Algorithmique

Cours avec 957 exercices et 158 problèmes

Algorithmique

Cours avec 957 exercices et 158 problèmes

Thomas H. Cormen
Professeur d'informatique au Dartmouth College

Charles E. Leiserson
Professeur d'informatique au MIT

Ronald L. Rivest
Professeur d'informatique et d'électrotechnique au MIT

Clifford Stein
Professeur de génie industriel et de recherche opérationnelle
à l'université Columbia

3e édition

DUNOD

L'édition originale de ce livre a été publiée aux États-Unis par The MIT Press, Cambridge, Massachusetts, sous le titre *Introduction to Algorithms*, third edition.

Illustration de couverture : *Glowing abstract wave background in red*
© wenani-Fotolia.com

© Dunod, Paris, 1994, 2004, 2010
ISBN 978-2-10-054526-1

Préface à l'édition française

Vous savez compter. Un ordinateur aussi ! Mais connaissez-vous les mécanismes utilisés ? Êtes-vous vraiment sûr que le résultat affiché soit juste ? Combien de temps devrez-vous attendre la fin du calcul ? N'y a-t-il pas un moyen de l'obtenir plus vite ? Que vous soyez ingénieur, mathématicien, physicien, statisticien et surtout informaticien, toutes ces questions vous vous les posez. Si vous êtes étudiant, elles surgiront très rapidement.

Étudier l'algorithmique, c'est apporter des réponses à vos questions.

Cette science est le cœur de l'informatique. Pour tout ceux qui doivent ou devront faire travailler un ordinateur, il est essentiel de comprendre ses principes fondamentaux et de connaî tre ses éléments de base. Une formule 1 ne se conduit pas comme une voiture à pédales. De même, un ordinateur se s'utilise pas comme un boulier. L'algorithmique est le permis de conduire de l'informatique. Sans elle, il n'est pas concevable d'exploiter sans risque un ordinateur.

Cette introduction remarquable à l'algorithmique donne au lecteur, d'une part, les bases théoriques indispensables et lui fournit, d'autre part, les moyens de concevoir rigoureusement des programmes efficaces permettant de résoudre des problèmes variés, issus de différentes applications.

L'éventail des algorithmes présentés va des plus classiques, comme les algorithmes de tri et les fonctions de hachage, aux plus récents comme ceux de la cryptographie. On trouve ici rassemblés des algorithmes numériques, par exemple pour l'inversion de matrices ou la transformée de Fourier, et des algorithmes combinatoires comme les algorithmes de graphes ou la recherche de motif.

Une très large place est faite aux structures de données, des plus simples comme les listes, aux plus sophistiquées comme les tas de Fibonacci. Notons au passage l'importance accordée aux différentes mesures de complexité (pire des cas, amortissement, en moyenne) qui permettent d'approfondir entre autres l'étude de l'efficacité des algorithmes de tri et des structures de données.

Il est certain que la plupart des informaticiens spécialisés trouveront dans ce livre leurs algorithmes de base, exprimés de façon unifiée, ainsi que certaines avancées récentes dans leur domaine. Cet ouvrage met donc en relief le rôle central joué par l'algorithmique dans la science Informatique.

La présence de chapitres méthodologiques comme ceux consacrés à la programmation dynamique et aux algorithmes gloutons, ainsi que les deux derniers qui traitent de la complexité des problèmes et de la conception d'algorithmes approchés, permet au lecteur d'amorcer une réflexion plus poussée sur la manière d'aborder un problème et de concevoir une méthode de résolution. Ces chapitres sont illustrés par des exemples d'application bien choisis, et là encore très divers.

Cette gamme de sujets, riche par sa variété et ses niveaux de difficulté, est soutenue par une pédagogie constante, qui rend la lecture de l'ouvrage facile et agréable, sans nuire à l'exigence de rigueur. À titre d'exemple, on peut citer la clarté remarquable des chapitres consacrés aux graphes, qui, à partir d'un algorithme générique, introduisent toute une famille de variantes efficaces dont les différentes implémentations sont analysées avec finesse.

D'une manière générale, les notions présentées sont systématiquement introduites de façon informelle à partir d'un exemple ou d'une application particulière, avant d'être formalisées. Les propriétés et les algorithmes sont toujours démontrés. La présence d'une partie consacrée aux fondements mathématiques utilisés est tout à fait bienvenue et rend l'ouvrage accessible avec très peu de prérequis.

Le lecteur peut facilement se familiariser et approfondir les notions rencontrées grâce aux nombreux exercices de difficulté graduée. Les problèmes permettent d'aller plus loin dans la compréhension du chapitre et sont souvent une occasion de connaî tre différentes applications pratiques des algorithmes présentés. De ce point de vue, ce livre est une mine d'or pour tout enseignant d'algorithmique.

La lecture de cet ouvrage est tout à fait recommandée aux étudiants de second et de troisième cycle d'informatique et de mathématiques, ainsi qu'aux élèves ingénieurs. Tous y trouveront une aide et un support de cours utile tout au long de leurs études.

La diversité des sujets abordés, l'efficacité des algorithmes présentés et leur écriture dans un pseudo-code proche des langages *C* et Pascal, qui les rend très faciles à implémenter, font aussi de ce livre un recueil fort utile dans la vie professionnelle d'un informaticien ou d'un ingénieur.

Enfin, au-delà de ses besoins propres, nous souhaitons que le lecteur, qu'il soit ingénieur, étudiant, ou simplement curieux, prenne comme nous plaisir et intérêt à la lecture de cet ouvrage.

Paris, mars 1994

PHILIPPE CHRÉTIENNE, CLAIRE HANEN,
ALIX MUNIER, CHRISTOPHE PICOULEAU
Université Pierre et Marie Curie (LIP6)

À propos de la seconde édition :

C'est avec une curiosité renouvelée que l'enseignant, le chercheur ou l'étudiant abordera cette seconde édition du livre de référence de l'algorithmique. L'algorithmicien déjà familier de la précédente édition y trouvera, parmi moult enrichissements disséminés tout le long de l'ouvrage, de nouvelles parties, notamment celle dédiée à la programmation linéaire, de nombreux exercices et problèmes inédits, ainsi qu'une présentation uniformisée des preuves d'algorithmes ; le lecteur, étudiant ou enseignant, qui découvre cette *Introduction à l'algorithmique*, y trouvera sous un formalisme des plus limpides, le nécessaire, voire un peu plus, de cette discipline qui est l'une des pierres angulaires de l'informatique.

Paris, août 2002 CHRISTOPHE PICOULEAU,
Laboratoire CEDRIC CNAM

À propos de la troisième édition :

Depuis bientôt deux décennies *Introduction à l'algorithmique* constitue la référence internationale pour le chercheur et l'enseignement en algorithmique. C'est donc avec le plus vif intérêt que l'étudiant, l'enseignant, le chercheur francophone prendra en main cette nouvelle édition de l'ouvrage.

Les principales nouveautés de cette édition consistent en trois nouveaux chapitres, de nouveaux exercices et exemples introductifs, et un formalisme allégé pour l'écriture du pseudo-code. Le lecteur est invité à la lecture de la préface pour y découvrir une description détaillée des modifications apportées pour cette dernière édition.

Concernant les choix de traduction effectués, comme pour les deux éditions précédentes, notre objectif a été d'employer exclusivement la langue française. Cependant, l'algorithmique est une discipline en évolution permanente. Si certains termes s'officialisent dans la communauté francophone nous les avons entérinés ici. Pour d'autres, plus rares, correspondant aux thématiques les plus récentes abordées ici, nous avons préféré avoir recours au vocabulaire couramment utilisé par les spécialistes ; ainsi il nous a semblé préférable de conserver le terme anglais *multithread* plutôt que l'expression française moins consensuelle *processus léger*. La prochaine édition de *Algorithmique* sera certainement l'occasion d'affermir nos choix en matière de vocabulaire.

Paris, avril 2010 CHRISTOPHE PICOULEAU,
Laboratoire CEDRIC CNAM

Préface

Les algorithmes existaient avant les ordinateurs. Mais maintenant qu'il y a des ordinateurs, il y a encore plus d'algorithmes et les algorithmes sont au cœur de l'informatique.

Nous proposons dans ce livre une introduction complète à l'étude contemporaine des algorithmes informatiques. De nombreux algorithmes y sont présentés et étudiés en détail, de façon à rendre leur conception et leur analyse accessibles à tous les niveaux de lecteurs. Nous avons essayé de maintenir la simplicité des explications, sans sacrifier la profondeur de l'étude ni la rigueur mathématique.

Chaque chapitre présente un algorithme, une technique de conception, un domaine d'application ou un sujet s'y rapportant. Les algorithmes sont décrits en français et dans un pseudo-code conçu pour être lisible par quiconque ayant déjà programmé. Le livre contient 244 figures, dont beaucoup se composent de plusieurs parties, qui illustrent le fonctionnement des algorithmes. Comme nous mettons l'accent sur *l'efficacité* comme critère de conception, les temps d'exécution de tous nos algorithmes sont soigneusement analysés.

Ce texte est en premier lieu un support du cours d'algorithmique ou de structures de données de deuxième ou troisième cycle universitaire. Il est également bien adapté à la formation personnelle des professionnels, puisqu'il s'intéresse aux problèmes d'ingénierie ayant trait à la conception d'algorithmes, ainsi qu'à leurs aspects mathématiques.

Dans cette édition, qui est la troisième, nous avons de nouveau modifié l'ensemble du livre. Les changements vont de la simple amélioration du style jusqu'à l'ajout de nouveaux chapitres en passant par l'amélioration du pseudo-code.

a) Pour l'enseignant

Ce livre se veut à la fois complet et polyvalent. Il se révélera utile pour toutes sortes de cours, depuis un cours de structures de données en deuxième cycle jusqu'à un cours d'algorithmique en troisième cycle. Un cours trimestriel étant beaucoup trop court pour aborder tous les sujets étudiés ici, on peut voir ce livre comme un « buffet garni » où vous pourrez choisir le matériel le mieux adapté aux cours que vous souhaitez enseigner.

Vous trouverez commode d'organiser votre cours autour des chapitres dont vous avez vraiment besoin. Nous avons fait en sorte que les chapitres soient relativement indépendants, de manière à éviter toute subordination inattendue ou superflue d'un chapitre à l'autre. Chaque chapitre commence en présentant des notions simples et se poursuit avec les notions plus difficiles, le découpage en sections créant des points de passage naturels. Dans un cours de deuxième cycle, on pourra ne faire appel qu'aux premières sections d'un chapitre donné ; en troisième cycle, on pourra traiter le chapitre en entier.

Nous avons inclus **957 exercices** et **158 problèmes**. Chaque section se termine par des exercices et chaque chapitre se termine par des problèmes. Les exercices sont généralement des questions courtes de contrôle des connaissances. Certains servent surtout à tester la compréhension du sujet ; d'autres, plus substantiels, sont plutôt du genre devoir à la maison. Les problèmes sont des études de cas plus élaborées qui introduisent souvent de nouvelles notions ; ils sont composés le plus souvent de plusieurs questions qui guident l'étudiant à travers les étapes nécessaires pour parvenir à une solution.

Les sections et exercices munis d'une astérisque ($\star$) recouvrent des thèmes destinés plutôt aux étudiants de troisième cycle. Un passage étoilé n'est pas forcément plus ardu qu'un passage non étoilé, mais risque d'exiger des connaissances mathématiques plus pointues. De même, un exercice étoilé risque de demander un niveau théorique plus important ou d'incorporer des subtilités au-dessus de la moyenne.

b) Pour l'étudiant

Nous espérons que ce livre vous fournira une introduction agréable à l'algorithmique. Nous avons essayé de rendre chaque algorithme accessible et intéressant. Pour vous aider lorsque vous rencontrez des algorithmes peu familiers ou difficiles, nous les avons tous décrits étape par étape. Nous expliquons aussi avec soin les notions mathématiques nécessaires pour comprendre l'analyse des algorithmes. Si vous êtes déjà familiarisé avec un sujet, vous constaterez que l'organisation des chapitres vous permet de sauter les sections d'introduction et d'aller rapidement aux concepts plus avancés.

Ceci est un livre volumineux, et votre cours n'en couvrira sans doute qu'une partie. Nous avons pourtant essayé de faire un livre qui vous servira aussi bien maintenant comme support de cours, que plus tard dans votre carrière, comme référence mathématique ou manuel d'ingénierie.

Quels sont les pré-requis pour lire ce livre ?

– Vous devez avoir une expérience de la programmation. En particulier, vous devez comprendre les procédures récursives et les structures de données simples comme les tableaux et les listes chaînées.
– Vous devez être familiarisé avec les démonstrations mathématiques, surtout avec le raisonnement par récurrence. Certaines parties de ce livre reposent sur des connaissances de calcul élémentaire. Cela dit, les parties 1 et 8 de ce livre vous apprendront toutes les techniques mathématiques dont vous aurez besoin.

Nous avons tenu compte de la demande concernant les solutions des problèmes et des exercices. La page de notre site (`http://www.dunod.com/`) dédiée à cet ouvrage contient des liens vers des solutions à quelques-uns des problèmes et exercices.

c) Pour le professionnel

La grande variété de sujets présents dans ce livre en fait un excellent manuel de référence sur les algorithmes. Chaque chapitre étant relativement indépendant des autres, vous pourrez vous concentrer sur les sujets qui vous intéressent le plus.

La plupart des algorithmes étudiés ont une grande utilité pratique. Nous mettons donc l'accent sur l'implémentation et autres problèmes d'ingénierie. Nous offrons le plus souvent des alternatives pratiques aux quelques algorithmes qui sont surtout d'intérêt théorique.

Si vous souhaitez implémenter l'un de ces algorithmes, vous n'aurez aucun mal à traduire notre pseudo-code dans votre langage de programmation favori. Le pseudo-code est conçu pour présenter chaque algorithme de façon claire et succincte. Nous ne nous intéressons donc pas à la gestion des erreurs et autres problèmes de génie logiciel qui demandent des hypothèses particulières sur l'environnement de programmation. Nous essayons de présenter chaque algorithme simplement et directement de manière à éviter que son essence soit masquée par les idiosyncrasies d'un langage de programmation particulier.

Nous comprenons bien que, si vous employez ce livre en dehors d'un cours, vous ne pourrez pas comparer vos solutions aux problèmes et exercices avec les solutions fournies par un enseignant.
La page de notre site (`http://www.dunod.com/`) dédiée à cet ouvrage contient des liens vers des solutions à certains des problèmes et exercices, ce qui vous permettra de contrôler votre travail.

d) Pour nos collègues

Nous donnons une bibliographie très complète et des références à la littérature courante. Chaque chapitre se termine par une partie « Notes » qui fournit des détails et des références historiques. Les notes de chapitre ne fournissent pas, toutefois, une référence complète au vaste champ des algorithmes. Malgré la taille imposante de cet ouvrage, nous avons dû renoncer, faute de place, à inclure nombre d'algorithmes intéressants.

Nonobstant les innombrables supplications émanant d'étudiants, nous avons décidé de ne pas fournir de solutions à tous les problèmes et exercices ; ainsi, l'étudiant ne succombera pas à la tentation de regarder la solution au lieu d'essayer de la trouver par lui-même.

e) Modifications apportées à la troisième édition

Qu'est-ce qui a changé par rapport à la deuxième édition de ce livre ? Le volume des modifications est comparable au volume des modifications de la deuxième édition par rapport à la première. Comme nous l'avions dit alors, on trouvera que le livre a bien changé ou peu changé, selon le point de vue adopté.

Un examen sommaire de la table des matières montre que la plupart des chapitres et des sections de la deuxième édition figurent aussi dans la troisième. Nous avons supprimé deux chapitres et une section, mais nous avons ajouté trois nouveaux chapitres et deux nouvelles sections réparties sur d'autres chapitres.

Voici un résumé des changements les plus significatifs pour la troisième édition :

– Nous avons ajouté deux nouveaux chapitres, consacrés aux arbres de van Emde Boas et aux algorithmes multithread, et regroupé dans une annexe tout ce qui concerne les notions de base sur les matrices.

– Nous avons refondu le chapitre sur les récurrences, afin de traiter de manière plus large la technique diviser-pour-régner, et ses deux premières sections appliquent cette technique pour résoudre deux problèmes. La deuxième section de ce chapitre présente l'algorithme de Strassen pour la multiplication matricielle, que nous avons ôté du chapitre traitant du calcul matriciel.

– Nous avons supprimé deux chapitres, qui étaient rarement enseignés : les tas binomiaux et les réseaux de tri. Un concept fondamental du chapitre sur les réseaux de tri, le principe du zéro-un, apparaît dans cette édition dans le problème 8.7, et ce sous la forme du lemme de tri 0-1 pour algorithmes de comparaison-permutation. Le traitement des tas de Fibonacci ne s'appuie plus sur les tas binomiaux.

– Nous avons refondu notre traitement de la programmation dynamique et des algorithmes gloutons. La programmation dynamique débute désormais par un problème plus intéressant, la découpe de barres, que le problème d'ordonnancement de chaîne de montage présenté dans la deuxième édition. En outre, nous mettons davantage l'accent sur la mémoïsation et nous introduisons la notion de graphe de sous-problème comme façon de comprendre la durée d'exécution d'un algorithme de programmation dynamique. Dans notre exemple d'introduction aux algorithmes gloutons, le problème du choix d'activités, nous allons à l'algorithme glouton plus directement que dans la deuxième édition.

– La façon dont nous supprimons un nœud dans les arbres de recherche binaires (ce qui inclut les arbres rouge-noir) garantit désormais que le nœud que l'on veut supprimer est bien le nœud qui est supprimé. Dans les deux premières éditions, dans certains cas, il y avait suppression d'un autre nœud, dont le contenu était transféré dans le nœud transmis à la procédure de suppression. Avec notre nouvelle façon de supprimer les nœuds, si d'autres composants d'un programme gèrent des pointeurs vers les nœuds de l'arbre, ils ne se retrouveront pas avec des pointeurs obsolètes vers des nœuds ayant été supprimés.

– L'étude des réseaux de transports fait désormais reposer entièrement les flux sur les arcs. Cette approche est plus intuitive que le flot employé dans les deux premières éditions.

– Suite au transfert vers d'autres chapitres des fondamentaux sur les matrices et de l'algorithme de Strassen, le chapitre consacré au calcul matriciel est plus court que dans la deuxième édition.
– Nous avons modifié notre traitement de l'algorithme de comparaison de chaîne de Knuth-Morris-Pratt.
– Plusieurs erreurs ont été corrigées, dont la plupart avaient été publiées sur notre site Web consacré aux errata de la seconde édition.
– Suite à moult demandes, nous avons modifié la syntaxe de notre pseudo-code. Nous employons maintenant "=" pour indiquer l'affectation et "==" pour tester l'égalité, comme le font C, C++, Java et Python. De même, nous avons supprimé les mots clé **faire** et **alors** et adopté "**//**" comme symbole pour commentaire de fin de ligne. En outre, nous utilisons la notation pointée pour indiquer les attributs d'objet. Notre pseudo-code reste procédural, au lieu d'être orienté objet. Autrement dit, au lieu d'exécuter des méthodes sur des objets, nous nous contentons d'appeler des procédures en leur passant des objets comme paramètres.
– Nous avons ajouté 100 nouveaux exercices et 28 nouveaux problèmes. Nous avons aussi actualisé nombre de références bibliographiques et en avons ajouté plusieurs.
– Enfin, nous avons revu et corrigé des phrases, des paragraphes et des sections dans tout le livre, afin qu'il soit plus lisible et plus compréhensible.

f) Site web

Vous pouvez utiliser notre site web, *http://mitpress.mit.edu/algorithms/*, pour obtenir des informations supplémentaires en anglais et communiquer avec nous. Le site offre des liens vers une liste d'erreurs connues, vers les solutions d'exercices et de problèmes en anglais, etc. Le site explique aussi comment faire pour nous signaler les erreurs et nous proposer des suggestions.

g) Comment nous avons créé ce livre

Comme la deuxième édition, la troisième édition a été créée en LaTeX 2_ε. Nous avons utilisé la police Times avec les caractères mathématiques MathTime Pro 2. Nous remercions Michael Spivak de Publish or Perish, Inc., Lance Carnes de Personal TeX, Inc. et Tim Tregubov de Dartmouth College pour leur assistance technique. Comme dans les deux éditions précédentes, nous avons compilé l'index avec Windex, un programme C écrit par nous, et créé la bibliographie avec BibTeX. Les fichiers PDF de cet ouvrage ont été créés sur un MacBook exécutant OS 10.5.

Les images de la troisième édition ont été faites avec MacDraw Pro, certaines des expressions mathématiques figurant sur les images ayant été ajoutées à l'aide du paquetage psfrag pour LaTeX 2_ε. Malheureusement, MacDraw Pro est un logiciel hérité, qui n'est plus vendu depuis plus de 10 ans. Heureusement, nous avons encore deux Macintosh capables d'exécuter l'environnement Classic sous OS 10.4 et donc capables d'exécuter à peu près MacDraw Pro. Même sous l'environnement Classic, nous trouvons MacDraw Pro bien plus facile à

utiliser que n'importe quel autre logiciel de dessin pour les types d'illustrations qui accompagnent les textes informatiques, sans compter qu'il produit des résultats magnifiques.(1) Qui sait combien de temps vont encore durer nos Macs pré Intel ? Si donc quelqu'un de chez Apple nous écoute : *S'il vous plaît, créez une version compatible OS X de MacDraw Pro !*

h) Remerciements pour la troisième édition

Nous travaillons avec MIT Press depuis plus de 20 ans, pour notre plus grande satisfaction ! Merci à Ellen Faran, Bob Prior, Ada Brunstein et Mary Reilly pour leur aide et leur assistance.

Nous étions éparpillés géographiquement pour créer la troisième édition, travaillant au Dartmouth College Department of Computer Science, au MIT Computer Science and Artificial Intelligence Laboratory et au Columbia University Department of Industrial Engineering and Operations Research. Nous remercions nos universités et collègues respectifs de nous avoir fourni des environnements aussi favorables et stimulants.

Julie Sussman, P.P.A., a fait une fois de plus un excellent travail de correction technique. Nous avons été parfois stupéfaits de voir des erreurs qui nous avaient échappé être détectées par Julie. Elle nous a aussi aidé à améliorer notre présentation à plusieurs endroits. S'il existe un mémorial des correcteurs techniques, Julie mérite sans contestation d'y figurer. C'est rien moins qu'un phénomène. Merci, merci, merci, Julie ! Priya Natarajan a également trouvé des erreurs, que nous avons pu corriger avant la mise sous presse. Toutes les erreurs susceptibles de figurer encore dans ce livre sont de la responsabilité des auteurs (et ont sans douté été insérées après que Julie a lu le texte).

Le traitement des arbres de van Emde Boas dérive des notes d'Erik Demaine, qui elles-mêmes ont été influencées par Michael Bender. Nous avons aussi incorporé des idées de Javed Aslam, Bradley Kuszmaul et Hui Zha dans cette édition.

Le chapitre sur le multithread s'appuie sur des notes initialement écrites en collaboration avec Harald Prokop. Ces matériaux ont été influencés par plusieurs autres participants au projet Cilk au MIT, dont Bradley Kuszmaul et Matteo Frigo. La conception du pseudo-code multithread s'inspire des extensions MIT Cilk pour C et des extensions Cilk Arts's Cilk++ pour C++.

Nous remercions également les nombreux lecteurs des deux premières éditions qui ont signalé des erreurs ou soumis des suggestions visant à améliorer le livre. Nous avons corrigé toutes les erreurs *bona fide* signalées et tenu compte du maximum de suggestions. Le nombre de ces contributeurs est devenu si grand qu'il n'est malheureusement plus possible de les citer tous.

(1) Nous avons étudié plusieurs logiciels de dessin fonctionnant sous Mac OS X, mais tous présentent des lacunes significatives comparé à MacDraw Pro. Nous avons brièvement essayé de produire les images de ce livre avec un autre programme de dessin bien connu. Nous avons constaté qu'il fallait au moins cinq fois plus de temps pour produire chaque image qu'avec MacDraw Pro et que les résultats n'étaient pas aussi beaux. D'où le choix de revenir à MacDraw Pro exécuté sur de vieux Macintosh.

Nous remercions enfin nos épouses, Nicole Cormen, Wendy Leiserson, Gail Rivest et Rebecca Ivry, et nos enfants, Ricky, Will, Debby et Katie Leiserson ; Alex et Christopher Rivest ; et Molly, Noah et Benjamin Stein, pour leur amour et leur aide pendant que nous écrivions cet ouvrage. La patience et l'encouragement de nos familles a rendu ce projet possible. Nous leur dédions donc ce livre avec affection.

THOMAS H. CORMEN — *Lebanon, New Hampshire*
CHARLES E. LEISERSON — *Cambridge, Massachusetts*
RONALD L. RIVEST — *Cambridge, Massachusetts*
CLIFFORD STEIN — *New York, New York*

Février 2009

Table des matières

PRÉFACE À L'ÉDITION FRANÇAISE V

PRÉFACE IX

PARTIE 1 • INTRODUCTION

CHAPITRE 1 • **RÔLE DES ALGORITHMES EN INFORMATIQUE** 3

1.1 Algorithmes 3
Exercices 8

1.2 Algorithmes en tant que technologie 9
Exercices 11
PROBLÈMES 12

CHAPITRE 2 • **PREMIERS PAS** 13

2.1 Tri par insertion 13
Exercices 19

2.2 Analyse des algorithmes 19
Exercices 25

2.3 Conception des algorithmes 26
Exercices 34
PROBLÈMES 35

CHAPITRE 3 • **CROISSANCE DES FONCTIONS** 39

3.1 Notation asymptotique 40
Exercices 48

3.2 Notations standard et fonctions classiques 49
Exercices 55
PROBLÈMES 55

CHAPITRE 4 • **DIVISER POUR RÉGNER** 59

4.1 Le problème du sous-tableau maximal 62
Exercices 68

4.2 Algorithme de Strassen pour la multiplication matricielle 69
Exercices 75

4.3 La méthode de substitution pour la résolution des récurrences 76
Exercices 80

4.4 La méthode de l'arbre récursif pour la résolution des récurrences 81
Exercices 85

4.5 La méthode générale pour la résolution des récurrences 86
Exercices 89

4.6 Démonstration du théorème général 89
Exercices 97

PROBLÈMES 98

CHAPITRE 5 • **ANALYSE PROBABILISTE ET ALGORITHMES RANDOMISÉS** 105

5.1 Le problème de l'embauche 105
Exercices 108

5.2 Variables aléatoires indicatrices 109
Exercices 112

5.3 Algorithmes randomisés 113
Exercices 118

5.4 Analyse probabiliste et autres emplois des variables aléatoires indicatrices 120
Exercices 131

PROBLÈMES 132

PARTIE 2 • TRI ET RANGS

CHAPITRE 6 • **TRI PAR TAS** 139

6.1 Tas 140
Exercices 141

6.2 Conservation de la propriété de tas 142
Exercices 144

6.3 Construction d'un tas 144
Exercices 147

6.4 Algorithme du tri par tas 147
Exercices 148

6.5 Files de priorités 149
Exercices 152
PROBLÈMES 153

CHAPITRE 7 • **TRI RAPIDE** 157

7.1 Description du tri rapide 157
Exercices 161
7.2 Performances du tri rapide 161
Exercices 164
7.3 Une version randomisée du tri rapide 165
Exercices 166
7.4 Analyse du tri rapide 166
Exercices 170
PROBLÈMES 171

CHAPITRE 8 • **TRI EN TEMPS LINÉAIRE** 177

8.1 Minorants pour le tri 177
Exercices 179
8.2 Tri par dénombrement 180
Exercices 181
8.3 Tri par base 182
Exercices 184
8.4 Tri par paquets 185
Exercices 188
PROBLÈMES 189

CHAPITRE 9 • **MÉDIANS ET RANGS** 197

9.1 Minimum et maximum 198
Exercices 199
9.2 Sélection en temps espéré linéaire 199
Exercices 203
9.3 Sélection en temps linéaire dans le cas le plus défavorable 203
Exercices 206
PROBLÈMES 207

PARTIE 3 • STRUCTURES DE DONNÉES

CHAPITRE 10 • **STRUCTURES DE DONNÉES ÉLÉMENTAIRES** 215

10.1 Piles et files 215
Exercices 218

10.2 Listes chaînées 219
Exercices 222

10.3 Implémentation de pointeurs et d'objets 223
Exercices 227

10.4 Représentation des arbres 227
Exercices 229

PROBLÈMES 230

CHAPITRE 11 • **TABLES DE HACHAGE** 235

11.1 Tables à adressage direct 236
Exercices 237

11.2 Tables de hachage 238
Exercices 242

11.3 Fonctions de hachage 243
Exercices 249

11.4 Adressage ouvert 250
Exercices 257

11.5 Hachage parfait 258
Exercices 262

PROBLÈMES 262

CHAPITRE 12 • **ARBRES BINAIRES DE RECHERCHE** 267

12.1 Qu'est-ce qu'un arbre binaire de recherche ? 268
Exercices 269

12.2 Requête dans un arbre binaire de recherche 270
Exercices 273

12.3 Insertion et suppression 274
Exercices 278

12.4 Arbres binaires de recherche construits aléatoirement 279
Exercices 282

PROBLÈMES 283

CHAPITRE 13 • **ARBRES ROUGE-NOIR** 287

13.1 Propriétés des arbres rouge-noir 287
Exercices 290

13.2 Rotations 291
Exercices 292

13.3 Insertion 293
Exercices 300

13.4 Suppression 301
Exercices 307

PROBLÈMES 308

CHAPITRE 14 • **EXTENSION DES STRUCTURES DE DONNÉES** 315

14.1 Rangs dynamiques 316
Exercices 320

14.2 Comment étendre une structure de données 321
Exercices 323

14.3 Arbres d'intervalles 324
Exercices 328

PROBLÈMES 329

PARTIE 4 • TECHNIQUES AVANCÉES DE CONCEPTION ET D'ANALYSE

CHAPITRE 15 • **PROGRAMMATION DYNAMIQUE** 333

15.1 découpe de barres 334
Exercices 343

15.2 Multiplications matricielles enchaînées 343
Exercices 350

15.3 Éléments de programmation dynamique 351
Exercices 361

15.4 Plus longue sous-séquence commune 362
Exercices 367

15.5 Arbres binaires de recherche optimaux 368
Exercices 374

PROBLÈMES 375

CHAPITRE 16 • **ALGORITHMES GLOUTONS** 383

16.1 Un problème de choix d'activités 384
Exercices 390

16.2 Éléments de la stratégie gloutonne 391
Exercices 395

16.3 Codages de Huffman 396
Exercices 403

16.4 Matroïdes et méthodes gloutonnes 404
Exercices 409

16.5 Un problème d'ordonnancement de tâches en tant que matroïde 410
Exercices 412

PROBLÈMES 412

CHAPITRE 17 • **ANALYSE AMORTIE** 417

17.1 Analyse de l'agrégat 418
Exercices 422

17.2 Méthode comptable 422
Exercices 424

17.3 Méthode du potentiel 424
Exercices 427

17.4 Tables dynamiques 428
Exercices 436

PROBLÈMES 437

PARTIE 5 • STRUCTURES DE DONNÉES AVANCÉES

CHAPITRE 18 • **B-ARBRES** 447

18.1 Définition d'un B-arbre 451
Exercices 453

18.2 Opérations fondamentales sur les B-arbres 453
Exercices 460

18.3 Suppression d'une clé dans un B-arbre 461
Exercices 464

PROBLÈMES 464

CHAPITRE 19 • **TAS DE FIBONACCI** 467

19.1 Structure des tas de Fibonacci 469

19.2 Opérations sur les tas fusionnables 471
Exercices 479

19.3 Diminution d'une clé et suppression d'un nœud 479
Exercices 483

19.4 Borne du degré maximal 483
Exercices 486

PROBLÈMES 486

CHAPITRE 20 • **ARBRES DE VAN EMDE BOAS** 491

20.1 Approches préliminaires 492
Exercices 495

20.2 Une structure récursive 496
Exercices 503

20.3 L'arbre de van Emde Boas 504
Exercices 514

PROBLÈMES 514

CHAPITRE 21 • **STRUCTURES DE DONNÉES POUR ENSEMBLES DISJOINTS** 519

21.1 Opérations sur les ensembles disjoints 519
Exercices 522

21.2 Représentation d'ensembles disjoints par des listes chaînées 522
Exercices 525

21.3 Forêts d'ensembles disjoints 526
Exercices 529

21.4 Analyse de l'union par rang avec compression de chemin 530
Exercices 537

PROBLÈMES 538

PARTIE 6 • ALGORITHMES POUR LES GRAPHES

CHAPITRE 22 • **ALGORITHMES ÉLÉMENTAIRES POUR LES GRAPHES** 545

22.1 Représentation des graphes 545
Exercices 548

22.2 Parcours en largeur 549
Exercices 556

22.3 Parcours en profondeur 557
Exercices 564

22.4 Tri topologique 566
Exercices 568

22.5 Composantes fortement connexes 568
Exercices 572

PROBLÈMES 573

CHAPITRE 23 • **ARBRES COUVRANTS MINIMAUX** 577

23.1 Construction d'un arbre couvrant minimal 578
Exercices 582

23.2 Algorithmes de Kruskal et de Prim 583
Exercices 588

PROBLÈMES 589

CHAPITRE 24 • **PLUS COURTS CHEMINS À ORIGINE UNIQUE** 595

24.1 Algorithme de Bellman-Ford 602
Exercices 605

24.2 Plus courts chemins à origine unique dans les graphes orientés sans circuit 606
Exercices 608

24.3 Algorithme de Dijkstra 609
Exercices 613

24.4 Contraintes de différence et plus courts chemins 614
Exercices 618

24.5 Démonstrations des propriétés de plus court chemin 620
Exercices 625

PROBLÈMES 626

CHAPITRE 25 • **PLUS COURTS CHEMINS ENTRE TOUTES PAIRES DE SOMMETS** 631

25.1 Plus courts chemins et multiplication de matrices 633
Exercices 637

25.2 L'algorithme de Floyd-Warshall 639
Exercices 645

25.3 Algorithme de Johnson pour les graphes peu denses 646
Exercices 650

PROBLÈMES 650

CHAPITRE 26 • **FLOT MAXIMUM** 653

26.1 Réseaux de flot 654
Exercices 657

26.2 La méthode de Ford-Fulkerson 658
Exercices 672

26.3 Couplage biparti maximum 673
Exercices 676

26.4 Algorithmes pousser-réétiqueter 677
Exercices 686

26.5 Algorithme réétiqueter-vers-l'avant 688
Exercices 698

PROBLÈMES 698

PARTIE 7 • MORCEAUX CHOISIS

CHAPITRE 27 • **ALGORITHMES MULTITHREAD** 709

27.1 Fondements du multithread dynamique 711
Exercices 726

27.2 Multiplication matricielle multithread 727
Exercices 731

27.3 Tri par fusion multithread 732
Exercices 738

PROBLÈMES 739

CHAPITRE 28 • **CALCUL MATRICIEL** 747

28.1 Résolution de systèmes d'équations linéaires 747
Exercices 760

28.2 Inversion des matrices 760
Exercices 764

28.3 Matrices symétriques définies positives et approximation par la méthode des moindres carrés 765
Exercices 770

PROBLÈMES 771

CHAPITRE 29 • **PROGRAMMATION LINÉAIRE** 775

29.1 Forme standard et forme canonique 782
Exercices 788

29.2 Formulation de problèmes comme programmes linéaires 789
Exercices 794
29.3 Algorithme du simplexe 795
Exercices 807
29.4 Dualité 808
Exercices 814
29.5 La solution réalisable de base initiale 814
Exercices 820
PROBLÈMES 822

CHAPITRE 30 • **POLYNÔMES ET TRANSFORMÉE DE FOURIER RAPIDE** 827
30.1 Représentation des polynômes 829
Exercices 834
30.2 Transformée discrète de Fourier et transformée rapide de Fourier 835
Exercices 842
30.3 Implémentations efficaces de la transformée rapide de Fourier 842
Exercices 847
PROBLÈMES 847

CHAPITRE 31 • **ALGORITHMES DE LA THÉORIE DES NOMBRES** 853
31.1 Notions élémentaires de la théorie des nombres 854
Exercices 858
31.2 Plus grand commun diviseur 860
Exercices 864
31.3 Arithmétique modulaire 865
Exercices 870
31.4 Résolution d'équations linéaires modulaires 871
Exercices 874
31.5 Théorème du reste chinois 875
Exercices 877
31.6 Puissances d'un élément 878
Exercices 881
31.7 Le cryptosystème à clés publiques RSA 881
Exercices 887
31.8 Test de primarité 888
Exercices 896
31.9 Factorisation des entiers 897
Exercices 901

PROBLÈMES 902

CHAPITRE 32 • **RECHERCHE DE CHAÎNES DE CARACTÈRES** 905

32.1 Algorithme naïf de recherche de chaîne de caractères 908
Exercices 909

32.2 Algorithme de Rabin-Karp 910
Exercices 914

32.3 Recherche de chaîne de caractères au moyen d'automates finis 915
Exercices 921

32.4 Algorithme de Knuth-Morris-Pratt 921
Exercices 929

PROBLÈMES 930

CHAPITRE 33 • **GÉOMÉTRIE ALGORITHMIQUE** 933

33.1 Propriétés des segments de droite 934
Exercices 938

33.2 Déterminer si deux segments donnés se coupent 940
Exercices 945

33.3 Recherche de l'enveloppe convexe 946
Exercices 955

33.4 Recherche des deux points les plus rapprochés 956
Exercices 959

PROBLÈMES 960

CHAPITRE 34 • **NP-COMPLÉTUDE** 965

34.1 Temps polynomial 970
Exercices 977

34.2 Vérification en temps polynomial 977
Exercices 980

34.3 NP-complétude et réductibilité 982
Exercices 991

34.4 Preuves de NP-complétude 992
Exercices 998

34.5 Problèmes NP-complets 999
Exercices 1012

PROBLÈMES 1013

CHAPITRE 35 • **ALGORITHMES D'APPROXIMATION** 1017

35.1 Problème de la couverture de sommets 1019
Exercices 1022

35.2 Problème du voyageur de commerce 1022
Exercices 1027

35.3 Problème de la couverture d'ensemble 1027
Exercices 1032

35.4 Randomisation et programmation linéaire 1032
Exercices 1036

35.5 Problème de la somme de sous-ensemble 1037
Exercices 1042

PROBLÈMES 1042

ANNEXES • ÉLÉMENTS DE MATHÉMATIQUES

ANNEXE A • **SOMMES** 1051

A.1 Formules et propriétés des sommes 1051
Exercices 1055

A.2 Bornes des sommes 1055
Exercices 1061

PROBLÈMES 1061

ANNEXE B • **ENSEMBLES, ETC.** 1063

B.1 Ensembles 1063
Exercices 1067

B.2 Relations 1067
Exercices 1069

B.3 Fonctions 1070
Exercices 1071

B.4 Graphes 1072
Exercices 1076

B.5 Arbres 1076
Exercices 1081

PROBLÈMES 1082

ANNEXE C • **DÉNOMBREMENT ET PROBABILITÉS** 1085

C.1 Dénombrement 1085
Exercices 1089

C.2 Probabilités 1091
Exercices 1096

C.3 Variables aléatoires discrètes 1097
Exercices 1100

C.4 Distributions géométrique et binomiale 1102
Exercices 1106

C.5 Queues de la distribution binomiale 1107
Exercices 1113

PROBLÈMES 1114

ANNEXE D • **MATRICES** 1115

D.1 Matrices et opérations matricielles 1115
Exercices 1120

D.2 Propriétés fondamentales des matrices 1120
Exercices 1123

PROBLÈMES 1124

BIBLIOGRAPHIE 1127

INDEX 1151

PARTIE 1

INTRODUCTION

Cette partie présente les fondamentaux qui doivent vous guider pour concevoir et analyser des algorithmes. Elle a pour objectif d'exposer en douceur la manière de spécifier les algorithmes, certaines stratégies de conception qui nous serviront tout au long de ce livre, ainsi que bon nombre des concepts essentiels sous-jacents à l'analyse des algorithmes. Les parties suivantes de ce livre s'appuieront sur toutes ces fondations.

Le chapitre 1 donne une présentation générale des algorithmes et de leur place dans les systèmes informatiques modernes. Ce chapitre définit ce qu'est un algorithme et donne des exemples. Il signale aussi que les algorithmes sont une technologie, au même titre que les matériels, les interfaces utilisateur graphiques, les systèmes orientés objet ou les réseaux.

Au chapitre 2, nous verrons nos premiers algorithmes, lesquels résolvent le problème qui consiste à trier une suite de n nombres. Ils seront écrits en un pseudo-code qui, même s'il n'est pas directement traduisible en quelque langage de programmation traditionnel que ce soit, reflète la structure de l'algorithme de manière suffisamment claire pour que vous puissiez le mettre en œuvre dans le langage de votre choix. Les algorithmes de tri que nous étudierons sont le tri par insertion, qui utilise une stratégie incrémentale, et le tri par fusion, qui utilise une technique récursive baptisée « diviser pour régner ». Bien que la durée d'exécution de chacun de ces algorithmes croisse avec la valeur de n, le taux de croissance n'est pas le même pour les deux algorithmes. Nous déterminerons ces temps d'exécution au chapitre 2 et introduirons une notation très pratique pour les exprimer.

Le chapitre 3 définira plus formellement cette notation, que nous appellerons notation asymptotique. Il commencera par définir plusieurs notations asymptotiques qui nous serviront à borner, à majorer et/ou à minorer, les durées d'exécution des algorithmes. Le reste du

chapitre 3 consistera essentiellement en une présentation de notations mathématiques. Le but recherché est de garantir que les notations que vous employez concordent avec celles utilisées dans cet ouvrage, et non de vous enseigner de nouveaux concepts mathématiques.

Le chapitre 4 approfondira la méthode diviser-pour-régner, introduite au chapitre 2. Il fournira des exemples supplémentaires d'algorithmes diviser-pour-régner, dont la surprenante méthode de Strassen pour la multiplication de deux matrices carrées. Ce chapitre 4 donnera des méthodes pour la résolution des récurrences, qui sont très utiles pour décrire les durées d'exécution des algorithmes récursifs. Une technique très puissante ici est celle appelée « méthode générale », que nous utilisons souvent pour résoudre les récurrences induites par les algorithmes diviser-pour-régner. Une bonne partie du chapitre 4 sera consacrée à la démonstration de la justesse de la méthode générale ; vous pouvez, sans inconvénient aucun, sauter cette démonstration.

Le chapitre 5 introduira l'analyse probabiliste et les algorithmes randomisés. L'analyse probabiliste sert généralement à déterminer la durée d'exécution d'un algorithme dans le cas où, en raison de la présence d'une distribution probabiliste intrinsèque, le temps d'exécution peut varier pour différentes entrées de la même taille. Dans certains cas, nous supposerons que les entrées obéissent à une distribution probabiliste connue, de sorte que nous ferons la moyenne des temps d'exécution sur l'ensemble des entrées possibles. Dans d'autres cas, la distribution probabiliste ne viendra pas des entrées, mais de choix aléatoires faits pendant l'exécution de l'algorithme. Un algorithme dont le comportement dépend non seulement de l'entrée, mais aussi de valeurs produites par un générateur de nombres aléatoires est un algorithme randomisé. Nous pouvons employer des algorithmes randomisés pour garantir une distribution probabiliste sur les entrées, et ainsi garantir qu'aucune entrée particulière n'entraîne systématiquement de piètres performances, voire même pour borner les taux d'erreur des algorithmes qui sont autorisés à donner, dans une certaine limite, des résultats erronés.

Les annexes A–D renferment d'autres sujets mathématiques qui vous faciliteront la lecture de cet ouvrage. Vous avez certainement déjà vu une bonne partie de ces notions (bien que les définitions et conventions de notation spécifiques que nous utilisons ici puissent, à l'occasion, différer de ce que vous connaissiez) ; vous pouvez donc considérer les annexes comme une sorte de référence. En revanche, la plupart des concepts présentés dans la partie 1 sont vraisemblablement inédits pour vous. Tous les chapitres de la partie 1 et toutes les annexes ont été rédigés dans un esprit très didactique.

Chapitre **1**

Rôle des algorithmes en informatique

Qu'est-ce qu'un algorithme ? En quoi l'étude des algorithmes est-elle utile ? Quel est le rôle des algorithmes par rapport aux autres technologies informatiques ? Ce chapitre a pour objectif de répondre à ces questions.

1.1 ALGORITHMES

Voici une définition informelle du terme **algorithme** : procédure de calcul bien définie qui prend en **entrée** une valeur, ou un ensemble de valeurs, et qui donne en **sortie** une valeur, ou un ensemble de valeurs. Un algorithme est donc une séquence d'étapes de calcul qui transforment l'entrée en sortie.

L'on peut aussi considérer un algorithme comme un outil permettant de résoudre un **problème de calcul** bien spécifié. L'énoncé du problème spécifie, en termes généraux, la relation désirée entre l'entrée et la sortie. L'algorithme décrit une procédure de calcul spécifique permettant d'obtenir cette relation entrée/sortie.

Supposons, par exemple, qu'il faille trier une suite de nombres dans l'ordre croissant. Ce problème, qui revient fréquemment dans la pratique, offre une base fertile pour l'introduction de nombre de techniques de conception et d'outils d'analyse standard. Voici comment nous définissons formellement le **problème de tri** :

Entrée : Une suite de n nombres $\langle a_1, a_2, ..., a_n \rangle$.

Sortie : Une permutation (réorganisation) $\langle a'_1, a'_2, ..., a'_n \rangle$ de la suite donnée en entrée, de façon que $a'_1 \leqslant a'_2 \leqslant \cdots \leqslant a'_n$.

Ainsi, à partir de la suite $\langle 31, 41, 59, 26, 41, 58 \rangle$, un algorithme de tri produit en sortie la suite $\langle 26, 31, 41, 41, 58, 59 \rangle$. À propos de la suite donnée en entrée, on parle d'**instance** du problème de tri. En général, une **instance d'un problème** consiste en l'entrée (satisfaisant aux contraintes, quelles qu'elles soient, imposées dans l'énoncé du problème) requise par le calcul d'une solution au problème.

Le tri est une opération majeure en informatique (maints programmes l'emploient comme phase intermédiaire), ce qui explique que l'on ait inventé un grand nombre d'algorithmes de tri. L'algorithme optimal pour une application donnée dépend, entre autres facteurs, du nombre d'éléments à trier, de la façon dont les éléments sont plus ou moins triés initialement, des restrictions potentielles concernant les valeurs des éléments, de l'architecture de l'ordinateur, ainsi que du type de périphérique de stockage à utiliser : mémoire principale, disques ou même bandes.

Un algorithme est dit **correct** si, pour chaque instance en entrée, il se termine en produisant la bonne sortie. L'on dit qu'un algorithme correct **résout** le problème donné. Un algorithme incorrect risque de ne pas se terminer pour certaines instances en entrée, voire de se terminer sur une réponse autre que celle désirée. Contrairement à ce que l'on pourrait croire, un algorithme incorrect peut s'avérer utile dans certains cas, si son taux d'erreur est susceptible d'être contrôlé. Nous en verrons un exemple au chapitre 31, quand nous étudierons des algorithmes servant à déterminer les grands nombres premiers. En général, seuls nous intéresseront toutefois les algorithmes corrects.

Un algorithme peut être spécifié en langage humain ou en langage informatique, mais peut aussi être basé sur un système matériel. L'unique obligation est que la spécification fournisse une description précise de la procédure de calcul à suivre.

a) Quels sont les types de problème susceptibles d'être résolus par des algorithmes ?

Le tri n'est absolument pas l'unique problème pour lequel ont été inventés des algorithmes. (Vous l'avez sans douté deviné rien qu'en voyant la taille de cet ouvrage.) Les applications concrètes des algorithmes sont innombrables, entre autres :

– Le projet du génome humain a bien avancé vers ses objectifs qui sont d'identifier les 100 000 gènes de l'ADN humain, de déterminer les séquences des 3 milliards de paires de bases chimiques qui constituent l'ADN humain, de stocker ces informations dans des bases de données et de développer des outils d'analyse de données. Chacune de ces étapes exige des algorithmes très élaborés. Les solutions aux divers problèmes sous-jacents sortent du cadre de ce livre, mais les concepts traités dans nombre de chapitres de cet ouvrage sont utilisés pour résoudre ces problèmes de biologie, permettant ainsi aux scientifiques de faire leur travail tout en utilisant les ressources avec efficacité. Cela fait gagner du temps, aussi bien au niveau des hommes que des machines, et de l'argent, du fait que les expériences de laboratoire peuvent ainsi fournir davantage d'informations.

– Internet permet à des gens éparpillés un peu partout dans le monde d'accéder rapidement à toutes sortes de données. Les sites Internet reposent sur des algorithmes intelligents qui leur permettent de gérer et manipuler de grosses masses de données. Exemples de

problèmes faisant un usage essentiel d'algorithmes : recherche de routes optimales pour l'acheminement des données (ce genre de technique sera présenté au chapitre 24) ; utilisation d'un moteur de recherche pour trouver rapidement les pages contenant tel ou tel type de données (les techniques afférentes seront vues aux chapitres 11 et 32).

– Le commerce électronique, qui permet de négocier et échanger, de manière électronique, biens et services, exige que l'on préserve la confidentialité de données telles que numéros de carte de crédit, mots de passe et relevés bancaires. La cryptographie à clé publique et les signatures numériques (traitées au chapitre 31), qui font partie des technologies fondamentales employées dans ce contexte, s'appuient sur des algorithmes numériques et sur la théorie des nombres.

– Dans l'industrie et le commerce, il faut souvent optimiser l'allocation de ressources limitées. Une compagnie pétrolière veut savoir où placer ses puits de façon à maximiser les profits escomptés. Un candidat à la présidence veut savoir dans quels supports publicitaires il doit investir pour maximiser ses chances d'élection. Une compagnie aérienne désire réaliser l'affectation des équipages aux vols de telle façon que les coûts soient minimisés, les vols assurés sans défaillance et la législation respectée. Un fournisseur de services Internet veut savoir où placer des ressources supplémentaires pour desservir ses clients de manière plus efficace. Voilà des exemples de problèmes susceptibles d'être résolus par la programmation linéaire, que nous étudierons au chapitre 29.

Certains détails de ces exemples sortent du cadre de cet ouvrage, mais nous donnerons des techniques qui s'appliquent à ces problèmes et catégories de problème. Nous montrerons aussi, dans ce livre, comment résoudre maints problèmes concrets, dont les suivants :

– Soit une carte routière sur laquelle sont indiquées toutes les distances entre intersections adjacentes ; il faut déterminer le trajet le plus court entre deux intersections. Le nombre d'itinéraires peut être énorme, même si l'on n'a pas d'itinéraire qui se recoupe. Comment trouver le trajet le plus court parmi tous les trajets possibles ? Ici, nous modélisons la carte (qui, elle-même, modélise les routes réelles) sous la forme d'un graphe (sujet traité dans la partie VI et dans l'annexe B), puis nous cherchons à déterminer le chemin le plus court entre deux sommets du graphe. Le chapitre 24 montrera comment résoudre ce problème de manière efficace.

– Soient deux suites ordonnées de symboles, $X = \langle x_1, x_2, \ldots, x_m \rangle$ et $Y = \langle y_1, y_2, \ldots, y_n \rangle$, pour lesquelles nous voulons trouver une sous-suite commune de longueur maximale. Une sous-suite de X n'est autre que X après suppression de certains (ou tous ou aucun) de ses éléments. Par exemple, une sous-suite de $\langle A, B, C, D, E, F, G \rangle$ est $\langle B, C, E, G \rangle$. La longueur d'une sous-suite commune maximale de X et Y donne une indication de la similitude entre ces deux suites. Par exemple, si les deux suites sont des paires de bases dans des nœuds ADN, nous pourrions les considérer comme similaires si elles ont une longue sous-séquence commune. Si X possède m symboles et Y n symboles, alors X et Y possèdent 2^m et 2^n sous-séquences possibles, respectivement. La sélection de toutes les sous-séquences possibles de X et Y et leur comparaison risquent de prendre un temps prohibitif sauf si m et n sont très petits. Le chapitre 15 montrera comment employer une

technique générale dite programmation dynamique pour résoudre ce problème beaucoup plus efficacement.

– Soit un schéma mécanique défini par une bibliothèque de pièces détachées, où chaque pièce peut contenir des instances d'autres pièces, pour lequel nous devons énumérer les pièces de telle façon que chaque pièce apparaisse avant toutes les autres pièces qui l'utilisent. Si le schéma se compose de n pièces, il existe $n!$ ordres possibles, où $n!$ désigne la factorielle. Comme la fonction factorielle croît encore plus vite qu'une fonction exponentielle, il n'est pas pratique de générer tous les ordres possibles puis de vérifier que, au sein de cet ordre, chaque pièce apparaît avant les pièces l'utilisant (sauf si le nombre de pièces est très petit). Ce problème est une instance du tri topologique et le chapitre 22 montrera comment résoudre efficacement ce problème.
– Soient n points du plan, dont nous voulons déterminer l'enveloppe convexe. Cette enveloppe est le plus petit polygone convexe qui contient les points. Intuitivement, nous pouvons nous représenter chaque point comme un clou planté dans une planche. L'enveloppe convexe serait alors représentée par un élastique qui entoure tous les clous. Chaque clou autour duquel s'enroule l'élastique est un sommet de l'enveloppe convexe. (Voir figure 33.6 pour un exemple.) N'importe lequel des 2^n sous-ensembles des points pourrait correspondre aux sommets de l'enveloppe convexe. Seulement, il ne suffit pas de savoir quels sont les points qui sont des sommets de l'enveloppe ; il faut aussi connaître l'ordre dans lequel ils apparaissent. Il existe donc bien des choix pour les sommets de l'enveloppe convexe. Le chapitre 33 donnera deux bonnes méthodes pour la détermination de l'enveloppe convexe.

Ces énumérations sont loin d'être exhaustives (comme vous l'avez probablement deviné en soupesant ce livre), mais témoignent de deux caractéristiques que l'on retrouve dans nombre de problèmes algorithmiques intéressants :

1) Il existe beaucoup de solutions candidates, mais la plupart d'entre elles ne résolvent pas le problème. Trouver une solution qui convienne, ou une qui est « la meilleure », voilà qui n'est pas toujours évident.
2) Ils ont des applications concrètes. Parmi les problèmes précédemment énumérés, la recherche du chemin le plus court fournit les exemples les plus élémentaires. Une entreprise de transport, par exemple une société de camionnage ou la SNCF, a intérêt à trouver les trajets routiers ou ferroviaires les plus courts, car cela diminue les coûts de main d'œuvre et d'énergie. Un nœud de routage Internet peut avoir à déterminer le chemin le plus court à travers le réseau pour minimiser le délai d'acheminement d'un message. Une personne désirant aller de Marseille à Nice peut avoir besoin d'indications d'itinéraire fournies par un site Web, ou bien peut se servir de son GPS tout en conduisant.

Un problème résolu par des algorithmes ne possède pas toujours un ensemble facilement identifié de solutions candidates. Supposez, par exemple, que vous ayez un ensemble donné de valeurs numériques représentant des échantillons d'un signal et que vous vouliez calculer la transformée de Fourier discrète de ces échantillons. La transformée discrète convertit le domaine des temps en domaine des fréquences, en produisant un ensemble de coefficients