Mathématiques et Applications 83

Brigitte Chauvin
Julien Clément
Danièle Gardy

Arbres pour l'Algorithmique

SMAI

Springer

Mathématiques et Applications

Directeurs de la collection :
M. Hoffmann et V. Perrier

83

More information about this series at http://www.springer.com/series/2966

MATHÉMATIQUES & APPLICATIONS

Comité de Lecture 2018–2021/Editorial Board 2018–2021

Directeurs de la collection
M. HOFFMANN et V. PERRIER

Brigitte Chauvin • Julien Clément • Danièle Gardy

Arbres pour l'Algorithmique

Springer

Brigitte Chauvin
Laboratoire de Mathématiques
Université Versailles
Saint-Quentin-en-Yvelines
Versailles Cedex, France

Julien Clément
GREYC, CNRS UMR 6072
Normandie Université
Caen Cedex, France

Danièle Gardy
Laboratoire DAVID
Université Versailles
Saint-Quentin-en-Yvelines
Versailles Cedex, France

ISSN 1154-483X ISSN 2198-3275 (electronic)
Mathématiques et Applications
ISBN 978-3-319-93724-3 ISBN 978-3-319-93725-0 (eBook)
https://doi.org/10.1007/978-3-319-93725-0

Library of Congress Control Number: 2018949724

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

À Philippe Flajolet,
sans qui ce livre n'aurait pas existé

Préface

Trees are a fundamental object in graph theory and combinatorics as well as a basic object for data structures and algorithms in computer science. During the last few decades, research related to trees has been constantly increasing and several algorithmic, asymptotic and probabilistic techniques have been developed in order to study and to describe characteristics of interest related to trees in different settings.

The *French School* was (and still is) certainly a driving force in this development. This is also reflected in a considerable number of research groups that devote their research mainly to tree structures and related questions. In this context I would like to mention Philippe Flajolet (1948–2011) who was not only the initiator of research activities related to various tree models but was also a mentor for many researchers in France and abroad, in particular for the authors of the present book and also for myself.

This book is a very valuable outcome of this fruitful development in France, where the three authors have taken a very active part. They have succeeded in writing an advanced textbook that can serve as an introduction to trees for students in mathematics and/or computer science as well as a reference book for scientists of other fields like biology. What is really new is that the authors present different approaches to problems on trees that range from algorithms to combinatorics and to probability theory. They have invested a considerable effort to demonstrate the connections and relations between these points of view. This is a very important added value which cannot be found somewhere else in such a transparent and consistent way.

The choice of topics is also done with care. The models are very well motivated and range from trees that are related to branching processes, to binary search trees, to tries and to urn models. There are also several very useful appendices and a very carefully chosen list of exercises which make the book even more useful—also for classes. It would be very good if an English version could be available in the near future.

I want to express my congratulations to the authors for this excellently written book. I am sure it will develop into a standard text and reference book in the field.

Vienna, Austria
November 2016

Michael Drmota

Avant-propos

Les recherches sur les structures arborescentes relèvent historiquement de deux domaines initialement disjoints : les mathématiques, notamment les mathématiques discrètes et les probabilités, et l'informatique fondamentale, avec l'analyse d'algorithmes. Nous avons cherché à présenter simultanément ces deux approches, que nous estimons être complémentaires plutôt que concurrentes ; ce livre en est le résultat. Il est en partie issu d'un cours de troisième cycle, enseigné il y a plusieurs années par deux des auteurs pour un diplôme de Mathématiques-Informatique, et qui visait déjà à présenter conjointement les méthodes issues de la combinatoire analytique et des probabilités.

À qui s'adresse cet ouvrage ? Notre livre s'adresse typiquement aux étudiants de niveau master scientifique ou en dernière année d'école d'ingénieurs, qui auraient auparavant suivi un cursus en informatique ou en mathématiques et qui aspirent à explorer davantage les contrées intermédiaires entre ces deux disciplines ; les étudiants visant une double compétence en mathématiques et informatique pourront ainsi y trouver un intérêt. Il s'adresse en outre à toute personne dotée d'un bagage scientifique « minimal », qui serait amenée à utiliser des structures arborescentes liées à des algorithmes, et qui souhaiterait avoir une meilleure connaissance de ces structures et une idée des performances des algorithmes associés, sans se plonger dans les travaux originaux. Il peut ainsi constituer une introduction à la recherche sur ces sujets, sans prétendre davantage et notamment sans prétendre se situer à l'état de l'art. En effet, il ne s'agit pas d'un livre sur les arbres en général, mais d'un livre sur les arbres pour l'algorithmique. Nous espérons que les spécialistes y trouveront eux aussi un intérêt. Les probabilistes pourront y trouver des indications sur l'utilisation des arbres aléatoires en informatique, notamment dans le champ de l'analyse d'algorithmes ; de leur côté, les informaticiens pourront bénéficier d'un éclairage probabiliste sur certains de leurs objets de base.

Nous ne prétendons en aucun cas à l'exhaustivité, mais plus raisonnablement à rendre compte, selon des critères éminemment subjectifs (il a fallu faire des choix !), des résultats qui nous paraissent à la fois importants et suffisamment simples à exposer. Nous avons aussi souhaité mettre en avant des méthodes

devenues classiques pour établir ces résultats. Certaines parties, qualifiées parfois de « folklore », et certains théorèmes sont connus depuis des décennies et se trouvent éparpillés dans divers ouvrages ; quelques-uns de ces ouvrages sont indiqués dans le chapitre d'introduction et un plus grand nombre sont présentés dans une perspective historique dans l'annexe D. D'autres parties de ce livre ont trait à des développements plus récents qui sont encore peu (ou pas) diffusés sous forme de livre. Enfin quelques résultats importants sont seulement mentionnés car leur exposition détaillée avec preuve « sortirait du cadre de ce livre ». Pour signaler au lecteur ces parties, nous les avons écrites sur fond grisé et nous avons distingué plusieurs cas :

 indique qu'il faut prendre un papier et un crayon mais cette partie est élémentaire, il n'y a pas de difficulté, ni technique ni autre ;

 indique que cette partie est plus technique ;

 indique que pour cette partie il est nécessaire d'avoir recours à des notions qui ne sont pas dans ce livre.

En outre, nous avons souvent proposé en exercices des parties de preuves.

Dans notre souci de présenter simultanément des approches a priori distinctes (probabiliste, combinatoire et algorithmique), nous avons tenté d'unifier les notations et définitions employées par les deux communautés différentes que sont les mathématiciens et les informaticiens. Il a fallu faire des compromis ; parfois l'unification était hors d'atteinte. Ainsi des notations pourront sembler lourdes, des définitions paraîtront inutilement formelles ; inversement, des passages seront plus intuitifs et moins formels, reposant sur l'exemple. Dans l'ensemble, nous avons été guidés par un souci de cohérence et de (relative) simplicité. Nous avons ainsi simplifié certains passages mathématiquement touffus, au nom de l'accessibilité et de la pédagogie, en espérant ne pas avoir sacrifié la rigueur.

Plan du livre Dans les chapitres 1 à 3 sont posées les bases, sont définis les *modèles* : ce qu'est un arbre, ce qu'il modélise et quels sont ses emplois les plus courants en informatique, ce que signifie la notion d'« arbre aléatoire ». Nous avons fait le choix de séparer drastiquement l'aléa dans cette présentation, afin de mieux distinguer ensuite dans les analyses ce qui ressort plutôt de méthodes combinatoires ou plutôt de méthodes probabilistes.

– Au chapitre 1 sont définies de multiples variétés d'arbres, planaires ou non, marqués ou non, ainsi que les paramètres les plus classiques sur ces arbres. Il est tout à fait possible de ne pas lire ce chapitre d'une traite, mais de s'y référer seulement en cas de besoin.
– Le chapitre 2 enrichit les définitions, en introduisant les types d'aléa, différents suivant les variétés d'arbres. De même que le chapitre 1, c'est essentiellement un chapitre de référence, à lire selon le besoin.
– Le chapitre 3 contient plusieurs exemples de modélisations par des structures arborescentes, soit pour les problèmes classiques que sont le traitement de

chaînes de caractères, la recherche d'un élément dans un ensemble, et le tri d'un ensemble, soit pour des situations plus élaborées ; c'est ce chapitre qui justifie un éventuel intérêt pratique du livre.

Dans les chapitres 4 à 9 sont *analysées* de nombreuses familles d'arbres, qui diffèrent notamment par le type d'aléa.

- Au chapitre 4 sont étudiés d'abord les arbres binaires planaires, puis différentes familles d'arbres planaires (familles simples d'arbres, tas, arbres équilibrés), et ensuite les arbres non planaires, le tout sous un modèle probabiliste où les arbres de même taille (parfois de même hauteur) sont *équiprobables*. Nous sommes ainsi dans le domaine de la combinatoire.
- Au chapitre 5 sont présentés les processus de branchement, notamment les arbres de Galton-Watson et les marches aléatoires branchantes. Un lien est également établi entre les arbres de Galton-Watson et les arbres « combinatoires » étudiés au chapitre précédent. Le point de vue est largement probabiliste, même si la récursivité partout présente n'est pas sans rappeler les raisonnements du type « diviser pour régner » utilisés aux chapitres 2 et 4.
- Le chapitre 6 concerne les arbres binaires de recherche et plusieurs de leurs extensions, venues soit des probabilités (arbres biaisés), soit de l'informatique (arbres récursifs, arbres binaires de recherche randomisés, lien avec le tri rapide). Nous utilisons les deux points de vue probabiliste ou combinatoire de façon complémentaire.
- Les tries, qui sont la structure digitale de base, sont analysés dans le chapitre 7, essentiellement avec des outils de combinatoire analytique.
- Deux types d'arbres de recherche, étendant les classiques arbres binaires de recherche, sont analysés en détail dans le chapitre 8 : les arbres m-aires, dans lesquels il est possible d'avoir plusieurs clés dans un même nœud, puis les arbres quadrants, qui permettent de prendre en compte des clés multi-dimensionnelles.
- Enfin, au chapitre 9 sont approfondis les résultats sur les arbres de recherche, en voyant certains de leurs paramètres comme une urne de Pólya ; les analyses font intervenir successivement combinatoire analytique et probabilités.

Les annexes qui terminent ce livre devraient permettre à nos lecteurs de trouver les bases nécessaires pour suivre nos modélisations et analyses.

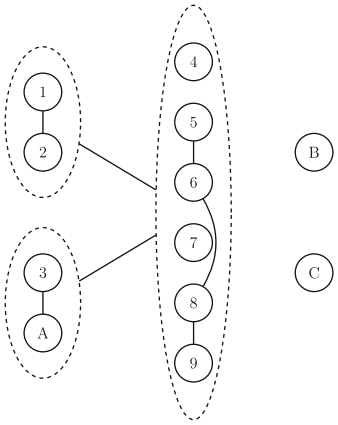
- À l'annexe A sont présentés les algorithmes les plus classiques sur la plupart des structures arborescentes que nous rencontrons dans les chapitres précédents.
- Les annexes B et C, quant à elles, sont des compendiums des notions mathématiques, respectivement combinatoires et probabilistes, nécessaires à la compréhension des modèles et des analyses présentés dans les chapitres 1 à 9.
- Enfin, l'annexe D présente une histoire, partielle et partiale, de l'utilisation des arbres en analyse d'algorithmes.

Prérequis Une familiarité avec les outils mathématiques de base (niveau L2) est souhaitable. Une connaissance de tout ou partie des outils que nous utilisons (combinatoire analytique, probabilités), ou des implémentations informatiques des

structures arborescentes, est utile mais pas indispensable ; en outre, il y a plusieurs niveaux de lecture de ce livre, selon que la lectrice/le lecteur souhaite plutôt utiliser algorithmiquement un résultat donné, ou maîtriser quelques méthodes de démonstration.

Possibilités de lecture (voir l'illustration ci-dessous) Les chapitres 1 et 2 pourront servir de chapitres de référence plutôt qu'être lus en tant que tels. Parmi les chapitres 3 à 9, plusieurs choix pourront être retenus selon les intérêts de la lectrice/du lecteur ou l'orientation que cette personne souhaiterait donner à un cours qui serait basé sur ce livre.

Si l'intérêt porte essentiellement sur les algorithmes, le chapitre 3 (arbres comme modèles d'analyse d'algorithmes) est fondamental ; il sera ensuite loisible de choisir, dans chaque chapitre, les sections qui mettent en perspective les résultats mathématiques sur les paramètres des structures arborescentes et les relient aux performances des algorithmes les utilisant. Si l'accent est mis sur la combinatoire analytique, le choix portera sur les chapitres 4 (modèle combinatoire pour plusieurs familles d'arbres), 7 (structures digitales), la seconde partie du chapitre 8 (arbres quadrants) et la première partie du chapitre 9 (urnes de Pólya). Si l'accent est mis sur l'analyse probabiliste, le choix portera prioritairement sur les chapitres 5 (processus de branchement) et 6 (arbres binaires de recherche), puis sur la première partie du chapitre 8 et le chapitre 9 pour aborder des extensions des arbres binaires de recherche.



Style Le style est celui de Springer. Ce style a parfois induit une lisibilité réduite. Par exemple, la fin des définitions n'est signalée que par un saut de ligne et non par un changement de fonte.

Remerciements Nous remercions ici nos collègues qui ont relu tout ou partie des différentes versions du livre : Marie-Louise Bruner, Élie de Panafieu, Philippe

Duchon, Patrick Gardy, Antoine Genitrini, Cécile Mailler, Cyril Nicaud, Nicolas Pouyanne, Yann Strozecki, Brigitte Vallée, Frédéric Voisin.

Danièle Gardy remercie également le Département de Mathématiques Discrètes et Géométrie (DMG) de l'Université de Technologie de Vienne, qui l'a accueillie notamment pour une année sabbatique en 2012–13, et où une part importante de sa contribution a été écrite.

Caen, France — Brigitte Chauvin
Versailles, France — Julien Clément
Vienne, Autriche — Danièle Gardy
Juillet 2017

Introduction

Qu'est-ce qu'un arbre ?

Étudiés depuis longtemps par les mathématiciens avec des outils probabilistes ou combinatoires, les arbres font partie des structures de données fondamentales en informatique ; ils sont donc aussi sujet naturel d'étude pour les informaticiens. Dans cet ouvrage, nous ne choisirons pas l'un ou l'autre point de vue, mais essaierons de présenter les deux de façon complémentaire.

Qu'est-ce qu'un arbre ? Nous le définirons rigoureusement dans le chapitre 1 ; nous en donnons deux premiers exemples dans la figure 1, et deux visions :

– au sens mathématique, c'est un graphe connexe sans cycle, parfois enraciné[1] ;
– au sens informatique, c'est une structure de données récursive : un « nœud » appelé « racine » et ses « enfants », qui sont eux-mêmes des arbres.

Comme les arbres que les informaticiens utilisent, notamment pour stocker des informations, ont (presque) toujours une racine, i.e. un nœud qui joue un rôle particulier et sert d'« ancre » à l'arbre,

les arbres considérés dans ce livre sont tous enracinés.

Ainsi, l'arbre de gauche de la figure 1 – nous employons, pour dessiner un arbre, la convention de mettre la racine en haut ; l'arbre « pousse » donc vers le bas – a une racine, qui a elle-même trois enfants ; en lisant de gauche à droite, les

[1] Tout livre sur les graphes fournit les définitions de base sur le sujet, en particulier celle d'un arbre ; nous y renvoyons la lectrice intéressée, et ne poursuivons pas cette approche ici.

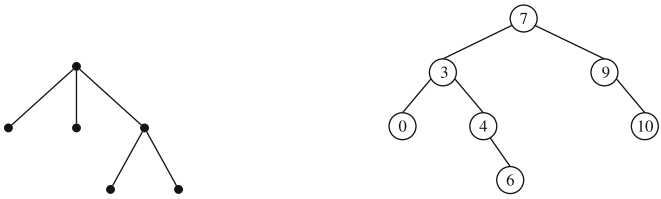
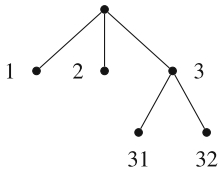


Fig. 1 Deux arbres (enracinés) : le premier a six nœuds ; le second a sept nœuds, dont chacun est marqué par une valeur (ici un nombre entier)

deux premiers n'ont pas d'enfants, et le troisième a lui-même deux enfants. C'est pourquoi la représentation canonique de cet arbre est la suivante :



Il est question dans ce livre des arbres « pour l'algorithmique », i.e. les arbres sont pour nous *à la fois* des structures de données informatiques et des objets mathématiques sous-jacents. Notre intérêt pour les arbres vient de leur pertinence à modéliser de nombreuses situations, notamment des algorithmes et des méthodes de résolution de problèmes informatiques, qu'ils soient de base (recherche, tri, etc.) ou plus complexes. Les propriétés de ces arbres permettent d'évaluer le « coût » ou les « performances » des algorithmes qui l'utilisent. Examinons maintenant de plus près ces notions de coût et de performance d'un algorithme ou d'une structure de données.

Évaluation de la performance d'un algorithme

Lors de la conception d'un système informatique ou après sa réalisation advient une phase d'évaluation de ses performances, définies en termes de temps d'exécution, de place mémoire requise, de nombre de messages échangés, etc. Cette évaluation porte, soit sur le système tout entier, soit – et c'est cela qui nous intéresse ici – sur une partie du système : sur un algorithme spécifique, ou sur une structure de données et les algorithmes l'utilisant. Pour cela, il y a plusieurs manières de faire : mesures sur un système existant, simulations, analyses théoriques. Ces approches sont toutes utiles et complémentaires.

– Les *mesures* apportent des informations précises sur le comportement d'un système donné, mais leur résultat peut dépendre de l'art du programmeur, de la machine (matériel) et du système (logiciel) – il faut recommencer les mesures

quand l'environnement change – de la charge du système à un moment donné (toute observation modifie le système observé), etc. De plus, pour mesurer les performances d'un système il faut l'avoir déjà réalisé : comment faire, si nous voulons les prédire avant même la réalisation ? Faire des choix de conception entre plusieurs possibilités ?

– Les *simulations* peuvent elles aussi dépendre de la machine, de l'art de la personne qui a programmé la simulation, etc. Elles sont moins liées à un système existant que les mesures, et sont intéressantes pour obtenir des analyses ou des prédictions lorsque nous ne pouvons pas construire de modèle mathématique, ou qu'un modèle précis est trop complexe pour être exploitable.
– Les *analyses théoriques* fournissent des théorèmes ; une fois mis en évidence, le coût « théorique » est indépendant de l'art du programmeur et de la machine. Cette approche demande la maîtrise d'outils mathématiques parfois sophistiqués. Les résultats fournis sont idéalement complétés par des mesures ; par exemple, si un algorithme nécessite n^2 opérations unitaires (lecture de symbole, comparaison de clés, envoi de message) où n est un paramètre connu du système (nombre de symboles à lire, de données à trier, d'agents à synchroniser), encore faut-il connaître, pour une machine donnée, le temps pris par une opération unitaire pour évaluer la durée d'exécution de cet algorithme.

Le présent ouvrage se place dans le cadre des analyses théoriques, et s'intéresse à des aspects précis de parties d'un système informatique : *l'analyse* fine du comportement d'un *algorithme,* agissant sur une *structure de données* arborescente. Il y a plusieurs manières d'envisager cette analyse :

– soit nous nous intéressons aux cas extrémaux, i.e. au comportement de l'algorithme dans le pire (ou le meilleur) des cas ;
– soit nous cherchons plutôt à caractériser un comportement moyen, ou mieux encore la distribution de probabilité d'un paramètre du système.

La première approche conduit par exemple à identifier des classes de problèmes résolubles dans le pire des cas en un temps polynomial en la taille des données, et d'autres qui ne le sont pas : c'est la « théorie de la complexité ». La seconde repose sur l'idée que le comportement d'un algorithme est souvent caractérisé de façon plus pertinente par ses performances sur la plupart des données d'entrée que sur des cas exceptionnels. C'est ce qui est couramment appelé « analyse d'algorithmes » ou (de façon indûment restrictive) « analyse en moyenne », et c'est ce qui sous-tend les analyses présentées dans ce livre.

Une démarche générale

Prenons donc un algorithme, c'est-à-dire une méthode de résolution pour un certain problème, par exemple le tri d'un grand nombre de données, ou la recherche d'une valeur dans un ensemble de grande taille. Cet algorithme prend en entrée des

données (de l'information, dont la représentation dépend de l'utilisation qui en est faite) généralement stockées dans des *structures de données* (par exemple des arbres), et va répondre au problème posé en renvoyant un résultat : pour les deux exemples ci-dessus, les données triées en ordre croissant, ou bien la valeur cherchée, dans la mesure où elle est effectivement présente. L'analyse (des performances) de cet algorithme peut être décomposée en plusieurs étapes :

- Tout d'abord vient une phase de *modélisation,* qui a pour but d'établir un modèle représentant les données, algorithmes, ou systèmes, qui sont l'objet de l'étude ; nous nous limitons dans ce livre aux structures arborescentes et aux algorithmes les utilisant. Les données étant le plus souvent aléatoires ou considérées comme telles, le modèle inclut l'établissement d'une distribution de probabilité sur ces données. À la fin de cette première étape, nous avons donc défini un modèle avec données aléatoires ; pour ce qui nous intéresse dans ce livre, c'est un *arbre aléatoire,* appelons-le τ.
- Dans un deuxième temps, il nous faut dégager les paramètres qui mesurent le coût, ou la « complexité », de l'algorithme, typiquement son temps d'exécution, ou la place mémoire pour représenter une structure de données. Pour fixer les idées dans la suite, appelons $c(\tau)$ ce coût de l'algorithme, exécuté sur un arbre τ.
- Vient alors une phase d'étude mathématique de ce coût d'exécution. L'arbre τ défini dans la première phase est *aléatoire ;* en conséquence le coût $c(\tau)$ est une *variable aléatoire*. Suivant le type de résultat cherché, nous étudions le coût moyen de $c(\tau)$, ou bien ses moments, voire sa distribution de probabilité et sa convergence éventuelle vers une distribution limite.
- La dernière phase est celle du retour au problème algorithmique ; bien qu'essentielle, elle est souvent passée sous silence, ce qui est dommage car nombre de résultats sur les paramètres d'arbres ont une traduction immédiate en termes de performances de l'algorithme étudié.

Dans la phase de modélisation proprement dite, le modèle arborescent pour représenter les données est souvent l'un de ceux présentés en chapitre 1 ; le modèle probabiliste est issu du chapitre 2. Quant à la mise en évidence des propriétés de l'arbre qui déterminent le coût, elle fait le plus souvent intervenir l'un des *paramètres* classiques présentés à la fin du chapitre 1 en section 1.3. La troisième étape, l'analyse des paramètres, constitue le sujet des chapitres 4 à 9 ; suivant le cas analysé et selon le type de résultats cherchés, elle utilise des outils de combinatoire analytique ou de probabilités.

Avant de présenter les arbres étudiés et les méthodes employées, illustrons la modélisation des performances d'un algorithme et l'analyse de sa complexité sur l'exemple (on ne peut plus classique !) des arbres binaires de recherche.

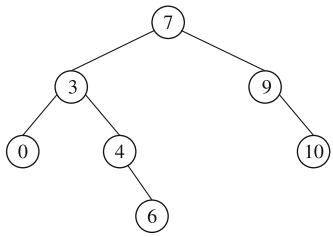


Fig. 2 Un arbre binaire de recherche à 7 clés

Un exemple : coût dans les arbres binaires de recherche

Définissons un arbre binaire de recherche, construit à partir d'un ensemble totalement ordonné $\mathcal{E}$ de clés distinctes, comme suit[2] :

- si $\mathcal{E} = \emptyset$, l'arbre est vide ;
- si $\mathcal{E} = \{x\}$, l'arbre est réduit à un nœud unique contenant la clé x ;
- sinon, choisissons une des clés de $\mathcal{E}$, appelons-la x, qui va dans le nœud racine ; les autres clés de $\mathcal{E}$ servent à construire deux sous-arbres : celui de gauche[3] contient les clés inférieures à x et celui de droite contient les clés supérieures à x. Ces sous-arbres gauche et droit sont eux-mêmes structurés récursivement en arbres binaires de recherche.

L'arbre marqué de la figure 1, qui est redessiné figure 2, est en fait un arbre binaire de recherche construit sur (par exemple) la suite de clés $(7, 3, 9, 4, 0, 6, 10)$: la valeur 7 à la racine divise les clés restantes en deux sous-suites ; les clés de $(3, 4, 0, 6)$ forment le sous-arbre gauche et les clés de $(9, 10)$ forment le sous-arbre droit ; ces deux sous-arbres sont eux-mêmes des arbres binaires de recherche.

Regardons ce qui se passe, pour notre arbre exemple, lorsque nous cherchons la valeur 4. Nous la comparons d'abord à la valeur à la racine, $7 : 4 < 7$, et nous poursuivons la recherche dans le sous-arbre gauche. Sa racine contient la clé 3, à laquelle nous comparons $4 : 4 > 3$, donc nous partons dans le sous-arbre droit de ce sous-arbre gauche. Une dernière comparaison nous permet de vérifier que 4 est la racine de ce sous-arbre, et est bien présent dans l'arbre. Il nous a fallu 3 comparaisons, ce qui correspond aussi au nombre de niveaux testés : la racine, puis un enfant et un petit-enfant de la racine.

Supposons maintenant que nous cherchions la valeur 8. Nous allons toujours comparer cette valeur à la clé à la racine, $7 : 8 > 7$, donc nous allons vers le fils droit. Sa propre racine contient la clé 9, plus grande que 8, donc nous regardons dans son sous-arbre gauche… qui est vide ! Nous savons maintenant que 8 n'est

[2]Nous donnons dans la section 1.2.5 une définition plus rigoureuse des arbres binaires de recherche.

[3]Les arbres sont dessinés dans le plan, ordonnés de gauche à droite par ordre croissant.

pas présent dans l'arbre (s'il se trouvait ailleurs, ce ne serait plus un arbre binaire de recherche) ; de plus nous avons trouvé la place où devrait aller 8, si nous souhaitions l'ajouter à l'ensemble des clés : ce serait comme fils gauche de 9.

Nous avons vu sur cet exemple qu'un arbre binaire de recherche permet de structurer un ensemble $\mathcal{E}$ de clés, de telle sorte qu'il soit facile de retrouver une clé de valeur donnée ; il est également aisé d'ajouter ou de supprimer une clé dans $\mathcal{E}$. Plus formellement, pour chercher une clé x dans un arbre binaire de recherche τ, nous comparons d'abord x à la clé contenue y dans la racine de l'arbre ; si $x = y$, la recherche est finie ; dans le cas contraire, nous poursuivons récursivement la recherche dans le sous-arbre gauche, si $x < y$, et dans le sous-arbre droit sinon. La recherche s'arrête, soit lorsque nous avons trouvé la clé cherchée, soit lorsque nous arrivons à un sous-arbre vide – dans ce cas, nous avons d'ailleurs trouvé la place où x doit être inséré dans l'arbre, si nous souhaitons l'ajouter tout en gardant la structure d'arbre binaire de recherche.

Quel est le coût d'une *recherche avec succès* d'une clé x dans un arbre binaire de recherche τ construit sur un ensemble $\mathcal{E}$ de clés que nous supposons toutes distinctes ? Une unité de mesure possible est le nombre de nœuds visités pour trouver x ; c'est aussi le nombre de comparaisons de x à des clés présentes dans l'arbre. Ce nombre est un paramètre classique : c'est $1+$ la *profondeur*[4] (ou niveau) du nœud contenant x dans l'arbre τ ; notons-le $1 + \mathrm{Prof}\,(x, \tau)$. Regardons maintenant le coût « moyen » d'une recherche avec succès d'une clé x dans un arbre τ, en supposant que chacune des clés présentes dans l'arbre a la même probabilité d'être la clé cherchée : ce coût est alors $1/|\tau| \sum_{x\in\tau} (1 + \mathrm{Prof}\,(x, \tau))$, où $|\tau|$ est le nombre de nœud de τ soit encore la taille de l'arbre τ. Or la somme $\sum_{x\in\tau} \mathrm{Prof}\,(x, \tau)$ est un autre paramètre classique, la *longueur de cheminement* de l'arbre, notée $\mathrm{lc}(\tau)$; le coût moyen d'une recherche avec succès est donc $1 + \mathrm{lc}(\tau)/|\tau|$.

Quant au maximum du coût d'une recherche avec succès, il est obtenu pour les clés les plus éloignées de la racine, et le paramètre associé est le maximum des profondeurs des nœuds de l'arbre : c'est sa *hauteur*.

Ajoutons un niveau d'aléa ; supposons que n clés (leur nombre est supposé fixé) sont choisies selon une certaine distribution de probabilité dans un ensemble.[5] L'arbre, noté τ_n, devient aléatoire, ainsi que sa longueur de cheminement $\mathrm{lc}(\tau_n)$, et l'objectif est de caractériser la loi de $\mathrm{lc}(\tau_n)$, obtenant ainsi la loi du coût d'une recherche avec succès d'une clé dans un arbre binaire de recherche de taille n. Pour la moyenne $\mathrm{lc}_n := \mathbb{E}\,(\mathrm{lc}(\tau_n))$ de la longueur de cheminement, il est intéressant d'utiliser les outils de la combinatoire analytique afin d'établir une équation de récurrence sur lc_n. En la résolvant (soit directement, soit par l'intermédiaire d'une *fonction génératrice*), il est possible d'obtenir une formule close sur lc_n, ainsi que son comportement asymptotique lorsque le nombre n de clés devient grand. Pour

[4] Il est convenu que la racine est à profondeur 0.

[5] Il n'est pas question ici de la manière de construire l'arbre, qui peut dépendre de l'ordre dans lequel les clés sont entrées ; voir pour cela le chapitre 2 et l'annexe A.

étudier non seulement la moyenne mais les moments d'ordres supérieurs ou la loi limite de la longueur de cheminement, il s'avère souvent pertinent de recourir à d'autres méthodes : des théorèmes de point fixe, des outils probabilistes, par exemple en faisant apparaître des martingales.

Pour une recherche sans succès dans un arbre τ, nous pouvons toujours mesurer son coût en nombre de nœuds visités, ou en nombre de comparaisons de clés : c'est la *profondeur d'insertion* de la clé x dans τ. Remarquons que cette profondeur d'insertion de x est la profondeur à laquelle nous trouverons la clé lors de recherches avec succès ultérieures.

Quels arbres ?

Ce livre ne prétend pas être exhaustif. Nous avons retenu les classes d'arbres suivantes, fréquemment rencontrées en informatique :

- **les arbres binaires planaires,** qui sont « la » structure arborescente de base, et plus généralement les **arbres planaires**, puis les familles simples d'arbres et les **arbres non planaires** ;
- **les tas,** qui sont des arbres binaires particuliers, essentiels pour une méthode de tri dite (justement !) « par tas », et qui permettent aussi d'implémenter des files de priorité ;
- **les structures digitales,** essentiellement les tries, qui apparaissent souvent dans les algorithmes sur des chaînes de caractères, et permettent en outre de modéliser le comportement d'algorithmes venant de domaines variés ;
- **les arbres de Galton-Watson** et d'autres **processus de branchement** dont les marches aléatoires branchantes ;
- **les arbres binaires de recherche,** les arbres récursifs qui en sont proches, et certaines de leurs variantes : arbres quadrants, arbres 2-3, arbres-B, arbres m-aires de recherche ; l'analyse de certaines de ces structures, appelée parfois « analyse de frange », fait souvent intervenir des urnes de Pólya.

Nous avons par ailleurs fait le choix d'étudier prioritairement certains paramètres sur les arbres, tout d'abord le nombre d'arbres de taille (nombre de nœuds) donnée, puis la longueur de cheminement et la hauteur, paramètres qui sont à la fois les plus classiques et parmi les plus utiles en analyse d'algorithmes.

Les différents types d'arbres, ainsi que les paramètres liés aux complexités des algorithmes les plus courants sur ces arbres, sont présentés dans les chapitres 1 à 2 ; dans ces deux premiers chapitres sont également mis en place les principaux cadres mathématiques permettant l'analyse de ces paramètres. Le chapitre 3 est à la fois plus informel et au cœur du sujet : il présente un certain nombre d'exemples, plus ou moins classiques, de l'utilisation très diverse de structures arborescentes en algorithmique et en analyse d'algorithmes et de la manière dont les paramètres d'arbres déterminent la complexité d'un algorithme. L'analyse desdits paramètres

est effectuée dans les chapitres 4 à 9, le plus souvent possible sous les deux angles de la combinatoire analytique et des probabilités.

Plusieurs annexes rappellent, d'une part les structures de données informatiques et les algorithmes permettant de manipuler ces structures arborescentes (c'est l'annexe A), d'autre part les outils mathématiques nécessaires pour suivre les analyses (ce sont l'annexe B pour la combinatoire et l'analyse, et l'annexe C pour les probabilités). Enfin l'annexe D présente un historique et une bibliographie subjectifs et non exhaustifs. Donnons maintenant un bref aperçu des méthodes utilisées.

Méthodes

L'exemple des arbres binaires de recherche montre qu'il y a en gros deux classes de méthodes possibles : analytiques et combinatoires d'une part, probabilistes d'autre part. Tout au long de ce livre, nous nous efforçons de présenter ces méthodes simultanément et concurremment.

Combinatoire analytique

Supposons avoir dégagé d'une part une notion de *taille* des données en entrée d'un algorithme et d'autre part une notion de *coût* d'un algorithme, exprimé en fonction d'un paramètre de l'arbre τ associé aux données. Appelons ce coût $c(\tau)$. Lorsque les données sont supposées aléatoires, alors $c(\tau)$ devient une variable aléatoire. Il est alors loisible de définir le *coût moyen,* appelons-le c_n, pour une donnée aléatoire de taille n. Une technique puissante de résolution, lorsqu'il existe une relation de récurrence sur c_n, ou directement sur $c(\tau)$, consiste à introduire la *fonction génératrice* de la suite (c_n), disons $C(z) = \sum_n c_n z^n$. L'obtention de $C(z)$, ou plus fréquemment d'une équation dont elle est solution, est possible grâce à des techniques de combinatoire, en particulier la *méthode symbolique*. Il arrive parfois d'obtenir une expression explicite pour $C(z)$ et pour ses coefficients. Lorsque ce n'est pas le cas, il est néanmoins fructueux de considérer $C(z)$ comme une fonction analytique dans le plan complexe, et d'obtenir sinon la valeur exacte du n-ième coefficient de la fonction, i.e., de c_n, du moins son comportement asymptotique – c'est souvent le plus intéressant, la question du coût d'un algorithme n'ayant un intérêt pratique que pour des données de « grande » taille. L'outil de base pour cette étude asymptotique est la formule de Cauchy ou les adaptations qui en ont été faites, la plus notable pour notre sujet étant le *lemme de transfert* de Flajolet et Odlyzko [90].

Cette approche, qui fait appel à la combinatoire et à l'analyse, a été développée et systématisée par Flajolet, qui lui a donné le nom de *Combinatoire analytique*. Elle permet, à partir d'une spécification formelle des objets combinatoires étudiés (ici, des arbres), d'étudier leurs paramètres, non seulement en moyenne, mais en

distribution, i.e., d'obtenir moments ou convergence vers une loi limite. Nous renvoyons aux livres de Flajolet et Sedgewick [93, 94] pour une présentation de ce domaine de recherche et pour les applications en analyse d'algorithmes.

Probabilités

Les structures arborescentes qui apparaissent aussi bien dans la représentation des données que dans la représentation des choix d'exécution d'un algorithme conduisent naturellement à une modélisation par des arbres aléatoires, en d'autres termes par un ensemble d'arbres sur lequel une loi de probabilité a été définie. De plus, les arbres aléatoires peuvent aussi être vus comme un cas particulier de graphes aléatoires, qui sont eux-mêmes des modèles intervenant par exemple pour décrire des réseaux de télécommunication. Rappelons cependant que dans ce livre un arbre sera toujours *enraciné,* i.e. un sommet est désigné comme racine (à la différence de la plupart des arbres rencontrés en théorie des graphes, où aucun sommet ne joue a priori de rôle particulier).

Une bonne manipulation des arbres aléatoires nécessite de travailler sur des espaces probabilisés. Les processus aléatoires (on dit aussi stochastiques) qui appartiennent à la famille des processus de branchement seront naturellement présents. Ils peuvent décrire simplement l'évolution d'une population en comptant le nombre d'individus à la n-ième génération (ce sont les processus de Galton-Watson), ou bien décrire des évolutions plus riches, en marquant les nœuds ou les branches d'un arbre par toutes sortes de variables qui présentent un intérêt pour l'étude. Les marches aléatoires branchantes en sont un exemple.

D'autres processus stochastiques peuvent aussi intervenir, par exemple pour l'étude des arbres binaires de recherche. Dans les cas détaillés dans cet ouvrage, des méthodes probabilistes classiques, couplées parfois à des méthodes analytiques, permettent de déterminer le comportement asymptotique d'une variable aléatoire correspondant au coût d'un algorithme. Des martingales apparaissent, des marches aléatoires sont associées aux arbres, et les différentes notions de convergence fournissent autant de notions de limite.

Un exemple d'utilisation conjointe : les urnes de Pólya

Les urnes de Pólya sont un modèle polyvalent qui apparaît dès qu'il y a un choix uniforme à faire entre des objets (boules) de différentes espèces (couleurs). Ce sera typiquement le cas lors de l'insertion d'une nouvelle clé dans un arbre de recherche, par exemple dans un arbre 2–3 suivant que la feuille où prend place cette clé contient déjà une ou deux clés. Dans une modélisation d'arbre par une urne de Pólya, les

boules sont les nœuds ou les feuilles, de différentes espèces ou couleurs. Il apparaît alors une matrice de remplacement qui permet de suivre l'évolution des différentes espèces de nœuds lors de mises à jour.

Concurremment, l'urne peut être étudiée soit par combinatoire analytique et par description en termes de fonctions génératrices, soit par des méthodes probabilistes et algébriques. Des informations précises sont obtenues à temps fini ou asymptotiquement sur la répartition des nœuds selon les différentes couleurs.

Table des matières