Advanced Data Structures and Algorithms

Learn how to enhance data processing with more complex and advanced data structures

Abirami A
Priya R L

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55517-937

www.bpbonline.com

Dedicated to

Our beloved students

&

Our family members and friends

About the Authors

- **Mrs. Abirami A** has completed her M.E (Information Technology) from Mumbai University, and is currently pursuing Ph.D. (Information Technology), from Noorul Islam Centre for Higher Education, Kumaracoil, Tamil Nadu. Her research interests are Computer Networks, Cyber security, Network Forensics and Analysis of algorithms. She has 17 years of teaching experience and has published many papers in national and international conferences and journals and work as an editorial board member of the Editorial Board of various SCI/SCOPUS indexed journals.

  Asst. Professor III - Information Technology Department,

  Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu.

- **Mrs. Priya R. L** is currently working as an Assistant Professor in V.E.S Institute of Technology, Mumbai, having 18 years of teaching experience for undergraduate students of Computer Engineering and Information Technology disciplines in different Engineering Institutes. She obtained her Master's degree in Information Technology Engineering from Mumbai University. She has also worked as a Software Engineer in different firms in Chennai and Mumbai for 4 years. Her research interest is more into Artificial Intelligence, Internet of Things (IoT), Software engineering, Data Structures, Data Mining and Next Generation Networks. She has published more than 40 papers in various reputed SCI / SCOPUS indexed journals and international conferences.

  Assistant Professor, Department of Computer Engineering,

  V.E.S. Institute of Technology, Mumbai, India.

About the Reviewers

- **Mrs. Lifna C S**, Assistant Professor at the Vivekanand Education Society's Institute of Technology, Mumbai. Her research interests are Big Data Analytics, Social Analytics, Machine Learning, UX Design, Blockchain. She has around 26+ research papers published in international conferences & journals. She is an Innovation Ambassador of VESIT - IIC (Institute Innovation Cell) formed under MHRD from 2019 onwards. She is also a member of the AI Research Group established at VESIT, in association with LeadingIndia.ai at Bennett University.
- **Dr. Lakshmanaprakash S** completed his Ph.D. from Curtin University, Sarawak, Malaysia in 2019. He is currently working as Associate Professor in the Department of IT, BIT, and Erode, India. He has published many papers in various reputed national/International conferences and Journals. Dr. Lakshmanaprakash has also been a reviewer in various SCI Journals. His research interest mainly focuses on Cybersecurity and Machine Learning.

Acknowledgements

We wish to express our gratitude to all those people who assisted and helped us to complete this book. We would like to show our appreciation to the members of our student team (Riya Bhanushali, Amisha Das, Asiya Khan, Disha Shah, Jay Visaria, and Mansi Thakkar of Shah and Anchor Kutchhi Engineering College, Mumbai, and Ajay Nair from Vivekanand Education Society's Institute of Technology, Mumbai). They helped us to understand their needs for the preparation of the contents associated with this book during the Engineering course. First and foremost, I want to express my gratitude to our friends and family members for their unwavering support to complete the book successfully. Without their encouragement, we would never have been able to finish it.

We owe a debt of gratitude to the student team who accompanied us in writing this book. We gratefully acknowledge Dr. Lakshmanprakash S and Mrs. Lifna C S for their kind technical scrutiny of this book.

Our gratitude also extends to the BPB team, who were kind enough to give us enough time to complete the book's first section and to approve its publication.

Preface

This book covers a collection of complex algorithms and helps to face the challenges in algorithmic analysis. Analysis of algorithms and handling sophisticated data structures focus on the fundamentals of the computer programming field. The book highlights how to find the best optimal solution to a real-world problem using an appropriate algorithm. The book provides theoretical explanations and solved examples of most of the topics covered.

This book also introduces the importance of performance analysis of an algorithm, which helps to increase efficiency and reduces time and space complexity. It shows how to create and design a complex data structure. This book solves the basic understanding of greedy and dynamic programming. It also gives importance to various divide-and-conquer techniques. This book gives information about string-matching methods as well.

This book is divided into six chapters. The reader will go through advanced data structures, greedy and dynamic programming, optimal solutions, string matching using various techniques, and calculations of time and space complexity using Asymptotic notations. To help learners better comprehend the material, each topic is handled with appropriate examples. The specifics are mentioned as follows.

Chapter 1 emphasizes the basics of algorithmic analysis. It will discuss the need for analysis of algorithms and help us to choose a better suitable algorithm for a given problem statement. In algorithmic design, the complexity of an algorithm plays an important aspect to justify the design decisions. Accordingly, algorithm efficiency is measured from two perspectives such as time and space complexity. Hence, the major focus of this chapter is on various types of asymptotic notations used for the estimation of the time complexity of an algorithm and is discussed with examples.

Chapter 2 discusses different complex data structures that may be used to effectively tackle difficult situations. AVL Tree, Huffman Coding, Redblack Tree, and several more search trees are among the advanced data structures addressed.

Chapter 3 discusses the divide and conquer technique with the basic introduction and various methods that are involved in it with suitable examples. Divide and Conquer is the simplest and easiest technique of decomposing a larger problem into simpler problems, to solve any given problem statement.

Chapter 4 will cover information about various greedy algorithms such as the knapsack problem, optimal merge pattern, subset cover problem, and so on, in detail with various solved examples.

Chapter 5 discusses Dynamic Programming. It describes various classical computer science problems and their optimal solutions using dynamic programming approaches along with their applications. The need for dynamic algorithms and introduction to NP-Hard & NP-Complete are discussed with examples.

Chapter 6 describes different string-matching algorithms with suitable examples. The chapter also provides description of genetic algorithms.

Coloured Images

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/x0d84sg

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Analysis of Algorithm

Introduction

The chapter emphasizes the basics of algorithmic analysis. *Donald Knuth* defines the term *"analysis of algorithms"* as the common technique for theoretical estimation of required resources, that is used to provide a solution to any specific computational problem. It will discuss the need for analysis of algorithms and help us choose a more suitable algorithm for a given problem statement. In algorithmic design, complexity of an algorithm plays an important aspect in justifying the design decisions. Accordingly, algorithm efficiency is measured in two perspectives, such as time and space complexity. Hence, the major focus of this chapter is on various types of asymptotic notations used for the estimation of time complexity of an algorithm and is discussed with examples.

Structure

In this chapter, we will discuss the following topics:

- Analysis of algorithm
- Asymptotic Notations
- Time Complexity
- General Rules for time complexity calculation
- Recurrences

Objectives

This chapter discusses the basics of analysis of algorithm, the need for analysis of algorithms as well as the various notations, with examples. Apart from these, time complexity calculation is the major content focused on, in the chapter.

Analysis of algorithm

In this section, we give an overview of a generic framework on analysis of algorithms. It analyzes the efficiency of algorithms, in terms of space efficiency and time efficiency. To represent the complexity of an algorithm, asymptotic notations are a very crucial and important factor in designing algorithms. Therefore, various notations with solved examples are illustrated here.

Space complexity is defined as the amount of memory space required to execute an algorithm. Sometimes, space complexity is ignored as the space used is minimal. But time complexity refers to the amount of time required for the computation of an algorithm. Mostly, execution time depends on the various properties such as disk input/output speed, CPU speed, instructor set and so on. The calculation of time complexity and its rules with solved examples are also described in this chapter. It is followed by the recurrences in algorithm analysis and its three major types are discussed in the later sections of this chapter.

What is analysis of algorithms?

In general terms, analysis of algorithms discusses the efficiency of an algorithm. It tells the estimation of various resources required by an algorithm to crack a specific problem of computation. The resources are the necessary storage or the required time to execute a specific algorithm. The estimated running time of an algorithm is called time complexity and the estimated storage/memory needed for the execution of an algorithm is called space complexity.

Why to analyze algorithms?

Multiple solutions are available for a single problem; analysis will present the best algorithm to solve the given problem out of the multiple solutions. Even though the objective of the algorithms is to generate the expected output, the ways in which the outputs are generated, are different. The algorithm varies in the time and the space complexity. The various cases of analysis such as the worst, best and the average are performed with the help of asymptotic notations, to finalize the best algorithm.

Asymptotic notations

The main idea of asymptotic analysis is to have a measure of efficiency of algorithms, that does not depend on machine specific constants, and neither requires algorithms to be implemented and time taken by programs to be compared. Asymptotic notations are mathematical tools to represent time complexity of algorithms for asymptotic analysis. The following 3 asymptotic notations are mostly used to represent time complexity of algorithms.

Θ Notation

In this *theta* notation, the function bounds between the upper and the lower bound, and so it is called as an average case of T(n).

Average-case T(n) = average expected time of algorithm over all inputs of size n.

$$\Theta(g(n)) = \left\{ \begin{array}{l} f(n): \text{there exist positive constants } c_1, c_2, \text{and } n_0 \text{ s.t.} \\ \quad c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

Figure 1.1 features the average case Theta Θ Notation:

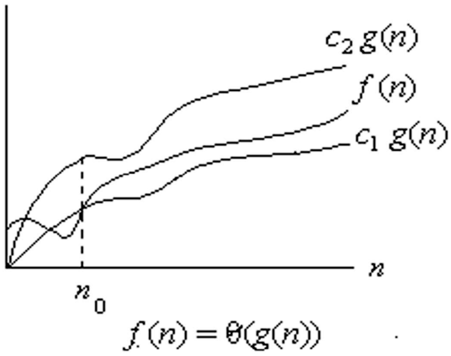

Figure 1.1: *Average case Theta Θ Notation*

Big O Notation

In this Big O notation, the function defines an upper bound, and so it is called the worst case of T(n).

$$T(n) = \text{maximum time of algorithm on any input of size } n.$$

$$O(g(n)) = \left\{ \begin{array}{l} f(n): \text{there exist positive constants } c \text{ and } n_0 \text{ s.t.} \\ f(n) \leq cg(n) \;\; \text{for all } n \geq n_0 \end{array} \right\}$$

Figure 1.2 features the worst-case Big O Notation:

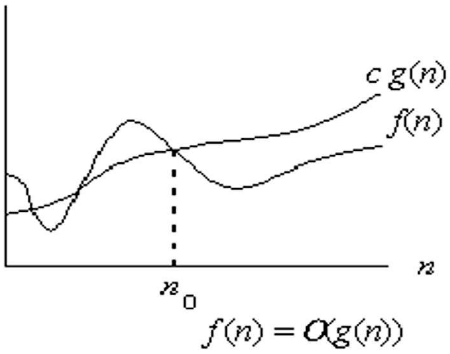

Figure 1.2: *Worst case Big O Notation*

Ω (Omega) Notation

In this Ω notation, the function defines a lower bound, and so it is called the best case of T(n).

T(n) = minimum time of algorithm on any input of size n.

It is a slow algorithm that works fast on some input.

$$\Omega(g(n)) = \left\{ \begin{array}{c} f(n): \text{there exist positive constants } c \text{ and } n_0 \text{ s.t.} \\ cg(n) \le f(n) \text{ for all } n \ge n_0 \end{array} \right\}$$

Figure 1.3 features the best-case Omega Ω Notation:

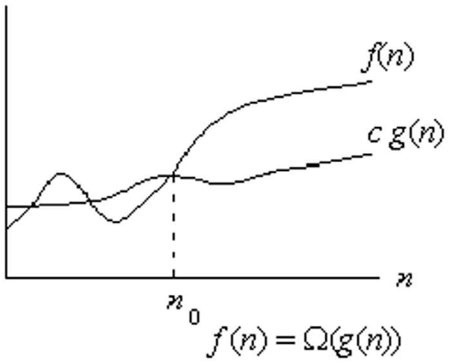

Figure 1.3: *Best case Omega Ω Notation*

Table 1.1 features the Asymptotic notations:

Asymptotic Notations	Case	Function Bound
Big O	Worst Case	Upper bound
Omega Ω	Best Case	Lower Bound
Theta Θ	Average Case	Tight Bound

Table 1.1: *Asymptotic Notations*

Example 1

Find Upper Bound, Lower Bound and Tight Bound for the following function 2n+5.

Solution 1:

Consider f(n)=2n+5,

g(n)=n {consider the highest degree of f(n)}

Refer to *Table 1.2:*

Lower Bound	Tight Bound	Upper Bound
Omega Ω	Theta Θ	Big O
Best Case	Average case	Worst case
2n	2n	3n

Table 1.2: *The constant value consideration for the asymptotic notations*

Big O:

*f(n)<=c*g(n)*

*2n+5<=c*n*

C=3// as per Table 1.2

*2n+5<=3*n*

For n=1, 7<=3 → False

For n=2, 9<=6 → False

For n=3, 11<=9 → False

For n=4, 13<=12 → False

For n=5, 15<=15 → True

Therefore f(n)=O(g(n))

2n+5=O(n) for all n>=5, C=3

Omega (Ω):

*f(n)>=c*g(n)*

*2n+5>=c*n*

C=2

*2n+5>=2*n*

For n=1, 7>=2 → True

Therefore f(n)= Ω(g(n))

2n+5= Ω(n) for all n>=1, C=2

Theta (Θ):

*C1*g(n) <= f(n) <= c2*g(n)*

*C1*n <= 2n+5 <= c2*n*

C1=2, c2=3

*2*n <= 2n+5 <= 3*n*

For n=1, 2 <= 7 <=3 → false

For n=2, 4 <= 9 <=6 → False

For n=3, 6 <= 11 <=9 → false

For n=4, 8 <= 13 <=12 → false

For n=5, 10<= 15 <=15 → true

Therefore f(n)=Θ(g(n))

2n+5= Θ(n) for all n>=5, c1=2, c2=3

Example 2

Find Upper Bound, Lower Bound and Tight Bound for the following function 3n+2

Solution 2:

Big O:

f(n)=O g(n);

3n+2 =O(n) for all n>=2, c=4

Omega Ω :

F(n)= Ω g(n);

3n+2= Ω(n) for all n>=1, c=3

Theta Θ:

F(n)= Θ g(n);

3n+2=Θ(n) for all n>=2 , c1=3 & c2=4.