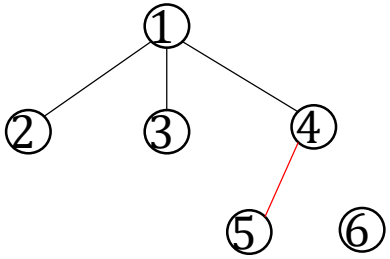
$S = \{1, 1, 4, 4\}$
$I = \{1, 2, 3, 4, 5, 6\}$



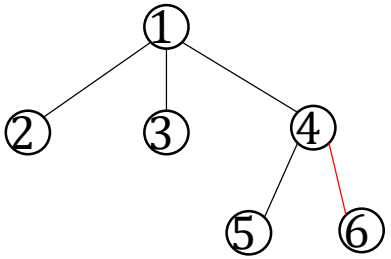
$S = \{1, 4, 4\}$
$I = \{1, 3, 4, 5, 6\}$



$S = \{4, 4\}$
$I = \{1, 4, 5, 6\}$



$S = \{4\}$
$I = \{4, 5, 6\}$

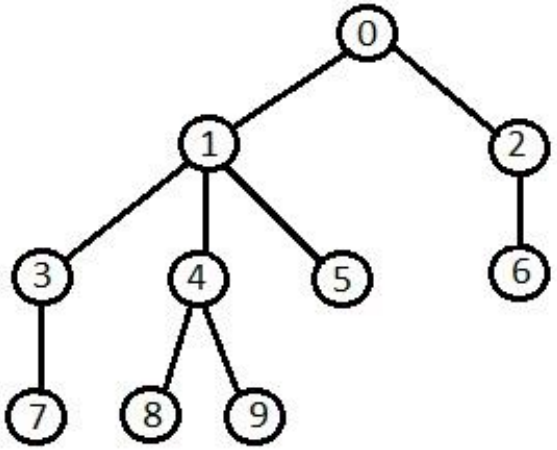


$S = \{\}$
$I = \{4, 6\}$

FIN

Exercice :

1. Appliquez l'algorithme de codage de Prüfer à l'arbre suivant :



2. Reconstruisez l'arbre du code $S = \{1,1,1,1,1\}$.

Corrigé :

1. La séquence de codage est S = {1, 2, 0, 1, 3, 1, 4, 4}.

2. Il s'agit d'un graphe avec le nœud 1 relié au enfants 2, 3, 4, 5, 6 et 7.§

Couplages

Dans un graph $G = \{V, E\}$, nous définissons un ***couplage*** comme un sous-ensembles d'arêtes $C \subseteq E$ tel qu'aucune paire d'arêtes de C n'ont de sommet commun.

Un couplage est dit ***maximum*** s'il est impossible d'ajouter une arête au couplage.

Les sommets couverts par un couplage sont dits des sommets ***saturés***.

Un ***couplage parfait*** est un couplage couvrant tous les sommets du graphe.

Exercice

Soit la matrice suivante décrivant qui parle quelle langue :

	Allemand	**Anglais**	**Arabe**	**Chinois**	**Français**	**Espagnol**	**Russe**
P_1				X	X		
P_2	X					X	
P_3					X	X	
P_4	X						
P_5		X					
P_6		X			X		X
P_7		X			X		
P_8			X	X		X	
P_9					X		X
P_{10}		X	X				
P_{11}	X	X					
P_{12}							X

Exercice

Le but est de trouver le maximum de binômes parlant la même langue (un personne ne pouvant être que dans un seuls binôme à la fois).

Modélisez ce problème sous forme de problème de couplage maximum.

Comment trouver le couplage maximum ? Est-il parfait ?

Corrigé

- Créez un nœud pour chaque personne P_1,…, P_{12},
- Reliez P_i à P_j s'ils ont au moins une langue en commun,
- Il existe au moins un couplage parfait, c'est donc une solution optimale :

(P_1, P_9), (P_2, P_8), (P_3, P_7), (P_4, P_{11}), (P_5, P_{10}), (P_6, P_{12}).

Problèmes de coloration

Etant donné un graphe $G = (V, E)$, trouver une partition des sommes de sorte que tout les éléments d'un même sous-groupe ne sont pas reliés 2 à 2.

Concept : attribuer une couleur à tous les sommets d'un même sous-groupe, 2 sommets de même couleur ne peuvent pas être reliés par un sommet.

Nous appelons une telle partition une ***coloration*** du graphe.

Nombre chromatique

Le nombre chromatique d'un graphe, noté $\gamma(G)$, correspond au nombre minimum de couleurs requis pour colorer un graphe.

Exercices

1. Quel est le nombre chromatique d'un graphe biparti ?

2. Quel est le nombre chromatique d'un graphe complet de taille m sommets ?

3. Quel est le nombre chromatique d'un arbre ?

Corrigé

1. 2 : par définition, le graphe biparti peut être décomposé en deux sous-ensembles de sommets non-adjacents entre eux.

2. Le graphe étant complet, chaque sommet est relié à tous les autres. Il est donc impossible d'avoir 2 sommets de même couleur – il faut donc m couleurs.

Corrigé

3. La réponse est 2 :

Définissons un nœud ***racine*** quelconque dans le graphe. Ce choix est purement arbitraire !

Nous définissons la ***distance*** d'un nœud à la racine le nombre d'arêtes séparant la racine au nœud.

Prenons un nœud v à distance d de la racine. v est relié à 1 seul nœud à distance $d-1$ et 1 ou plusieurs nœuds à distance $d+1$. n n'est donc relié à aucun autre nœud à distance d de la racine. De plus, il n'est relié qu'à des nœuds à distance $d-1$ ou $d+1$.

Par conséquent, nous pouvons séparer tout arbre en 2 groupes – les nœuds à distance impaire de la racine et les nœuds à distance paire de la racine (ce second groupe inclut la racine !).

Toutes les arêtes de l'arbre sont entre ces deux groupes. Un arbre est donc un graphe biparti – qui peut être coloré avec 2 couleurs !

Exercice

Suite à un sommet politique à huis clos, il est advenu que les politiciens rencontrés selon la matrice d'adjacence suivante :

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
P_1				1	1		
P_2				1	1	1	1
P_3					1		1
P_4	1	1			1		
P_5	1	1	1	1		1	1
P_6		1			1		1
P_7		1	1		1	1	

Sachant que les politiciens se sont forcément rencontrés s'ils se trouvaient dans la salle simultanément, quelle est la taille du plus grand groupe réuni simultanément ?

Corrigé

Construisons le graphe avec 1 sommet pour chaque politicien, deux sommets étant reliés s'il y a eu rencontre entre les deux politiciens. Le graphe n'est pas complet – il ne pouvait donc pas y avoir une séance avec tout le monde !

Il s'agit alors de colorer le graphe avec un nombre minimum de couleur – une couleur correspondant à 1 siège dans la salle.

Ayant un sous-graphe complet de taille 4 (P_2, P_5, P_6 et P_7), il faut au moins 4 couleurs – et on peut colorer le graphe en 4 couleurs. Cela nous donne donc une attribution des sièges.

Graphes - Equivalences

Proposition :

Le problème de coloration des arêtes d'un graphe $G = (V, E)$ NON ORIENTE est équivalent à un problème de coloration de sommets d'un graphe $G' = (V', E')$.
On appelle G' ***line graph*** ou ***graphe adjoint*** de G.

Preuve :

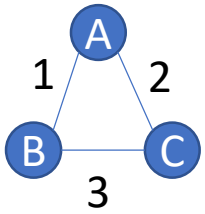
Construisons le graphe G' de sorte que nous créons un sommet pour v' pour chaque arête $e \in E$ du graphe original. Si on trouve une coloration des sommets dans G', on peut reconstruire une coloration des arêtes dans G, et vice-versa!

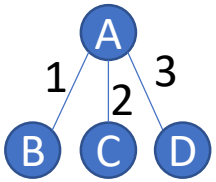
Exceptions

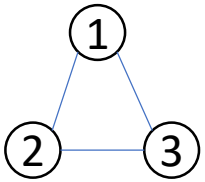
Notez que si le graphe adjoint est unique pour un graphe donné, la reconstruction du graphe initial à partir du graphe adjoint ne l'est pas toujours.

<u>Contre-exemple :</u>

Les deux graphes suivants on le même graphe adjoint :







Et la complexité ?

Décider si un graphe possède une coloration des sommets en k couleurs est un problème NP complet !

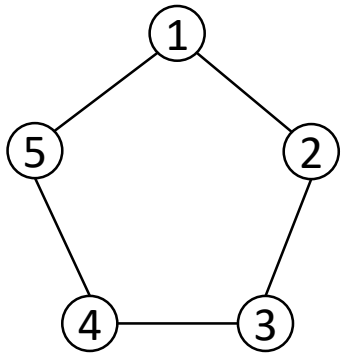
Par contre, nous pouvons donner des bornes sur le nombre chromatique $\gamma(G)$ pour un graphe $G = (V, E)$:

- $\gamma(G) \leq |V|$,
- $\gamma(G) \leq \max_{v \in \mathrm{V}}(d(v))$
- $\gamma(G) \geq \omega(G)$, où $\omega(G)$ est le nombre de nœuds dans le plus grand sous-graphe complet de G (clique de taille maximum).

Et la complexité ?

Y a-t-il égalité entre le nombre chromatique $\gamma(G)$ et la taille de la clique maximum $\omega(G)$?

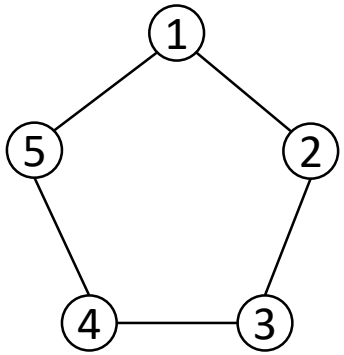
Non : contre-exemple



$\omega(G) = 2$, or $\gamma(G) = 3$ (il n'est pas possible de colorer les sommets de ce graphe avec 2 couleurs!)

Exercice

Prouvez que le graphe suivant ne peut pas être coloré avec 2 couleurs :



Exercice

Corrigé :
Notons que ce graphe est un cycle et que chaque sommet est à une distance impaire de lui-même.

Si nous construisons un chemin avec 2 couleurs en partant de n'importe que sommet, celui-ci doit alterner les 2 couleurs en questions. Donc, tous les sommets à distance impair auront une couleur, disons Rouge, et ceux à distance paire l'autre couleur disons Noir.

Or, un sommet étant à une distance impaire de lui-même, il est adjacent à un sommet à distance impair (1, le premier sommet), et un sommet pair (4, le dernier). Il est donc adjacent à deux sommets de couleur différente – il ne peut donc être d'aucune de ces deux couleurs – il faut donc au moins une troisième !