Thomas H. Cormen

Algorithmes

Notions de base

DUNOD

L'édition originale de ce livre a été publiée aux États-Unis
par The MIT Press, Cambridge, Massachussets,
sous le titre *Algorithms Unlocked*.
Copyright © 2013 Massachussets Institute of Technology

Traduit de l'américain par Hervé Soulard

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.
Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.
Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).

Illustration de couverture : © Droits réservés

© Dunod, Paris, 2013 pour la traduction française
ISBN 978-2-10-070151-3

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Table des matières

© Dunod – Toute reproduction non autorisée est un délit.

© Dunod – Toute reproduction non autorisée est un délit.

AVANT-PROPOS

Comment les ordinateurs font-ils pour résoudre des problèmes ? Comment votre petit GPS parvient-il à déterminer, parmi la multitude d'itinéraires possibles, le chemin le plus rapide vers votre destination, et cela en quelques secondes ? Lors d'un achat sur Internet, comment votre numéro de carte bancaire est-il protégé contre quiconque l'intercepterait ? La réponse à ces questions, comme à de nombreuses autres, tient dans les *algorithmes*. Ce livre va vous dévoiler les mystères des algorithmes.

Je compte parmi les auteurs de l'ouvrage *Algorithmique*. Si ce livre est vraiment très bien (peut-être suis-je un tantinet partisan), il peut être trop technique sur certains points.

Celui-ci est différent. Ce n'est pas un manuel pour étudiant « classique ». Il n'aborde pas le domaine des algorithmes informatiques de façon générale ni détaillée. Il n'enseigne pas de façon normative des techniques de conception des algorithmes. Il ne propose au lecteur aucun problème ou exercice.

Dans ce cas, qu'allez-vous y trouver ?

Cet ouvrage sera un point de départ si :

- vous voulez comprendre comment les ordinateurs résolvent des problèmes ;
- vous souhaitez savoir comment évaluer la qualité de ces solutions ;
- vous aimeriez mettre en rapport les approches employées dans la résolution des problèmes informatiques et le monde extérieur ;
- vous êtes capable de lire des expressions mathématiques finalement pas si compliquées ;
- vous n'avez pas nécessairement déjà écrit un programme informatique (mais l'avoir fait ne sera pas rédhibitoire).

Certains ouvrages qui traitent des algorithmes prennent une approche conceptuelle, avec peu de détails techniques. D'autres affichent une précision technique stupéfiante. D'autres encore trouvent leur place entre ces deux extrêmes. C'est le cas de celui-ci. Oui vous y trouverez des éléments mathématiques. Oui certains points seront abordés avec une grande

précision. Cependant, je vais éviter d'aller trop loin dans les détails, excepté peut-être vers la fin de l'ouvrage, là où j'ai un peu perdu le contrôle.

Je vois ce livre comme l'entrée d'un repas. Supposons que vous alliez au restaurant et que vous commandiez l'entrée. Avant de commander la suite, vous attendez qu'elle arrive et que vous la mangiez. Si vous ne l'aimez pas, vous décidez que le repas est terminé. Peut-être vous a-t-elle plu, mais elle a suffi à vous rassasier. Ou bien vous l'avez appréciée, mais vous avez encore faim et décidez de passer à la suite. En lisant cet ouvrage, je souhaite que vous arriviez dans l'un des deux derniers cas : vous êtes satisfait mais sans avoir envie d'aller plus loin dans le monde des algorithmes, ou vous avez tellement apprécié le contenu que vous souhaitez en apprendre plus. Chaque chapitre se termine par une section intitulée « Lectures complémentaires » qui propose d'autres ouvrages et articles relatifs au sujet traité.

Objectif du livre

Je ne suis pas en mesure de vous préciser ce que vous apprendrez en lisant cet ouvrage. En revanche, voici ce que j'ai l'intention de vous enseigner :

- Ce que sont les algorithmes informatiques, une façon de les décrire et comment les évaluer.
- Des méthodes simples qui permettent de rechercher des informations sur un ordinateur.
- Des méthodes qui permettent de réorganiser les informations afin qu'elles apparaissent dans l'ordre souhaité (autrement dit les trier).
- Comment résoudre des problèmes de base dont la modélisation informatique se fait à l'aide d'une structure mathématique appelée **graphe**. Parmi leurs nombreuses applications, les graphes conviennent parfaitement à la modélisation des réseaux routiers (quelles intersections mènent à d'autres intersections par des chemins directs et quelles sont les longueurs de ces routes ?), des dépendances entre des tâches (quelle tâche doit venir avant les autres ?), des relations financières (quels sont les taux de change entre les différentes monnaies ?) ou des interactions entre des personnes (qui connaît qui ? qui n'aime pas qui ? quel acteur apparaît dans un film, au côté de quel autre ?).
- Comment résoudre des problèmes qui posent des questions sur des chaînes de texte. Certains de ces problèmes se rencontrent dans différents domaines, comme la biologie où les caractères représentent des molécules de base et les chaînes de caractères, une structure ADN.
- Les principes de base de la cryptographie. Même si vous n'avez jamais chiffré un message vous-même, il est probable que votre ordinateur l'ait déjà fait (par exemple lors de l'achat de produits en ligne).
- Les concepts fondamentaux de la compression de données.

© Dunod – Toute reproduction non autorisée est un délit.

- Comprendre que certains problèmes sont difficiles à résoudre sur un ordinateur en un temps raisonnable, ou, tout au moins, que personne ne sait encore comment procéder.

Prérequis

Je l'ai indiqué précédemment, ce livre comprend du contenu mathématique. Si cela vous effraie, vous pouvez essayer de sauter ces passages ou de vous tourner vers un ouvrage moins technique. Toutefois, j'ai fait mon possible pour que ce contenu reste accessible au plus réfractaire.

Je ne suppose pas que vous ayez déjà écrit ou même lu un programme informatique. Si vous êtes capable de suivre des instructions données sous forme d'un plan, vous serez en mesure de comprendre la façon dont j'exprime les étapes qui constituent un algorithme. Si vous comprenez la blague suivante, vous avez déjà fait un grand pas :

> *Avez-vous déjà entendu parler de cet informaticien qui était resté bloqué dans sa douche ? Il se lavait les cheveux en suivant les instructions inscrites sur la bouteille de shampooing : « savonnez, rincez, recommencez ».*

Dans cet ouvrage, j'emploie un style relativement informel en espérant qu'une approche personnelle rendra le contenu plus accessible. Certains chapitres se fondent sur les précédents, mais les dépendances sont rares. Quelques-uns débutent de manière non technique, mais le deviennent progressivement. Si vous trouvez qu'un chapitre devient trop compliqué, vous aurez probablement intérêt à lire au moins le début du suivant.

Erreurs

Si vous découvrez une erreur dans cet ouvrage, n'hésitez pas à m'en avertir par courrier électronique à l'adresse *unlocked@mit.edu*.

Remerciements

Je suis extrêmement reconnaissant à mes co-auteurs de l'ouvrage *Algorithmique*, Charles Leiserson, Ron Rivest et Cliff Stein, car une grande partie du présent ouvrage se fonde sur son contenu. Vous constaterez que j'y fais souvent effrontément référence (comprendre que j'en prends des extraits). En rédigeant cet ouvrage seul, j'ai réalisé combien la collaboration avec Charles, Ron et Cliff pouvait me manquer. Je remercie également toutes les personnes mentionnées dans l'avant-propos du CLRS (sigle utilisé dans

le monde entier pour désigner l'ouvrage *Algorithmique*, en référence aux initiales de ses quatre auteurs).

J'ai également fait appel au contenu des cours que je donne à Dartmouth, notamment ceux intitulés *Computer Science* 1, 5 et 25. Merci à mes étudiants dont les questions pertinentes m'ont permis de connaître les approches pédagogiques qui fonctionnent et dont les silences glaciaux ont révélé celles à éviter.

La rédaction de ce livre a été suggérée par Ada Brunstein, notre éditrice chez MIT Press, pendant que nous préparions la troisième édition du CLRS. Ada a depuis quitté la société et a été remplacée par Jim DeWolf. À l'origine cet ouvrage devait rejoindre la collection *Essential Knowledge*, mais il a semblé trop technique (pouvez-vous imaginer cela, j'ai écrit un livre trop technique pour le MIT !). Jim a parfaitement géré cette situation potentiellement délicate en me permettant d'écrire non pas l'ouvrage imaginé à l'origine par MIT Press mais celui que je souhaitais. J'ai également apprécié le soutien d'Ellen Faran et de Gita Devi Manaktala, toutes deux œuvrant chez MIT Press.

Julie Sussman s'était chargée de la révision technique de la deuxième et de la troisième édition du CLRS. Je suis vraiment ravi qu'elle ait pu également s'occuper de celui-ci. Elle n'a rien laissé passer. Pour preuve, voici une partie du message que Julie m'a envoyé à propos d'un premier brouillon du Chapitre 5 :

> *« Cher M. Cormen,*
>
> *Les autorités ont appréhendé un chapitre évadé qui s'était caché dans votre livre. Nous ne sommes pas en mesure de déterminer de quel ouvrage il s'était échappé, mais nous ne pouvons pas imaginer qu'il ait pu se glisser dans le vôtre pendant autant de mois sans que vous le sachiez. Nous n'avons donc pas d'autre choix que de vous tenir pour responsable. Nous espérons que vous vous chargerez de le remanier et d'en faire un bon citoyen de votre livre. Vous trouverez ci-joint le rapport rédigé par l'officier qui a procédé à l'arrestation, Julie Sussman. »*

Je ne suis pas un spécialiste de la cryptographie et le chapitre qui traite de ses principes de base a énormément bénéficié des commentaires et des suggestions de Ron Rivest, Sean Smith, Rachel Miller et Huijia Rachel Lin. Une note de bas de page fait référence aux signes utilisés dans les matches de base-ball. Je remercie Bob Whalen, l'entraîneur du club de Dartmouth, pour m'avoir patiemment expliqué certains des systèmes employés. Ilana Arbisser a vérifié que les spécialistes de bio-informatique alignent les séquences ADN de la manière dont je l'explique au Chapitre 8. Travailler au département informatique du Dartmouth College, c'est vraiment génial. Mes collègues sont brillants et fraternels, et notre équipe professionnelle

© Dunod – Toute reproduction non autorisée est un délit.

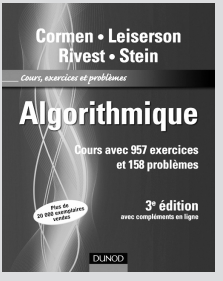
est à nulle autre pareille. Si vous recherchez un cours d'informatique de niveau premier ou second cycle, ou si vous recherchez un poste en informatique dans une université américaine, n'hésitez pas à contacter Dartmouth.

Merci enfin à mon épouse, Nicole Cormen, à mes parents, Renee et Perry Cormen, à ma sœur, Jane Maslin, et aux parents de Nicole, Colette et Paul Sage, pour leur amour et leur soutien. Mon père est certain que le dessin en page 2 est non pas un S mais un 5.

Tom Cormen, Hanover,
New Hampshire, novembre 2012

Note de l'éditeur

Nous remercions M. Christophe Picouleau, professeur au Cnam de Paris, pour sa relecture critique de la traduction du chapitre 10 sur la complexité.



L'ouvrage [CLRS09] auquel Thomas Cormen fait souvent allusion dans cet ouvrage est son cours de référence dont il existe une version française :
Algorithmique
Cours avec 957 exercices et 158 problèmes
T. H. Cormen, C. E. Leiserson, R. L. Rivest
et C. Stein,
3e édition, 1296 pages,
2010, Dunod.

1 INTRODUCTION AUX ALGORITHMES

Commençons par une question récurrente : « *Qu'est-ce qu'un algorithme ?* »[1]

Une réponse générale pourrait être « un ensemble d'étapes qui permettent d'accomplir une tâche ». Dans notre vie quotidienne, nous exécutons de nombreux algorithmes, par exemple lorsque nous nous brossons les dents : ouvrir le tube dentifrice, prendre la brosse à dents, déposer du dentifrice sur la brosse, fermer le tube, placer la brosse dans la bouche, brosser les dents verticalement pendant N secondes, etc. Pour nous rendre sur notre lieu de travail, nous déroulons également un algorithme. Il en va ainsi pour toutes les tâches.

Cet ouvrage concerne toutefois des algorithmes qui s'exécutent sur des ordinateurs ou, de façon plus générale, sur des appareils informatiques. Tout comme les algorithmes que nous déroulons affectent votre vie quotidienne, ceux qui s'exécutent sur les ordinateurs affectent ceux-ci. Lorsque nous utilisons notre GPS pour connaître l'itinéraire vers notre destination, il exécute un algorithme du « plus court chemin » pour la déterminer. Lorsque nous achetons des produits sur Internet, nous utilisons (ou devrions utiliser) un site web sécurisé qui met en place un algorithme de chiffrement. Ces produits sont apportés à notre domicile par un service de livraison qui utilise des algorithmes afin de répartir les paquets dans les différents camions et de déterminer l'ordre de passage chez les clients. Les algorithmes sont partout : sur notre ordinateur portable, sur les serveurs, sur notre smartphone, sur les systèmes embarqués (notre voiture, notre micro-onde ou notre climatisation), etc.

Quelle est la différence entre un algorithme qui s'exécute sur un ordinateur et un algorithme que nous déroulons ? Contrairement à un ordinateur, nous pouvons accepter que l'algorithme soit décrit de façon peu précise.

1. Ou comme le demanderait un collègue avec qui j'avais l'habitude de jouer au hockey : *« Qu'est-ce qu'un nalgorithme ? »*

© Dunod – Toute reproduction non autorisée est un délit.

Par exemple, si nous prenons notre voiture pour nous rendre sur notre lieu de travail, l'algorithme pourrait stipuler « si la circulation est encombrée, changer de chemin ». Nous saurons sans doute interpréter « mauvaise circulation », pas un ordinateur.

Par conséquent, un algorithme informatique est un ensemble d'étapes qui permettent d'accomplir une tâche et qui est décrit de manière suffisamment précise pour qu'un ordinateur puisse l'exécuter. Si vous avez déjà écrit des programmes informatiques en Java, C, C++, Python, Fortran, Matlab ou autres, vous savez ce que signifie précision. Dans le cas contraire, vous le découvrirez rapidement en étudiant les algorithmes étudiés dans cet ouvrage.

Passons à la question suivante : « *Qu'attendons-nous d'un algorithme informatique ?* »

Les algorithmes informatiques résolvent des problèmes informatiques. Nous en attendons deux choses : proposer une solution juste à partir des données d'entrée du problème et utiliser pour cela les ressources de calcul de manière efficace. Examinons tour à tour ces deux points.

Exactitude

Que signifie *« proposer une solution exacte à un problème »* ? En général, nous sommes en mesure d'indiquer précisément en quoi consiste une solution exacte. Par exemple, une solution juste donnée par notre GPS à notre problème de recherche du meilleur trajet vers notre destination serait la route qui, parmi toutes les possibilités existantes entre notre point de départ et notre point d'arrivée, prend le moins de temps. Ou bien celle qui correspond à la plus courte distance. Ou bien celle qui prend le moins de temps en évitant les péages. Bien entendu, les informations utilisées par le GPS pour déterminer l'itinéraire peuvent ne pas correspondre à la réalité. À moins qu'il ne soit capable d'accéder aux informations de trafic en temps réel, il peut supposer que le temps de parcours est égal à la distance divisée par les limitations de vitesse. Cependant, en cas d'embouteillage, le GPS pourrait donner un mauvais conseil si l'objectif était le trajet le plus rapide. Néanmoins, l'algorithme d'établissement de l'itinéraire exécuté par le GPS reste correct même si les entrées ne le sont pas. En effet, à partir des informations fournies à l'algorithme, il renvoie le chemin le plus rapide.

Dans le cas de certains problèmes, il peut être difficile, voire impossible, d'indiquer si un algorithme produit une solution exacte. Prenons pour exemple un algorithme de reconnaissance de caractères. L'image de 11×6 pixels suivante correspond-elle à un 5 ou à un S ?

Pour certaines personnes, il s'agit d'un 5, mais, pour d'autres, d'un S. Dans ce cas, comment pouvons-nous déclarer que le résultat donné par l'ordinateur est exact ou non ? Dans cet ouvrage, nous nous focalisons sur les algorithmes qui produisent des solutions connaissables.

Nous pouvons parfois accepter qu'un algorithme informatique produise une réponse incorrecte, tant que nous contrôlons leur fréquence. Le chiffrement nous en donne un bon exemple. Le cryptosystème RSA très utilisé détermine si des nombres très grands (vraiment très grands, avec plusieurs centaines de chiffres) sont premiers. Si vous avez déjà écrit un programme informatique, vous serez probablement capable d'en développer un qui détermine si le nombre n est premier. Il suffit de tester tous les diviseurs potentiels, de 2 à $n - 1$. Si l'un d'eux est un diviseur de n, alors n est un nombre composé. Dans le cas contraire, n est un nombre premier. Cependant, si n comprend plusieurs centaines de chiffres, les diviseurs potentiels sont extrêmement nombreux et, même sur un ordinateur très rapide, les tests ne pourront pas se faire en un temps raisonnable. Nous pouvons évidemment procéder à quelques optimisations, comme supprimer tous les nombres pairs après avoir déterminé que 2 n'est pas un diviseur ou arrêter le test en arrivant à $\sqrt{n}$ (en effet, si d est supérieur à $\sqrt{n}$ et si d est un diviseur de n, alors n / d est inférieur à $\sqrt{n}$ et est également un diviseur de n ; par conséquent, si n possède un diviseur, il aura été trouvé avant d'arriver à $\sqrt{n}$). Si n est constitué de centaines de chiffres, alors $\sqrt{n}$ en comprendra environ moitié moins, mais cela reste encore un très grand nombre. Voici une bonne nouvelle : il existe un algorithme qui détermine rapidement si un nombre est premier. Voici la mauvaise nouvelle : il peut se tromper. Plus précisément, s'il déclare que n est un nombre composé, alors c'est effectivement le cas. En revanche, s'il déclare que n est premier, il existe une possibilité que n soit en réalité composé. La mauvaise nouvelle ne l'est pourtant pas autant qu'on pourrait le penser : nous pouvons faire en sorte que le taux d'erreur soit réellement faible, par exemple de l'ordre d'une sur 2^{50}. Pour la plupart d'entre nous, ce taux est suffisamment faible pour que nous puissions employer dans RSA cette façon de déterminer si un nombre est premier.

Pour une autre classe d'algorithmes, appelés *algorithmes d'approximation*, l'exactitude est un point plus délicat. Ces algorithmes s'appliquent aux problèmes d'optimisation dans lesquels nous souhaitons obtenir la meilleure solution en fonction d'une certaine mesure quantitative. Trouver l'itinéraire le plus rapide, comme le fait un GPS, en est un exemple ; dans ce cas, la mesure quantitative est la durée du voyage. Pour certains problèmes, aucun algorithme ne permet d'arriver à une solution optimale en un temps raisonnable, mais nous savons qu'un algorithme d'approximation permet d'obtenir, en un temps raisonnable, une solution quasi optimale. En général, « quasi optimale » signifie que la mesure quantitative de la

© Dunod – Toute reproduction non autorisée est un délit.

solution déterminée par l'algorithme d'approximation est éloignée d'un facteur connu de la mesure quantitative de la solution optimale. Tant que nous pouvons préciser ce facteur, nous pouvons dire qu'une solution correcte fournie par l'algorithme d'approximation correspond à toute solution qui s'éloigne au plus de ce facteur de la solution optimale.

Utilisation des ressources

Dans le contexte d'un algorithme, que signifie une utilisation efficace des ressources de calcul ? Dans notre présentation des algorithmes d'approximation, nous avons fait allusion à une mesure d'efficacité : le temps. Un algorithme qui produit une solution correcte mais en prenant pour cela beaucoup de temps aura probablement un intérêt réduit, voire nul. S'il faut une heure à notre GPS pour déterminer le trajet, allons-nous vraiment nous en servir ? Le temps est ainsi la première mesure d'efficacité utilisée pour évaluer un algorithme, après que nous avons montré qu'il produit une solution exacte. Mais il existe d'autres mesures. Nous pouvons être concernés par la quantité de mémoire nécessaire à l'exécution de l'algorithme (son « empreinte mémoire ») car elle doit s'effectuer dans l'espace mémoire disponible. Voici d'autres ressources dont un algorithme peut avoir besoin : communication réseau, données aléatoires (les algorithmes qui effectuent des choix aléatoires ont besoin d'une source de nombres aléatoires) ou opérations disque (pour les algorithmes conçus de façon à manipuler des données stockées sur le disque).

Dans cet ouvrage, comme dans la plupart des livres qui traitent des algorithmes, nous nous focaliserons sur une seule ressource : le temps. Comment pouvons-nous juger du temps requis par un algorithme ? Contrairement à l'exactitude, qui ne dépend pas de l'ordinateur sur lequel s'exécute l'algorithme, le temps d'exécution réel dépend de plusieurs facteurs externes à l'algorithme lui-même : la rapidité de l'ordinateur, le langage de programmation utilisé pour sa mise en œuvre, le compilateur ou l'interpréteur qui traduit le programme en code exécutable sur la machine, les compétences du programmeur qui écrit le programme et les autres activités qui ont lieu sur l'ordinateur en même temps que le programme. Par ailleurs, tout cela suppose que l'algorithme s'exécute sur un seul ordinateur et que toutes ses données soient présentes en mémoire.

Si nous évaluons la rapidité d'un algorithme en l'implémentant à l'aide d'un langage de programmation réel, en l'exécutant sur un ordinateur précis avec des entrées données et en mesurant le temps nécessaire à son exécution, nous ne savons pas quelle pourra être sa rapidité avec des entrées de taille différente ou avec des entrées différentes de même taille. Par ailleurs, si nous voulons comparer la vitesse de cet algorithme à celle d'un autre algorithme pour le même problème, nous devons les implémenter tous les

deux et les exécuter avec des entrées différentes de différentes tailles. En résumé, comment pouvons-nous évaluer la rapidité d'un algorithme ?

La réponse se fonde sur une combinaison de deux idées. Premièrement, nous déterminons la durée d'un algorithme en fonction de la taille de son entrée. Dans notre exemple de recherche d'itinéraire, l'entrée correspond à une représentation de la carte routière et la taille dépend du nombre d'intersections et du nombre de routes qui les relient. (La taille physique du réseau routier n'a pas d'importance car nous pouvons représenter toutes les distances par des nombres, qui occupent tous la même taille dans l'entrée ; la longueur d'une route n'a pas d'impact sur la taille de l'entrée.) Dans un exemple plus simple, comme déterminer si un élément précis existe dans une liste donnée, la taille de l'entrée correspond au nombre d'éléments de cette liste.

Deuxièmement, nous nous focalisons sur la manière dont la rapidité de la fonction qui caractérise le temps d'exécution augmente avec la taille de l'entrée – le *taux de croissance* du temps d'exécution. Au Chapitre 2, nous donnerons les notations employées pour caractériser le temps d'exécution d'un algorithme. Le point le plus intéressant de notre approche est que nous examinons uniquement le terme dominant du temps d'exécution, sans tenir compte des coefficients. Autrement dit, nous nous concentrons sur l'**ordre de grandeur** du temps d'exécution. Par exemple, supposons que nous puissions déterminer qu'une certaine mise en œuvre d'un algorithme de recherche dans une liste de n éléments prend $50n + 125$ cycles processeur. Le terme $50n$ domine le terme 125 dès que n devient suffisamment grand, c'est-à-dire lorsque $n \geq 3$, et plus la liste est longue plus il est dominant. Par conséquent, dans la description du temps d'exécution de cet algorithme hypothétique, nous ne tenons pas compte du terme d'ordre inférieur 125. Vous pourriez être surpris d'apprendre que le coefficient 50 ne nous intéresse pas. Le temps d'exécution est donc caractérisé par une croissance linéaire en fonction de la taille de l'entrée, c'est-à-dire n. Prenons un autre exemple, celui d'un algorithme qui demande $20n^3 + 100n^2 + 300n + 200$ cycles processeur. Son temps d'exécution croît en fonction de n^3. À nouveau, les termes d'ordre inférieur – $100n^2$, $300n$ et 200 – deviennent de moins en moins significatifs au fur et à mesure que la taille de l'entrée n augmente.

Dans la pratique, les coefficients que nous ignorons ont une importance. Toutefois, ils dépendent tellement de facteurs externes qu'il est possible que, lors de la comparaison des algorithmes A et B ayant le même ordre de grandeur et s'exécutant sur la même machine, l'algorithme A soit plus rapide que l'algorithme B pour une certaine combinaison de machine, de langage de programmation, de compilateur/interpréteur et de programmeur, alors qu'avec une autre combinaison B soit plus rapide que A. Bien entendu, si les deux algorithmes A et B produisent des solutions exactes et

© Dunod – Toute reproduction non autorisée est un délit.

si A s'exécute toujours deux fois plus rapidement que B, alors, toutes choses étant égales par ailleurs, nous choisirons A à la place de B. Cependant, du point de vue de la comparaison abstraite des algorithmes, nous nous intéressons à l'ordre de grandeur, en oubliant les coefficients et les termes d'ordre inférieur.

La réponse à la dernière question de ce chapitre, « *Pourquoi s'intéresser aux algorithmes informatiques ?* », dépend de qui vous êtes.

Algorithmes pour les non-informaticiens

Même si vous ne vous considérez pas comme un informaticien, les algorithmes informatiques vous concernent. En effet, à moins que vous ne partiez à l'aventure dans une contrée sauvage sans un GPS, vous les utilisez probablement quotidiennement. Avez-vous effectué une recherche sur Internet aujourd'hui ? Le moteur de recherche que vous avez utilisé, qu'il s'agisse de Google, de Bing ou d'un autre, se fonde sur des algorithmes complexes pour parcourir le Web et décider de l'ordre dans lequel présenter les résultats. Avez-vous conduit votre voiture aujourd'hui ? À moins qu'il s'agisse d'un véhicule de collection, son ordinateur de bord a réalisé des millions de choix au cours de votre déplacement, tous déterminés par des algorithmes. Je pourrais citer de nombreux autres exemples.

En tant qu'utilisateur final des algorithmes, vous vous devez d'en savoir un peu plus sur leur conception, leur caractérisation et leur évaluation. Je suppose que cela vous intéresse au moins un minimum, car vous avez acheté cet ouvrage et l'avez lu jusqu'à ces lignes. Voyons si nous pouvons vous en donner plus afin que vous puissiez participer aux conversations lors de votre prochain cocktail si jamais les algorithmes venaient à en être le sujet[1].

Algorithmes pour les informaticiens

Si vous êtes informaticien, il est préférable que vous vous intéressiez de près aux algorithmes informatiques. Ils sont non seulement au cœur de tout ce qui se trouve dans l'ordinateur, mais ils constituent également autant une technologie que tous ces composants. Vous pouvez acheter un ordinateur au prix fort, avec le dernier et le plus puissant des processeurs, mais vous

1. Je sais parfaitement que, à moins de résider dans la Silicon Valley, il est peu probable que les algorithmes soient un sujet débattu lors des cocktails auxquels vous participez. Cependant, quelle qu'en soit la raison, nous, professeurs d'informatique, pensons qu'il est important que nos étudiants ne nous mettent pas dans l'embarras en raison d'un manque de connaissances dans certains domaines de l'informatique.