ADITYA BHARGAVA

LES ALGORITHMES

C'est plus simple avec un dessin !

+ EN LIGNE

OFFERT

Liste des codes Python

Les algorithmes,

c'est plus simple avec un dessin !

Les algorithmes,

c'est plus simple avec un dessin !

Aditya Y. Bhargava

Traduction de Bernard Desgraupes

deboeck SUPÉRIEUR

Ouvrage original

Aditya Y. Bhargava, *Grokking algorithms. An illustrated guide for programmers and other curious people*, Manning Publications and Co, 2016.

MANNING
SHELTER ISLAND

Dépôt légal :
Bibliothèque nationale, Paris : mars 2023
Bibliothèque royale de Belgique, Bruxelles : 2023/13647/008

ISBN : 978-2-8073-4533-1

À mes parents, Sangeeta et Yogesh

Table des matières

Avant-propos xiii

Remerciements xiv

Note de l'auteur xv

1 Introduction aux algorithmes 1

Introduction 1

Ce que vous apprendrez sur les performances 2

Ce que vous apprendrez sur la résolution de problèmes 2

Recherche binaire 3

Une meilleure façon de rechercher 5

Temps d'exécution 10

Notation Grand O 10

Les temps d'exécution des algorithmes augmentent à des rythmes différents 11

Visualisation des différents temps d'exécution en Grand O 13

Grand O établit un temps d'exécution pour le pire des cas 15

Quelques temps d'exécution courants de Grand O 15

Le commis voyageur 17

À retenir 19

2 Tri par sélection 21

Comment fonctionne la mémoire 22

Tableaux et listes chaînées 24

Listes chaînées 25

Tableaux 26

Terminologie 27

Insertion au milieu d'une liste 29
Suppressions 30
Tri par sélection 32
À retenir 36

3 Récursion 37

Récursion 38
Cas de base et cas récursif 40
La pile 42
La pile d'appels 43
La pile d'appels avec récursion 45
À retenir 50

4 Tri rapide 51

Diviser pour régner 52
Quicksort 60
La notation Grand O revisitée 66
Tri par fusion vs tri rapide 67
Cas moyen vs pire des cas 68
À retenir 72

5 Tables de hachage 73

Fonctions de hachage 76
Cas d'usage 79
Utilisation de tables de hachage pour les recherches 79
Empêcher les doublons en entrée 81
Utiliser des tables de hachage comme cache 83
À retenir 86
Collisions 86
Performance 88
Facteur de charge 90
Une bonne fonction de hachage 92
À retenir 93

6 algorithme de parcours en largeur 95

Introduction aux graphes 96
Qu'est-ce qu'un graphe ? 98

Parcours en largeur 99
Trouver le chemin le plus court 102
Files d'attente 103
Implémentation d'un graphe 105
Implémentation de l'algorithme 107
Temps d'exécution **111**
À retenir 114

7 Algorithme de Dijkstra 115

Travailler avec l'algorithme de Dijkstra 116
Terminologie 120
Échange contre un piano 122
Arêtes de poids négatif 128
Implémentation 131
À retenir 140

8 Algorithmes gloutons 141

Le problème des emplois du temps 142
Le problème du sac à dos 144
Le problème de la couverture d'ensemble 146
Algorithmes d'approximation 147
Problèmes NP-complets 152
Commis voyageur, étape par étape 153
Comment savoir si un problème est NP-complet ? 158
À retenir 160

9 Programmation dynamique 161

Le problème du sac à dos 161
La solution simple 162
Programmation dynamique 163
FAQ pour le problème du sac à dos 171
Que se passe-t-il si vous ajoutez un article ? 171
Que se passe-t-il si vous modifiez l'ordre des lignes ? 174
Pouvez-vous remplir la grille en colonnes au lieu de lignes ? 174
Que se passe-t-il si vous ajoutez un article plus petit ? 174
Pouvez-vous voler des fractions d'un objet ? 175
Optimiser votre itinéraire de voyage 175

Gérer des articles qui dépendent les uns des autres 177
Est-il possible que la solution nécessite plus de deux sous-sacs ? 177
Est-il possible que la meilleure solution ne remplisse pas complètement le sac à dos ? 178
Sous-chaîne commune la plus longue 178
Construction de la grille 179
Remplissage de la grille 180
La solution 182
Sous-séquence commune la plus longue 183
Sous-séquence commune la plus longue — solution 184
À retenir 186

10 Méthode des k plus proches voisins 187

Classifier des oranges par rapport aux pamplemousses 187
Construire un système de recommandations 189
Extraction de caractéristiques 191
Régression 195
Choisir les bonnes caractéristiques 198
Introduction à l'apprentissage automatique 199
OCR 199
Création d'un filtre anti-spam 200
Prédire le marché boursier 201
À retenir 202

11 Pour aller plus loin 203

Arbres 203
Index inversés 206
La transformée de Fourier 207
Algorithmes parallèles 208
MapReduce 209
Pourquoi les algorithmes distribués sont-ils utiles ? 209
La fonction map 209
La fonction reduce 210
Filtres Bloom et HyperLogLog 211
Filtres Bloom 212
HyperLogLog 213

Les algorithmes SHA 213
Comparaison de fichiers 214
Vérification des mots de passe 215
Hachage sensible à la position 216
Échange de clés Diffie-Hellman 217
Programmation linéaire 218
Épilogue 219
Réponses aux exercices 221
Index 235

Avant-propos

J'ai abordé la programmation comme un passe-temps. *Visual Basic 6 pour les Nuls* m'a appris les bases et j'ai continué à lire des livres pour en savoir plus. Mais le sujet des algorithmes restait impénétrable pour moi. Je me souviens avoir savouré la table des matières de mon premier livre sur les algorithmes en pensant « Je vais enfin comprendre ce sujet ! ». Mais c'était des trucs denses et j'ai abandonné au bout de quelques semaines. Ce n'est que lorsque j'ai eu un premier bon professeur d'algorithmes que j'ai réalisé à quel point ces idées étaient simples et élégantes.

Il y a quelques années, j'ai écrit mon premier article de blog illustré. Je suis un visuel lorsqu'il s'agit d'apprendre et j'aime vraiment le style illustré. Depuis lors, j'ai écrit quelques articles illustrés sur la programmation fonctionnelle, Git, l'apprentissage automatique et la concurrence. Au fait : j'étais un écrivain médiocre quand j'ai commencé. Expliquer des concepts techniques est difficile. Trouver de bons exemples prend du temps et expliquer un concept difficile prend aussi du temps. Il est donc plus facile de passer sous silence les choses difficiles. Je pensais que je faisais du très bon travail, jusqu'à ce qu'après qu'un de mes messages fut devenu populaire, un collègue soit venu me voir et m'ait dit : «J'ai lu votre message et je ne comprends toujours pas cela.» J'avais encore beaucoup à apprendre sur l'écriture.

Quelque part au milieu de la rédaction de ces articles de blog, Manning m'a contacté et m'a demandé si je voulais écrire un livre illustré. Et il s'avère que les éditeurs de Manning en savent beaucoup sur l'exposé des concepts techniques et ils m'ont appris à enseigner. J'ai écrit ce livre pour soulager une démangeaison particulière : je voulais écrire un livre qui expliquerait bien des sujets techniques difficiles et je voulais un livre sur les algorithmes qui soit facile à lire. Mon écriture a parcouru un long chemin depuis ce premier article de blog et j'espère que vous trouverez ce livre à la fois facile à lire et instructif.

Remerciements

Un grand merci à Manning pour m'avoir offert la chance d'écrire ce livre et m'avoir laissé une grande liberté de création. Merci à l'éditeur Marjan Bace, à Mike Stephens pour m'avoir intégré, à Bert Bates pour m'avoir appris à écrire et à Jennifer Stout qui a été une éditrice incroyablement réactive et utile. Merci également aux personnes de l'équipe de production de Manning : Kevin Sullivan, Mary Piergies, Tiffany Taylor, Leslie Haimes et tous les autres dans les coulisses. De plus, je tiens à remercier les nombreuses personnes qui ont lu le manuscrit et offert des suggestions : Karen Bensdon, Rob Green, Michael Hamrah, Ozren Harlovic, Colin Hastie, Christopher Haupt, Chuck Henderson, Pawel Kozlowski, Amit Lamba, Jean-François Morin, Robert Morrison, Sankar Ramanathan, Sander Rossel, Doug Sparling et Damien White.

Un grand merci aux personnes qui m'ont aidé à atteindre mon objectif : les personnes sur la plateforme de jeu Flaskhit pour m'avoir appris à coder ; les nombreux amis qui m'ont aidé en révisant des chapitres, en me donnant des conseils et en me laissant tester différentes explications, notamment Ben Vinegar, Karl Puzon, Alex Manning, Esther Chan, Anish Bhatt, Michael Glass, Nikrad Mahdi, Charles Lee, Jared Friedman, Hema Manickavasagam , Hari Raja, Murali Gudipati, Srinivas Varadan, et d'autres; et Gerry Brady qui m'a appris les algorithmes. Un autre grand merci aux universitaires spécialistes d'algorithmes comme CLRS, Knuth et Strang. Je suis vraiment épaulé par des géants.

Papa, maman, Priyanka et le reste de la famille : merci pour votre soutien constant. Et un grand merci à ma femme Maggie. De nombreuses aventures nous attendent, et certaines d'entre elles n'impliquent pas de rester chez soi un vendredi soir pour réécrire des paragraphes.

Enfin, un grand merci à tous les lecteurs qui ont tenté leur chance sur ce livre, et aux lecteurs qui m'ont donné leur avis sur le forum du livre. Vous avez vraiment contribué à améliorer cet ouvrage.

Note de l'auteur

Ce livre est conçu pour être facile à suivre. J'évite les grands sauts de réflexion. Chaque fois qu'un nouveau concept est introduit, je l'explique tout de suite ou je vous dis quand je l'expliquerai. Les concepts de base sont renforcés par des exercices et de multiples explications afin que vous puissiez vérifier vos hypothèses et vous assurer que vous suivez bien.

Je vous conduis avec des exemples. Au lieu d'écrire une soupe de symboles, mon objectif est de vous permettre de visualiser facilement les concepts. Je pense aussi qu'on apprend mieux en étant capable de se rappeler quelque chose qu'on sait déjà, et les exemples facilitent le rappel. Ainsi, lorsque vous essayez de vous souvenir de la différence entre les tableaux et les listes chaînées (expliquées au chapitre 2), vous pouvez simplement penser que vous vous asseyez pour voir un film. Aussi, au risque d'énoncer une évidence, je suis un visuel pour l'apprentissage. Ce livre est rempli de dessins.

Le contenu du livre est soigneusement organisé. Il n'est pas nécessaire d'écrire un livre qui couvre tous les algorithmes de tri, pour cela nous avons Wikipédia et Khan Academy. Tous les algorithmes que j'ai inclus sont pratiques. Je les ai trouvés utiles dans mon travail d'ingénieur en programmation, et ils fournissent une bonne base pour des sujets plus complexes. Bonne lecture !

Feuille de route

Les trois premiers chapitres de ce livre jettent les bases :

- **Chapitre 1 :** vous apprendrez votre premier algorithme pratique : la recherche binaire. Vous apprendrez également à analyser la vitesse d'un algorithme en utilisant la notation Grand O. La notation Grand O est utilisée tout au long du livre pour analyser la lenteur ou la rapidité d'un algorithme.

- **Chapitre 2 :** vous découvrirez deux structures de données fondamentales : les tableaux et les listes chaînées. Ces structures de données sont utilisées tout au long du livre, et elles sont employées pour créer des structures de données plus avancées telles que des tables de hachage (chapitre 5).
- **Chapitre 3 :** vous en apprendrez plus sur la récursion, une technique pratique utilisée par de nombreux algorithmes (comme le tri rapide, couvert au chapitre 4).

D'après mon expérience, la notation Grand O et la récursion sont des sujets difficiles pour les débutants. J'ai donc ralenti l'allure et passé plus de temps sur ces sections.

Le reste du livre présente des algorithmes avec de vastes applications.

- **Techniques de résolution de problèmes :** couvertes dans les chapitres 4, 8 et 9. Si vous rencontrez un problème et ne savez pas comment le résoudre efficacement, essayez de diviser pour mieux régner (chapitre 4) ou tentez la programmation dynamique (chapitre 9). Alternativement vous pouvez réaliser qu'il n'y a pas de solution efficace et obtenir une réponse approximative en utilisant un algorithme glouton à la place (chapitre 8).
- **Tables de hachage :** couvertes au chapitre 5. Une table de hachage est une structure de données très utile. Elle contient des ensembles de paires clés et valeur, comme le nom d'une personne et son adresse e-mail, ou un nom d'utilisateur et le mot de passe associé. Il est difficile d'exagérer l'utilité des tables de hachage. Quand je veux résoudre un problème, les deux plans d'attaque avec lesquels je commence sont « Puis-je utiliser une table de hachage ? » et « Puis-je modéliser cela sous forme de graphe ? »
- **Algorithmes sur graphes :** traités dans les chapitres 6 et 7. Les graphes sont un moyen de modéliser un réseau : un réseau social, ou un réseau de routes, ou de neurones, ou tout autre ensemble de connexions. La recherche par parcours en largeur (chapitre 6) et l'algorithme de Dijkstra (chapitre 7) sont des moyens de trouver la distance la plus courte entre deux points d'un réseau : vous pouvez utiliser cette approche pour calculer les degrés de séparation entre deux personnes ou le chemin le plus court vers une destination.
- **Voisins les plus proches** (en anglais *K-nearest neighbors* ou KNN) : couvert au chapitre 10. Il s'agit d'un algorithme d'apprentissage automatique simple. Vous pouvez utiliser KNN pour créer un système de recommandations, un moteur OCR, un système pour prédire les valeurs boursières, tout ce qui implique de prédire une valeur (« Nous pensons qu'Adit attribuera à ce film 4 étoiles ») ou de classer un objet (« Cette lettre est un Q»).
- **Étapes suivantes :** le chapitre 11 passe en revue 10 algorithmes qui feraient une bonne lecture par la suite.

Comment utiliser ce livre

L'ordre et le contenu de ce livre ont été soigneusement conçus. Si un sujet vous intéresse, n'hésitez pas à vous lancer. Sinon, lisez les chapitres dans l'ordre, ils s'appuient les uns sur les autres.

Je recommande fortement d'exécuter le code des exemples vous-même. Je ne saurais trop insister sur cette partie. Tapez simplement mes exemples de code mot à mot ou téléchargez-les et exécutez-les. Vous en retiendrez beaucoup plus si vous le faites.

www.lienmini.fr/45331-python

Ou encore via notre site web : www.deboecksuperieur.com/site/345331. Je recommande également de faire les exercices de ce livre. Les exercices sont courts, généralement une minute ou deux, parfois 5 à 10 minutes. Ils vous aideront à vérifier votre façon de penser, afin que vous sachiez si vous faites fausse route avant d'être allé trop loin.

Tous les exemples de code de ce livre utilisent Python 2.7. Tout le code du livre est présenté dans une police à largeur fixe comme celle-ci pour le séparer du texte ordinaire. Des annotations de code accompagnent certaines des listes, mettant en évidence des concepts importants.

Je pense qu'on apprend mieux si on aime vraiment apprendre, alors amusez-vous et exécutez les exemples de code !

Qui devrait lire ce livre

Ce livre s'adresse à toute personne connaissant les bases du codage et souhaitant comprendre les algorithmes. Peut-être avez-vous déjà un problème de codage et essayez-vous de trouver une solution algorithmique. Ou peut-être voulez-vous comprendre à quoi servent les algorithmes. Voici une liste courte et non exhaustive de personnes qui trouveront probablement ce livre utile :

- les codeurs amateurs
- les étudiants des camps d'entraînement à la programmation
- les diplômés en informatique à la recherche d'une remise à niveau
- les diplômés de physique, mathématiques, etc. intéressés par la programmation

À propos de l'auteur

Aditya Bhargava est ingénieur logiciel chez Etsy, un site de vente en ligne pour des produits faits à la main. Il est titulaire d'une maîtrise en informatique de l'Université de Chicago. Il rédige également un blog technique illustré populaire sur adit.io.

Introduction aux algorithmes 1

Dans ce chapitre

- Vous trouverez les bases pour le reste du livre.
- Vous écrirez votre premier algorithme de recherche (recherche binaire).
- Vous apprendrez à parler du temps d'exécution d'un algorithme (notation Grand O).
- Vous vous initierez à une technique courante de conception d'algorithmes (la récursion).

Introduction

Un *algorithme* est un ensemble d'instructions pour accomplir une tâche. Chaque morceau de code pourrait être appelé un algorithme, mais ce livre couvre les exemples les plus intéressants. Les algorithmes inclus dans ce livre ont été choisis parce qu'ils sont rapides, ou qu'ils résolvent des problèmes intéressants, ou les deux. Voici quelques faits saillants :

- Le chapitre 1 parle de la recherche binaire et montre comment un algorithme peut accélérer votre code. Dans un exemple, le nombre d'étapes nécessaires passe de 4 milliards à 32 !

- Un appareil GPS utilise des algorithmes graphiques (comme vous l'apprendrez dans les chapitres 6, 7 et 8) pour calculer l'itinéraire le plus court jusqu'à votre destination.
- Vous pouvez utiliser la programmation dynamique (discutée au chapitre 9) pour écrire un algorithme d'intelligence artificielle (IA) qui joue aux dames.

Dans chaque cas, je décrirai l'algorithme et je vous donnerai un exemple. Ensuite, je parlerai du temps d'exécution de l'algorithme en notation Grand O. Enfin, j'explorerai quels autres types de problèmes pourraient être résolus par le même algorithme.

Ce que vous apprendrez sur les performances

La bonne nouvelle est qu'une implémentation de chaque algorithme de ce livre est probablement disponible dans votre langage préféré, vous n'avez donc pas à écrire chaque algorithme vous-même ! Mais ces implémentations sont inutiles si vous ne comprenez pas les compromis qui entrent en jeu. Dans ce livre, vous apprendrez à évaluer les compromis impliqués par différents algorithmes : devriez-vous utiliser un tri par fusion ou un tri rapide ? Faut-il utiliser un tableau ou une liste ? Le simple fait d'utiliser une structure de données différente peut faire une grande différence.

Ce que vous apprendrez sur la résolution de problèmes

Vous apprendrez des techniques pour résoudre des problèmes qui étaient peut-être hors de votre portée jusqu'à présent. Par exemple :

- Si vous aimez créer des jeux vidéo, vous pourrez écrire un système d'IA qui suit l'utilisateur à l'aide d'algorithmes graphiques.
- Vous apprendrez à créer un système de recommandations en utilisant les k plus proches voisins.
- Certains problèmes ne peuvent pas être résolus en un temps raisonnable ! La partie de ce livre qui parle des problèmes NP-complets vous montre comment identifier ces problèmes et proposer un algorithme qui vous donne une réponse approximative.

Plus généralement, à la fin de ce livre, vous connaîtrez certains des algorithmes les plus largement appliqués. Vous pourrez ensuite utiliser vos nouvelles connaissances pour en savoir plus sur des algorithmes plus spécifiques pour l'IA, les bases de données, etc. Ou vous pourrez relever de plus grands défis au travail.

Ce que vous devez savoir

Vous aurez besoin de connaître les bases de l'algèbre avant de commencer ce livre. En particulier, prenons cette fonction : f(x) = x × 2. Qu'est-ce que f(5) ? Si vous avez répondu 10, vous êtes prêt.

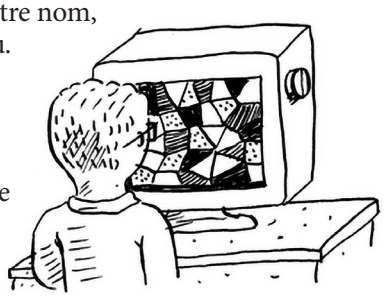
De plus, ce chapitre (et ce livre) sera plus facile à suivre si vous êtes familiarisé avec un langage de programmation. Tous les exemples du livre sont en Python. Si vous ne connaissez aucun langage de programmation et que vous souhaitez en apprendre un, choisissez Python, c'est parfait pour les débutants. Si vous connaissez un autre langage, comme Ruby, tout ira bien.

Recherche binaire

Supposons que vous cherchiez une personne dans l'annuaire téléphonique (quelle phrase démodée !). Leur nom commence par K. Vous pouvez commencer par le début et continuer à tourner les pages jusqu'à ce que vous arriviez aux K. Mais vous êtes plus susceptible de commencer à une page au milieu, car vous savez que les K vont être près du milieu de l'annuaire téléphonique.

Ou supposons que vous recherchiez un mot dans un dictionnaire et qu'il commence par O. Encore une fois, vous commencerez près du milieu.

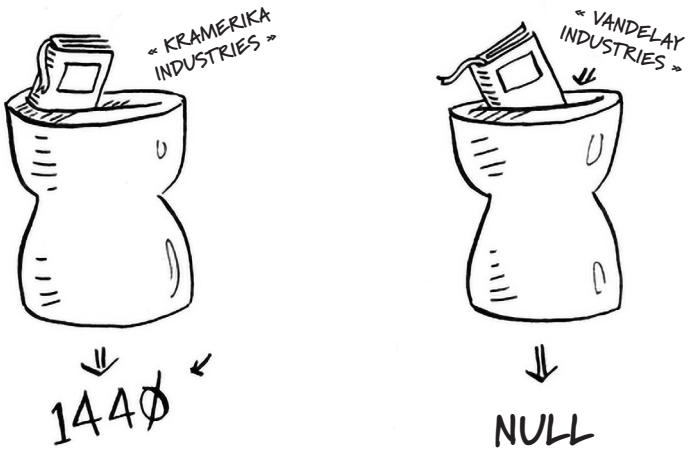
Supposons maintenant que vous vous connectiez à Facebook. Lorsque vous le faites, Facebook doit vérifier que vous avez un compte sur le site. Il doit donc rechercher votre nom d'utilisateur dans sa base de données. Supposons que votre nom d'utilisateur soit karlmageddon. Facebook pourrait commencer à partir des A et rechercher votre nom, mais il est plus logique qu'il commence quelque part au milieu.



C'est un problème de recherche. Et tous ces cas utilisent le même algorithme pour résoudre le problème : la *recherche binaire*.

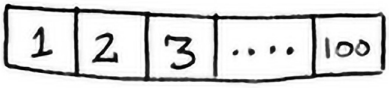
La recherche binaire est un algorithme ; son entrée est une liste triée d'éléments (j'expliquerai plus tard pourquoi elle doit être triée). Si un élément que vous recherchez se trouve dans cette liste, la recherche binaire renvoie la position où il se trouve. Sinon, la recherche binaire renvoie `null`.

Par exemple



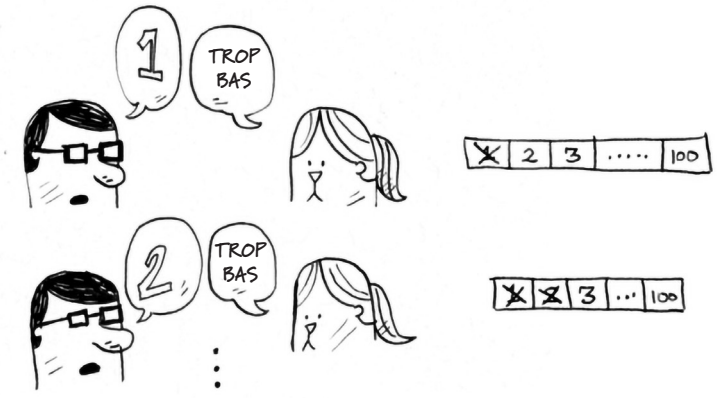
Recherche d'entreprises dans un annuaire au moyen de la recherche binaire

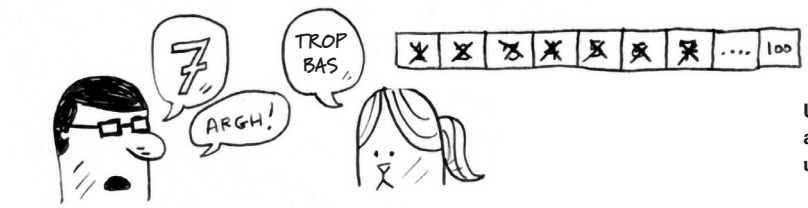
Voici un exemple du fonctionnement de la recherche binaire. Je pense à un nombre entre 1 et 100.



Vous devez essayer de deviner ce nombre en un minimum d'essais. À chaque supposition, je vous dirai si votre supposition est trop basse, trop élevée ou correcte.

Supposons que vous commenciez à deviner comme ceci : 1, 2, 3, 4 .... Voici comment cela se passerait.



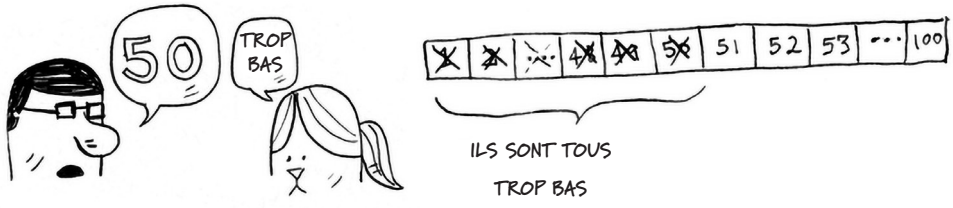


Une mauvaise approche pour deviner un nombre

C'est une *recherche simple* (peut-être que *recherche inepte* serait plus approprié !). À chaque supposition, vous n'éliminez qu'un seul nombre. Si mon nombre était 99, il vous faudrait 99 suppositions pour y arriver !

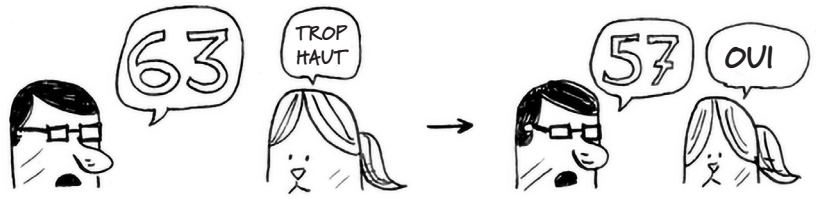
Une meilleure façon de rechercher

Voici une meilleure technique. Commencez par 50.

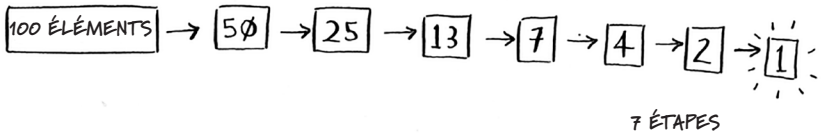


Trop bas, mais vous venez d'éliminer *la moitié* des nombres ! Maintenant, vous savez que les nombres de 1 à 50 sont trop bas. Prochaine tentative : 75.

Trop élevé, mais encore une fois, vous avez réduit de moitié les nombres restants ! *Avec la recherche binaire, vous essayez le nombre du milieu et éliminez la moitié des nombres restants à chaque fois.* Vient ensuite 63 (à mi-chemin entre 50 et 75).



C'est une recherche binaire. Vous venez d'apprendre votre premier algorithme ! Voici combien de nombres vous pouvez éliminer à chaque fois.



Éliminez la moitié des nombres à chaque fois avec la recherche binaire.

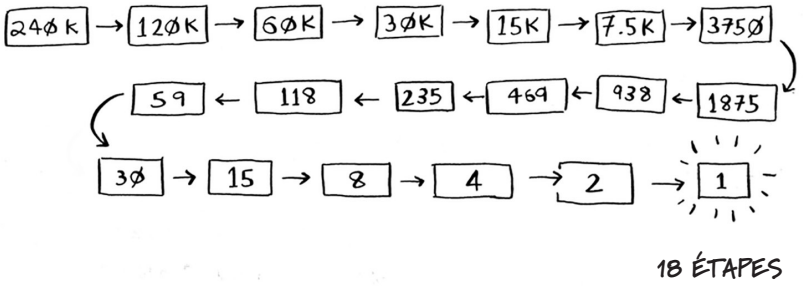
Quel que soit le nombre auquel je pense, vous pouvez deviner avec un maximum de sept suppositions, car vous éliminez tellement de nombres à chaque étape !

Supposons que vous cherchiez un mot dans le dictionnaire. Le dictionnaire compte 240 000 mots. *Dans le pire des cas,* combien d'étapes pensez-vous que chaque recherche prendra ?

RECHERCHE SIMPLE : _____ ÉTAPES

RECHERCHE BINAIRE : _____ ÉTAPES

Une recherche simple peut prendre 240 000 étapes si le mot que vous recherchez est le tout dernier du livre. À chaque étape de la recherche binaire, vous réduisez le nombre de mots de moitié jusqu'à ce qu'il ne vous reste qu'un seul mot.



***Les algorithmes, c'est plus simple avec un dessin !* est une approche didactique et attrayante des algorithmes et de l'informatique de base.**

Sans le savoir, nous utilisons des algorithmes dans la vie courante : recettes de cuisine, déclaration d'impôts, etc. Ce livre présente un grand nombre d'algorithmes testés et éprouvés par des programmeurs. Si vous voulez les comprendre, sans vous attarder sur des démonstrations de plusieurs pages, ce livre est fait pour vous.
Entièrement illustré, il facilite l'apprentissage et l'utilisation efficace des algorithmes les plus importants. L'auteur vous guide pas à pas en partant des exercices pour vous amener aux concepts.

- Vous y apprendrez **comment appliquer des algorithmes courants** aux problèmes de programmation pratiques auxquels vous êtes confrontés.
- Vous commencerez par des tâches comme **le tri et la recherche.**
- Vous **développerez vos compétences** et progressivement aborderez des problèmes plus complexes tels que la **programmation dynamique** et les **systèmes de recommandation.**

Chaque exemple comprend **des diagrammes utiles** et **des exemples de code** entièrement annotés en **Python**.

À la fin de ce livre, vous maitriserez des algorithmes applicables et saurez comment et quand les utiliser.

- Grand nombre d'algorithmes de recherche, de tri et de graphe
- Plus de 400 images avec des procédures pas à pas détaillées
- Exemples de code basés sur Python. Les codes à exécuter sont accessibles en ligne.

Ce manuel facile à lire et riche en images convient aux programmeurs autodidactes, aux ingénieurs ou à toute personne souhaitant approfondir les algorithmes.

Aditya Bhargava est un ingénieur logiciel avec une double formation en informatique et aux beaux-arts. Il a un blogue sur la programmation sur adit.io.

Traduction de l'anglais : Bernard Desgraupes† **:** ancien élève de l'École Normale Supérieure de la rue d'Ulm et agrégé de mathématiques, a mené une carrière de maître de conférences à l'Université Paris X. Il travaillait sur des projets de modélisation financière et de simulation.

ISBN : 978-2-8073-4533-1

deboeck SUPÉRIEUR MANNING

www.deboecksuperieur.com