Cours de mathématiques

TOME X

INFORMATIQUE

Mathématiques générales

France $\sim$ 2024

Écrit et réalisé par Louis Lascaud

Table des matières

Chapitre 1

Analyse numérique

Résumé

L'analyse numérique étudie les méthodes programmables en algorithmique pour résoudre des problèmes mathématiques. C'est donc une discipline bel et bien mathématique à l'interface avec l'informatique théorique. Parmi les problèmes les plus étudiés, citons :

- ⋆ la réduction des systèmes linéaires ;
- ⋆ la convergence des méthodes de calcul d'une valeur d'une fonction en un point ;
- ⋆ l'interpolation polynomiale ;
- ⋆ les méthodes d'approximation géométriques ;
- ⋆ le calcul intégral informatique ;
- ⋆ la résolution d'équations différentielles par discrétisation ;
- ⋆ l'optimisation, linéaire ou non linéaire.

1.1 Résolution de systèmes linéaires

Théorème. ***(Caractérisation de la solvabilité par l'orthogonal)***

Soit A une matrice de taille $m \times n$. L'équation $Ax = b$ admet au moins une solution si et seulement si b est orthogonal à $\mathrm{Ker}(A^*)$.

Chapitre 2

Théorie de la complexité

Résumé

Dans un premie temps, on définit les concepts de base du traitement algorithmique informatique : machines de Turing, automates, etc. Dans un second temps, on introduit l'outil des circuits booléens qui facilite nombre de preuves de la section suivante. Dans un troisième temps, on introduit plusieurs classes de complexité et l'on explicite les liens entre elles, dont certains énoncés constituent des questions encore ouvertes à ce jour, parmi lesquels l'un des plus fameux problèmes du millénaire.

2.1 Machines de Turing

2.1.1 Définition et premières propriétés

En informatique théorique, une machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur. Ce modèle a été imaginé par Alan Turing en 1936, en vue de donner une définition précise au concept d'algorithme ou de « procédure mécanique ». Il est toujours largement utilisé en informatique théorique, en particulier dans les domaines de la complexité algorithmique et de la calculabilité.

À l'origine, le concept de machine de Turing, inventé avant l'ordinateur, était censé représenter une personne virtuelle exécutant une procédure bien définie, en changeant le contenu des cases d'un ruban infini, en choisissant ce contenu parmi un ensemble fini de symboles. D'autre part, à chaque étape de la procédure, la personne doit se placer dans un état particulier parmi un ensemble fini d'états. La procédure est formulée en termes d'étapes élémentaires du type : « si vous êtes dans l'état 42 et que le symbole contenu sur la case que vous regardez est « 0 », alors remplacer ce symbole par un « 1 », passer dans l'état 17, et regarder maintenant la case adjacente à droite ».

Définition. ***(Machine de Turing)***

Une *machine de Turing* ou *machine de Turing déterministe* ou *MT* est

Reformulation pratique. ***(Machine de Turing)***

Une machine de Turing est un concept abstrait composé des éléments suivants :

(i) un *ruban infini* dans les deux sens ou *bande*, éventuellement plusieurs rubans dont l'un choisi par défaut est dit *ruban de travail*, divisé en *cases* ;

(ii) un alphabet $\mathcal{A}$, dit *alphabet de travail*, souvent $\{0,1\}$, de sorte que les cases sont remplies des symboles de l'alphabet, à raison d'une lettre par case ; on suppose que $\mathcal{A} \ni B$ est non vide où B fixé est appelé *symbole blanc*, aussi souvent noté 0 ou $\#$;

(iii) une *tête de lecture*, qui *est* dans un état (on dit, mais cela n'a pas de sens précis, qu'il existe un *registre d'état* qui mémorise l'*état courant* de la machine) ;

(iv) un ensemble fini Σ d'états.

À chaque *étape* d'un *calcul*, la tête est devant une case, lit son contenu et décide de trois choses : le contenu de ladite case à la prochaine étape, son propre état pour la prochaine étape, et si elle se déplace à droite ou à gauche. Autrement dit, la tête, ou *fonction de transition*, est une application

$$\delta : A \times \Sigma \to A \times \Sigma \times \{G,D\}.$$

On note aussi $G = \leftarrow$ et $D = \rightarrow$. Si l'on n'identifie pas la tête à δ, on l'appelle *table d'actions* ou *partie de contrôle* qui est une entité non réalisée.

En particulier, le choix de l'état suivant par la tête est indépendant de sa position, mais dépend a priori et du symbole de la case lue et de son propre état actuel. On suppose qu'il existe un état $q_0 \in \Sigma$ dit *état initial* dans lequel la machine se trouve au départ, *i.e.* à la première étape. De plus, on supose qu'il existe un état $h \in \Sigma$ dit *halte* de sorte que si la machine arrive à cet état, elle s'*arrête*. La suite des étapes peut être finie ou infinie.

Dans certains livres, δ est une fonction et non une application et l'image point de $(a,s) \in A \times \Sigma$ hors du domaine de définition est définie dans la définition précédente par (a,h,G) où G est fixé tout à fait arbitrairement.

On peut également fixé $\Sigma \subseteq \mathcal{A}$ un *alphabet d'entrée* constitué des symboles dans les cases à la première étape ne contenant pas nécessairement B, et un ensemble $F \subseteq \Sigma$ d'*états acceptants* ou *finals* ou *finaux* constitué des états qui peuvent précéder une halte selon δ.

⟶ *Convention.* Les cases du ruban dont on ne précise pas quel symbole de l'alphabet elles contiennent, contiennent par défaut le symbole blanc. En particulier toutes les cases sont remplies.

VOC L'*entrée* de la machine est la case où elle *commence*, c'est-à-dire où elle se trouve à la première étape. Elle n'est pas intrinsèque à la machine, ou, selon une convention annexe, le ruban à la première étape : dans ce cas, c'est que le ruban n'a qu'un nombre fini de symbole nons blancs et à la première étape, la tête de la machine se trouve sur le premier symbole non blanc. Parfois, on impose qu'une case contenant B ne peut être une entrée, mais c'est bénin. La *sortie* de

la machine est le symbole devant la tête quand la machine s'arrête, ou, selon une convention annexe, le ruban à la halte. Elle dépend uniquement de l'entrée pour une machine de Turing déterministe.

⟶ *Notation.* δ peut facilement être représentée sous la forme d'un tableau à deux entrées quitte à colorier d'une couleur ou d'une autre les cases selon qu'elles mènent à droite où à gauche.

Remarque. Lorsque le contexte le permet, ce qui est souvent le cas, on se permet quelques abus quant à l'argument ou à la valeur de δ en considérant des projections, par exemple.

Définition. ***(Configuration d'une machine de Turing)***

La *configuration* (d'une machine de Turing à une étape) du calcul consiste en les symboles dans les cases du ruban à cette étape, l'état de la tête à cette étape et la position de la tête à cette étape.

Heuristique

La notion de machine de Turing est trop générale pour être facilement manipulable.

2.1.2 Description

Une machine de Turing peut se voir de nos jours comme l'abstraction d'un ordinateur. Dans une machine de Turing,

- ⋆ la partie de contrôle représente un microprocesseur ;
- ⋆ la bande représente la mémoire interne de l'ordinateur ;
- ⋆ la tête de lecture représente la bus *(et l'on comprend pourquoi différencier abstraitement tête de lecture et partie de contrôle).*

2.1.3 Lecture seule, lecture unique

Définition. ***(Lecture seule)***

Un ruban d'une machine de Turing est dit *en lecture seule* si la machine ne peut pas changer le contenu des cases, autrement dit si $\pi_A \circ \delta = \pi_A$.

Définition. ***(Lecture unique)***

Un ruban d'une machine de Turing est dit *en lecture unique* si après avoir lu une case du ruban, la machine doit se déplacer à droite, autrement dit si $\mathrm{Im}(\pi_{\{G,D\}} \circ \delta) = \{D\}$.

2.1.4 Temps de calcul

⟶ *Notation.* On note $\{0,1\}^* = \bigcup_{n=1}^{\infty} \{0,1\}^n$ l'ensemble des suites finies de 0 et de 1. Si $x \in \{0,1\}^*$, on note $|x|$ la *longueur* de x qui est l'unique $n \in \mathbb{N}^*$ tel que $x \in \{0,1\}^n$.

Définition. ***(Fonction booléenne)***

Une *fonction booléenne* est une fonction $\phi : \{0,1\}^* \to \{0,1\}$.

Définition. ***(Calcul d'une fonction par une machine)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing. On dit que T *calcule* ϕ si pour chaque $x \in \{0,1\}^*$, si x est l'entrée de T, $\phi(x)$ est la sortie.

VOC On dit aussi qu'on *calcule* $\varphi(x)$ si la machine à entrée x a pour sortie $\varphi(x)$.

Définition. ***(Temps de calcul)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing calculant ϕ. Soit $x \in \{0,1\}^*$. Le *temps de calcul* de $\phi(x)$ par T est le nombre d'étapes de T avant que T ne s'arrête.

Le temps de calcul peut être infini.

2.1.5 Turing-complétude

2.1.6 Machine de Turing non déterministe

Définition. ***(Machine de Turing non déterministe)***

Une *machine de Turing non déterministe* ou *non déterminée* ou *MTND* est une machine avec deux fonctions de transitions, par exemple δ_0 et δ_1.

Définition. ***(Calcul d'une fonction par une machine non déterministe)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing non déterministe. On dit que T *calcule* ϕ si pour chaque $x \in \{0,1\}^*$, alors

(i) $\phi(x) = 1$ si et seulement si l'on peut trouver une suite de configurations $C_0,\ldots,C_m, m \in \mathbb{N}$, où C_0 est la configuration initiale avec entrée x, de sorte que l'état de T dans C_m est sa halte, sa sortie est 1, et pour tout $i \in [\![1,m]\!]$, $C_i = \delta_0(C_{i-1})$ ou $\delta_1(C_{i-1})$;

(ii) $\phi(x) = 0$ si et seulement si l'on peut trouver une suite de configurations

C_0,...,C_m, $m \in \mathbb{N}$, où C_0 est la configuration initiale avec entrée x, de sorte que l'état de T dans C_m est sa halte, sa sortie est 0, et pour tout $i \in [\![1,m]\!]$, $C_i = \delta_0(C_{i-1})$ ou $\delta_1(C_{i-1})$.

Définition. ***(Temps de calcul par une machine non déterministe)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing non déterministe calculant ϕ. Soit $x \in \{0,1\}^*$. Le *temps de calcul* de $\phi(x)$ par T est la longueur minimale d'une suite de configurations du point (i) de la définition précédente si $\phi(x) = 1$ et du point (ii) si $\phi(x) = 0$.

Remarque. Cette définition est cohérente avec la définition du cas déterministe : si $\delta_0 = \delta_1$, on retrouve le calul d'une fonction booléenne par une machine de Turing.

2.1.7 Machine prenant conseil

2.1.8 Automates

2.1.9 Machine de Turing universelle

2.1.10 Réalisation d'une machine de Turing

2.2 Circuits booléens

2.2.1 Définitions

Définition. ***(Circuit booléen)***

Un *circuit booléen* est un graphe orienté acyclique fini, dont les sommets sont étiquetés par $\wedge$ (une *porte ET*), $\vee$ (une *porte OU*) ou $\neg$ (une *porte NON*), sauf les sommets de degré d'entrée nul au sens des graphes orientés, qui sont les *entrées*. Les sommets de degré de sortie nul sont les *sorties*. On impose de plus qu'un sommet étiqueté par $\neg$ a toujours un unique fils.
La *taille* du circuit booléen est le nombre de sommets, soit de portes et d'entrées réunies.

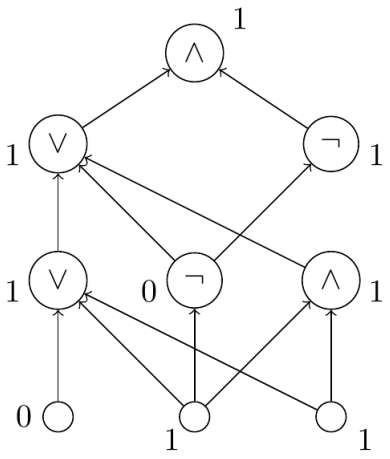

FIGURE 2.2.1 : *Exemple de circuit booléen.* —
On a de plus étiqueté les sommets pour associer au circuit une application booléenne selon la méthode décrite ci-dessous.

Méthode. ***(Application pseudo-booléenne associée à un circuit booléen)***

À un circuit booléen à n entrées rangées dans un certain ordre et m sorties, on peut associer une application $\varphi : \{0,1\}^n \to \{0,1\}^m$ défini récursivement : si les entrées sont $v_1,...,v_n$, pour calculer un certain $\phi(x)$ où $x = (x_1,...,x_n)$, on donne la valeur x_i en v_i, en respectant les règles suivantes.

1. La valeur d'une porte ET est 1 si et seulement si tous ses fils[a] ont la valeur 1.
2. La valeur d'une porte OU est 1 si et seulement s'il a aun moins un fils de valeur 1.
3. Une porte NON a un seul enfant, et sa valeur est l'opposée de la valeur de l'enfant.

En particulier, à un circuit booléen à m sorties, on peut associer une fonction booléenne. Réciproquement, si ϕ est booléenne, on dit qu'un circuit booléen *calcule* ϕ si ϕ est l'application booléenne associée à ce circuit.

[a] On rappelle que dans un graphe orienté $s_1 \to s_2$, s_1 est le fils de s_2.

Lemme

Chaque application $\phi : \{0,1\}^n \to \{0,1\}$ peut être calculée par un circuit booléen de taille inférieure à $2^{n+1} + n + 1$.

⊛ *(Idée de la preuve.)* Soit $X = \{x \in \{0,1\}^n \mid \phi(x) = 1\}$. Pour chaque $x \in X$, il existe un circuit dont l'unique sortie est étiquetée par 1 si et seulement si l'entrée est x. Si les sommets d'entrée sont $v_1,...,v_n$, si $x_i = 0$, on attache une porte NON à v_i, puis l'on attache une porte ET à toutes ses portes NON et aux sommets v_i tel que $x_i = 1$. On fait cela pour chaque $x \in X$, et l'on attache enfin une porte OU à toutes ses portes ET et c'est terminé. ■

2.3 Classes de complexité usuelles

2.3.1 Classes en temps

Principe. ***(Classes de complexité (temporelle))***

Une *classe de complexité* est un sous-ensemble de l'ensemble des fonctions booléennes caractérisé par l'existence d'une machine de Turing (dépendant de la fonction) qui la calcule en un temps de calcul majoré (dépendant de la fonction, indépendant de la machine). Cette majoration doit être en ordre de grandeur indépendante de la fonction.

2.3.1.1 La classe P

Définition. ***(Classe de complexité* P*)***

La *classe de complexité* P est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $p(|x|)$. Autrement dit, toute entrée de ϕ se calcule par une TM T en temps polynomial indépendant de la machine et de l'entrée.

Exemples. ***(Problèmes de classe* P*)***

1. Étant donné un graphe fini et deux sommets de ce graphe, pour $k \in \mathbb{N}$, le problème d'existence d'un chemin de longueur k sur ce graphe entre ces deux sommets, est encodable par une machine de Turing et se calcule en temps polynomial.

2.3.1.2 La classe NP

Définition. ***(Classe de complexité* NP*)***

La *classe de complexité* NP est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing non nécessairement déterministe T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $p(|x|)$. Autrement dit, toute entrée de ϕ se calcule par une TM ou une TMND T en temps polynomial indépendant de la machine et de l'entrée.

Exemples. ***(Problèmes de classe* NP*)***

1. Le problème d'existence d'un cycle hamiltonien dans un graphe est de classe NP.

Fait. ***(Inclusion* P $\subseteq$ NP*)***

P $\subseteq$ NP.

Conjecture ?

P = NP.

Heuristique

On croit que la hiérarchie polynomiale est une vraie hiérarchie, mais en général, il est très difficile de montrer qu'une classe de complexité est strictement incluse dans une autre. Au-delà du problème très célèbre de savoir si oui ou non P = NP, on ignore encore si l'égalité est infirmée pour des classes de complexité beaucoup plus faibles.

Théorème. ***(Définition équivalente de la classe*** NP***)***

Une fonction booléenne $\phi \in$ NP si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et une application $\psi \in$ P telle que $\forall x \in \{0,1\}^* \quad \phi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$.

▷ Si ϕ peut être calculée en temps polynomial par une machine de Turing non déterministe, pour chaque x on peut choisir une suite $\varepsilon_1,...,\varepsilon_m$ où $m \in \mathbb{N}, m \leqslant p(|x|)$ telle que si l'on choisit les fonctions de transition $\delta_{\varepsilon_1},...,\delta_{\varepsilon_m}$, on calcule $\phi(x)$. Alors, soit ψ l'application qui calcule $\phi(x)$ étant donnés x et $\varepsilon_1,...,\varepsilon_m$. Il est clair que $\psi \in P$.

Réciproquement, si une telle ψ existe, on peut utiliser une machine de Turing non déterministe pour écrire y et calculer $\psi(x,y)$. ■

La quantification introduite dans cette caractérisation rend intuitive la description de plusieurs autres classes de complexité. Par exemple la

Définition. ***(Classe de complexité*** co-NP***)***

Une fonction booléenne $\phi \in$ co-NP si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et une application $\psi \in$ P telle que $\forall x \in \{0,1\}^* \quad \phi(x) = 1 \iff \forall y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$.

Reformulation pratique. ***(Calcul de*** co-NP***)***

co-NP $= 1 - NP = \{1 - \phi, \phi \in \text{NP}\}$.

Cette autre définition n'est donc pas du tout symétrique à la première !

Conjecture ?

NP = co-NP.

2.3.1.3 La classe PH

On souhaite généraliser les trois constructions précédentes P, NP et co-NP. Pour définir la classe PH, on a besoin de sous-classes définies de manière imbriquées, qui sont les classes $(\Sigma_n)_{n\in\mathbb{N}}$ et $(\Pi_n)_{n\in\mathbb{N}}$.

On définit par récurrence :

- ⋆ $\Sigma_0 = \Pi_0 = \mathrm{P}$;
- ⋆ $\Sigma_1 = \mathrm{NP}$, $\Pi_1 = \text{co-NP}$;
- ⋆ soit $i \in \mathbb{N}$. On caractérise : $\phi : \Sigma_{i+1}$ s'il existe $\psi \in \Pi_i$ telle qu'il existe p polynomiale telle que $\forall x \quad \phi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$; de même, $\phi \in \Pi_{i+1}$ s'il existe $\psi \in \Sigma_i$ telle qu'il existe p polynomiale telle que $\forall x \quad \phi(x) = 1 \iff \forall y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$.

Définition. ***(Classe de complexité* PH*)***

On définit $\mathrm{PH} = \bigcup_{n=0}^{\infty} (\Sigma_n \cup \Pi_n)$.

Fait. ***(La classe* PH *est volumineuse)***

$\mathrm{P} \subseteq \mathrm{PH}$, $\mathrm{NP} \subseteq \mathrm{PH}$ et $\text{co-NP} \subseteq \mathrm{PH}$.

2.3.1.4 La classe L

On veut définir la classe des problèmes n'utilisant que des ressources « en quantité logarithmique ». Problème, une telle restriction telle quelle est évidemment trop forte : simplement l'écriture de l'entrée dans un ruban n'est pas un problème résolu en temps logarithmique, mais linéaire ! Il faut donc y aller plus subtilement.

Définition. ***(Classe de complexité* L*)***

La *classe de complexité* L est l'ensemble des fonctions booléennes telles que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing T à deux rubans dont le premier est à lecture seule, contient pour entrée x, et le deuxième est le ruban de travail, de sorte que pour chaque x, la machine ne regarde que $\mathcal{O}(\log(|x|))$ de cases du ruban de travail et calcule $\phi(x)$.

Exercice 1

On dit qu'une fonction booléenne $\phi \in \mathrm{LP}$ si pour tout $x \in \{0,1\}^*$, il existe une machine de Turing à deux rubans dont le premier est à lecture seule, contient pour entrée x, et le deuxième est de travail et tel que pour chaque x, la machine ne regarde que $\mathcal{O}(\log(p(|x|)))$ de cases du ruban de travail et calcule $\phi(x)$. Montrer que $\mathrm{LP} = \mathrm{L}$.

Heuristique

C'est la complexité minimale raisonnable.

L'inclusion extrêmement grossière suivante est peut-être une égalité, car on ne sait pas si

Conjecture ?

$\mathrm{L} \subsetneq \mathrm{PH}$.

Mnémonik : on ne peut pas prouver que les miracles n'existent pas...

Définition. ***(Classe de complexité*** NL***)***

La *classe de complexité* NL est l'ensemble des fonctions booléennes telles que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing non déterministe T à deux rubans dont le premier est à lecture seule, contient pour entrée x, et le deuxième est le ruban de travail, de sorte que pour chaque x, la machine ne regarde que $\mathcal{O}(\log(|x|))$ de cases du ruban de travail et calcule $\phi(x)$.

On voudrait dire : *"une fonction booléenne $\phi \in$ NP si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et une application $\psi \in$ L telle que $\phi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$"*, mais ce n'est pas le cas.

Exercice 2

Démontrer l'affirmation précédente.

On peut bel et bien caractériser, de même qu'avec P et NP, la classe NL à partir de L, mais il faut être plus fin.

Théorème. ***(Définition équivalente de la classe*** NL***)***

Une fonction booléenne $\phi \in \mathrm{NL}$ si et seulement si pour tout $x \in \{0,1\}^*$, il existe une machine de Turing à trois rubans dont le premier est à lecture seule et contient l'entrée x, le deuxième est à lecture unique et le troisième est un ruban de travail, dont la machine n'utilise qu'une quantité logarithmique de cases, avec : $\phi(x) = 1$ si et seulement s'il existe

une variable y et ψ calculée par une machine de Turing telle que si x et dans le premier ruban et y est dans le deuxième, alors la sortie de T est $\varphi(x)$.

⊛ *(Idée de la preuve.)* C'est un peu plus subtil et laissé aux lecteurs plus aiguisés. ■

2.3.1.5 La classe P/poly

Définition. ***(Classe de complexité*** P/poly***)***

Une fonction booléenne $\phi \in \mathrm{P/poly}$ si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et pour chaque $n \in \mathbb{N}$, une suite finie $(y_i) \in \{0,1\}^{p(n)}$ et une application $\psi_n \in \mathrm{P}$ telle que $\forall x \in \{0,1\}^* \quad \phi(x) = \psi(x, y_n)$. On appelle une telle $(y_i)_{i \in [\![1, p(n)]\!]}$ une *suite de conseil*.

2.3.2 Classes en espace

2.3.2.1 Les classes EXPTIME et NEXPTIME

Définition. ***(Classe de complexité*** EXPTIME***)***

La *classe de complexité* EXPTIME est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $\exp(p(|x|))$.

Définition. ***(Classe de complexité*** NEXPTIME***)***

La *classe de complexité* NEXPTIME est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing non nécessairement déterministe T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $\exp(p(|x|))$.

2.3.2.2 La classe PSPACE

Définition. ***(Classe de complexité*** PSPACE***)***

La *classe de complexité* PSPACE est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p et une machine de Turing T qui calcule ϕ tels que pour tout $x \in \{0,1\}^*$, la position de T pendant le calcul est comprise entre $-p(|x|)$ et $p(|x|)$ à translation près du ruban, ou ce qui est équivalent (pourquoi ?), dans un intervalle de cases de longueur $p(|x|)$.

Exemple fondamental. ***(Problème de classe*** PSPACE***)***

Le problème de détermination du gagnant au jeu de go, si les joueurs utilisent des stratégies optimales, n'est pas PH mais est PSPACE.

Chapitre 3

Exercices

Difficulté des exercices :

- ● ○ ○ ○ ○ Question de cours, application directe, exercice purement calculatoire sans réelle difficulté technique
- ● ● ○ ○ ○ Exercice faisable, soit intuitivement, soit en employant des moyens rudimentaires ou des techniques déjà vues
- ● ● ● ○ ○ Exercice relativement difficile et dont la résolution appelle à une réflexion plus importante à cause d'obstacles techniques ou conceptuels, qui cependant devraient être à la portée de la plupart des étudiants bien entraînés
- ● ● ● ● ○ Exercice très exigeant, destiné aux élèves prétendant aux concours les plus difficiles, exercice « classique ».
- ● ● ● ● ● La résolution de l'exercice requiert un raisonnement et des connaissances extrêmement avancés, dépassant les attentes du prérequis. Il est presque impossible de le mener à terme sans indication. Bien qu'exigibles à très peu d'endroits, ces exercices sont très intéressants et présentent souvent des résultats forts.

Appendice

Bibliographie

[1] *Titre du livre*, Auteur du livre, date, maison d'édition

Table des figures

Liste des tableaux