

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/334708010>

Fundamentals of Algorithm

Book · July 2018

CITATIONS

0

READS

16,957

2 authors, including:



[Sheetal Kashid](#)

Tata Institute of Social Science

5 PUBLICATIONS 0 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Working on Research Project Connected Learning Initiative (CLIX) at Tata Institute of Social Science [View project](#)



Personalized web search in web mining [View project](#)

FUNDAMENTALS OF ALGORITHMS

Archana Jadhav
Sheetal Kashid



Himalaya Publishing House

ISO 9001:2008 CERTIFIED

Fundamentals of Algorithms

As per the New Syllabus of Mumbai University for
S.Y.B.Sc. (Computer Science), Semester IV, 2017-18



Archana Jadhav

MCS (Pune University)

*Coordinator, Dept. of Computer Science,
N.G. Acharya & D.K. Marathe College,
Chembur, Mumbai.*

Sheetal Kashid

M.E. (Computer Science)

*Assistant Professor, Dept. of Computer Science,
N.G. Acharya & D.K. Marathe College,
Chembur, Mumbai.*



Himalaya Publishing House

ISO 9001:2008 CERTIFIED

© **Authors**

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise without the prior written permission of the publisher.

First Edition : 2018

Published by	: Mrs. Meena Pandey for Himalaya Publishing House Pvt. Ltd. , "Ramdoot", Dr. Bhalerao Marg, Girgaon, Mumbai - 400 004. Phone: 022-23860170, 23863863; Fax: 022-23877178 E-mail: himpub@vsnl.com; Website: www.himpub.com
Branch Offices	:
New Delhi	: "Pooja Apartments", 4-B, Murari Lal Street, Ansari Road, Darya Ganj, New Delhi - 110 002. Phone: 011-23270392, 23278631; Fax: 011-23256286
Nagpur	: Kundanlal Chandak Industrial Estate, Ghat Road, Nagpur - 440 018. Phone: 0712-2738731, 3296733; Telefax: 0712-2721216
Bengaluru	: Plot No. 91-33, 2nd Main Road, Seshadripuram, Behind Nataraja Theatre, Bengaluru - 560 020. Phone: 080-41138821; Mobile: 09379847017, 09379847005
Hyderabad	: No. 3-4-184, Lingampally, Besides Raghavendra Swamy Matham, Kachiguda, Hyderabad - 500 027. Phone: 040-27560041, 27550139
Chennai	: New No. 48/2, Old No. 28/2, Ground Floor, Sarangapani Street, T. Nagar, Chennai - 600 012. Mobile: 09380460419
Pune	: "Laksha" Apartment, First Floor, No. 527, Mehunpura, Shaniwarpeth (Near Prabhat Theatre), Pune - 411 030. Phone: 020-24496323, 24496333; Mobile: 09370579333
Lucknow	: House No. 731, Shekhupura Colony, Near B.D. Convent School, Aliganj, Lucknow - 226 022. Phone: 0522-4012353; Mobile: 09307501549
Ahmedabad	: 114, "SHAIL", 1st Floor, Opp. Madhu Sudan House, C.G. Road, Navrang Pura, Ahmedabad - 380 009. Phone: 079-26560126; Mobile: 09377088847
Ernakulam	: 39/176 (New No. 60/251) 1st Floor, Karikkamuri Road, Ernakulam, Kochi - 682 011. Phone: 0484-2378012, 2378016; Mobile: 09387122121
Bhubaneswar	: Plot No. 214/1342, Budheswari Colony, Behind Durga Mandap, Bhubaneswar - 751 006. Phone: 0674-2575129; Mobile: 09338746007
Kolkata	: 108/4, Beliaghata Main Road, Near ID Hospital, Opp. SBI Bank, Kolkata - 700 010. Phone: 033-32449649; Mobile: 07439040301
DTP by	: Sagar Yadav
Printed at	: Geetanjali Press Pvt. Ltd., Nagpur. On behalf of HPH.

Dedication

I would like to dedicate this book to my dearest daughter Anushka and son Atharva for their unconditional love which has made me stronger, better and more fulfilled than I could have ever imagined.

Archana Jadhav

I would like to dedicate this book to my father Vishwas Kashid who motivated me by his words '**Success is a result of continuous daily efforts**'. I would also like to dedicate this book to my Mother Rukhamini Kashid, whose loving spirit sustains me still. Last but not the least, I express my gratitude towards my dearest brother and sister for their love and support. I would also like to thank my co-author Archana Jadhav for believing in me and giving me this golden opportunity.

Sheetal Kashid

We would like to express our sincere thanks to Mr. S.K. Srivastava of Himalaya Publishing House Pvt. Ltd., for giving us the opportunity to write this book.



Preface

It gives us great pleasure to present a book “**Fundamentals of Algorithms**” for S.Y.B.Sc. (Sem. IV) Computer Science. This book is based on new syllabus prescribed by Mumbai University from 2017-18. The main topics have been divided into subtopics for easy understanding. A number of solved examples have been included in the book with their implementation in Python. ‘Mcq’s are provided to enhance aptitude skills.’

We hope this book will fulfil all your needs for the subject. As a step towards betterment and improvement of the book, suggestions from teachers and students are welcome.

Please send correspondence to archana.sjd@gmail.com and kashid.sheelal3@gmail.com

Finally, we would like to acknowledge our sincere respect and gratitude to Kiran Gurbani (Head of Computer Science and IT Department, R.K. Talreja College, Ulhasnagar) for reviewing this book thoroughly and providing an environment which stimulates new thinking and innovations and her support which helped us for bringing out this book in time.

We are thankful to Shri S.K. Srivastava and staff members of Himalaya Publishing House for their cooperation.

Authors

Syllabus

Fundamentals of Algorithms

Objectives:

1. To understand basic principles of algorithm design and why algorithm analysis is important
2. To understand how to implement algorithms in Python
3. To understand how to transform new problems into algorithmic problems with efficient solutions
4. To understand algorithm design techniques for solving different problems

Unit No.	Modules/Units
Unit 1	Introduction to algorithm, Why to analyse algorithm?, Running time analysis, How to compare algorithms?, Rate of Growth, Commonly Used Rates of Growth, Types of Analysis, Asymptotic Notation, Big-O Notation, Omega- Ω Notation, Theta- θ Notation, Asymptotic Analysis, Properties of Notations, Commonly used Logarithms and Summations, Performance characteristics of algorithms, Master Theorem for Divide and Conquer, Divide and Conquer Master Theorem: Problems and Solutions, Master Theorem for Subtract and Conquer Recurrences, Method of Guessing and Confirming.
Unit 2	<p>Tree Algorithms: What is a Tree? Glossary, Binary Trees, Types of Binary Trees, Properties of Binary Trees, Binary Tree Traversals, Generic Trees (N-ary Trees), Threaded Binary Tree Traversals, Expression Trees, Binary Search Trees (BSTs), Balanced Binary Search Trees, AVL (Adelson-Velskii and Landis) Trees.</p> <p>Graph Algorithms: Introduction, Glossary, Applications of Graphs, Graph Representation, Graph Traversals, Topological Sort, Shortest Path Algorithms, Minimal Spanning Tree.</p> <p>Selection Algorithms: What are Selection Algorithms? Selection by Sorting, Partition-based Selection Algorithm, Linear Selection Algorithm - Median of Medians Algorithm, Finding the K Smallest Elements in Sorted Order.</p>
Unit 3	<p>Algorithms Design Techniques: Introduction, Classification, Classification by Implementation Method, Classification by Design Method.</p> <p>Greedy Algorithms: Introduction, Greedy Strategy, Elements of Greedy Algorithms, Advantages and Disadvantages of Greedy Method, Greedy Applications, Understanding Greedy Technique.</p>

	<p>Divide and Conquer Algorithms: Introduction, What is Divide and Conquer Strategy?, Divide and Conquer Visualization, Understanding Divide and Conquer, Advantages of Divide and Conquer, Disadvantages of Divide and Conquer, Master Theorem, Divide and Conquer Applications.</p> <p>Dynamic Programming: Introduction, What is Dynamic Programming Strategy?, Properties of Dynamic Programming Strategy, Problems which can be solved using Dynamic Programming, Dynamic Programming Approaches, Examples of Dynamic Programming Algorithms, Understanding Dynamic Programming, Longest Common Subsequence.</p>
--	---

List of Practicals

1. Write Python program to perform matrix multiplication. Discuss the complexity of algorithm used.
2. Write Python program to sort n names using Quick sort algorithm. Discuss the complexity of algorithm used.
3. Write Python program to sort n numbers using Merge sort algorithm. Discuss the complexity of algorithm used.
4. Write Python program for inserting an element into binary tree.
5. Write Python program for deleting an element (assuming data is given) from binary tree.
6. Write Python program for checking whether a given graph G has simple path from source s to destination d. Assume the graph G is represented using adjacent matrix.
7. Write Python program for finding the smallest and largest elements in an array A of size n using Selection algorithm. Discuss Time complexity.
8. Write Python program for finding the second largest element in an array A of size n using Tournament Method. Discuss Time complexity.
9. Write Python program for implementing Huffman Coding Algorithm. Discuss the complexity of algorithm.
10. Write Python program for implementing Strassen's Matrix multiplication using Divide and Conquer method. Discuss the complexity of algorithm.

Question Paper Pattern

(Time: 2½ Hours)

[Total Marks: 75]

- N.B.** 1. All questions are compulsory.
2. Figures to the right indicate marks.

Q.1	Attempt All (Each of 5Marks) (a) Select correct answer from the following: (b) Fill in the blanks: (c) Define the following:	(15M) (5) (5) (5)
Q.2	Attempt the following (Any THREE) from Unit I (a) (b) (c) (d) (e) (f)	(15M)
Q.3	Attempt the following (Any THREE) from Unit II (a) (b) (c) (d) (e) (f)	(15M)
Q.4	Attempt the following (Any THREE) from Unit III (a) (b) (c) (d) (e) (f)	(15M)
Q.5	Attempt the following (Any THREE) from Unit I, II & III (a) (b) (c) (d) (e)	(15M)

Contents

Chapter No.	Name	Page No.
UNIT I		
1	Introduction to Algorithm	1 – 12
2	Asymptotic Analysis	13 – 23
3	Recurrence Relations	24 – 33
UNIT II		
4	Sorting	34 – 50
5	Trees	51 – 71
6	Graphs	72 – 108
UNIT III		
7	Algorithm Design Techniques	109 – 111
8	Greedy Algorithms	112 – 120
9	Divide and Conquer Algorithms	121 – 126
10	Dynamic Programming	127 – 139
	Practicals	140 – 164
	Multiple Choice Questions	165 – 173
	References	174 – 174



UNIT I

1

CHAPTER

Introduction to Algorithm

Structure:

- 1.1 What is an Algorithm?
- 1.2 Algorithm Analysis
- 1.3 Running Time Analysis
- 1.4 Comparing Algorithms
- 1.5 Rate of Growth
- 1.6 Asymptotic Notations

1.1 WHAT IS AN ALGORITHM?

In Mathematics or in Computer Science or in other related subjects, algorithm concept is used. Each algorithm is a list of well-defined instructions that completes or executes a given task.

For example: Consider the algorithm that tries to check whether battery is properly working or not.

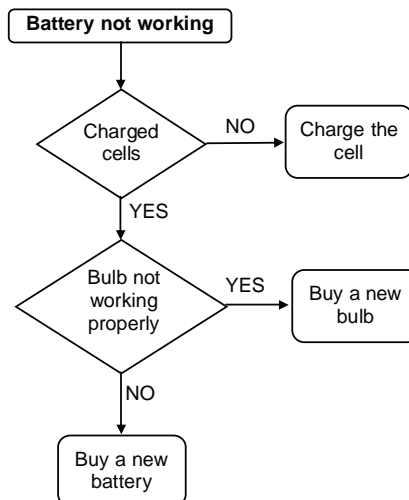


Fig. 1.1: Flowchart for checking battery



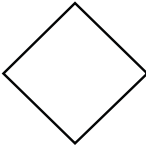


Flowchart

Flowchart is a graphical representation of an algorithm. To represent algorithm graphically, we use a flowchart. In flowchart of an algorithm, we use various types of shapes to implement the steps and these shapes connected via the arrows. The above flowchart shows all possible situations of battery, whether it is working or not.

First, we check the cells used in the battery whether it is charged or not. If not, then charge the cell. If cells are charged, check whether the bulb is working properly or not. If it is not working properly, then replace the bulb. If bulb is in working condition, the problem is somewhere in the battery itself, then buy the new bulb.

Symbolic Representation of Flowchart

American National Symbols Institute (ANSI) developed various types of symbols to represent an algorithm graphically. Table 1.1 represents the symbols used to implement flowchart.

Table 1.1		
Symbol/Shape	Name	Description
	Terminal	It is an oval rectangle, which represents the beginning or ending of a program. It is a starting and ending block.
	Process	It is a rectangle, which represents the collection of instructions to be executed.
	Decision	It is presented as a diamond. It is a decision making shape which decides the outcome based on the given condition.
	Input/Output	Represents a parallelogram, which is input statement to accept the user input and display the result as output statement.
	Flow Line	Represents an arrow to determine the flow of instructions.

Program

Program is a group of instructions performed to complete a particular task by the computer. It is execution of the algorithm.

Algorithm

An algorithm defined as a sequence of definite and effective instructions, which terminates with production of correct output with given input. It is a step-by-step instruction for solving a particular task in finite amount of time. Algorithm is an effective method for solving a problem expressed as finite sequence of instructions.

All algorithms must satisfy the following five conditions:

1. **Input:** There are zero or more quantities supplied as input to algorithm.
2. **Output:** By using inputs, that are extremely supplied an algorithm produce atleast one quantity as output.
3. **Definiteness:** The instructions used in algorithm must be clear and certain, i.e., unambiguous.
4. **Finiteness:** An algorithm must terminate after some finite number of steps. For all cases, i.e., every algorithm should have some termination or end-point after finite number of steps.
5. **Effectiveness:** To complete the task, each instruction must be basic that is human being can trace the instruction by using paper and pencil method.

For example: Write an algorithm and draw a flowchart to add two numbers and store the result.

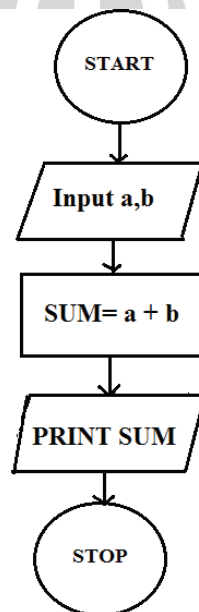


Fig. 1.2: Flowchart for addition of two numbers

Algorithm for addition of two numbers:

Step 1: Start

Step 2: Take two numbers a and b as input from the user

4 • Fundamentals of Algorithms

Step 3: Add two numbers and store it in the third variable SUM

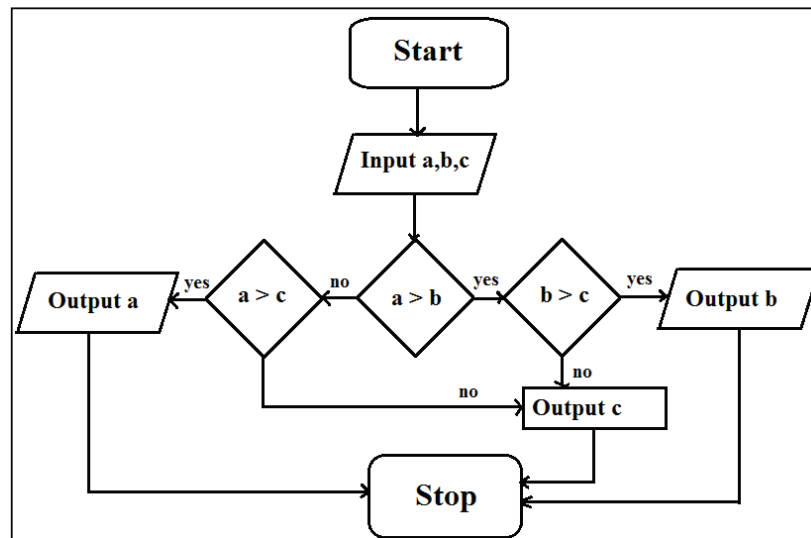
Step 4: Display SUM

Step 5: Stop

Data Structure and Algorithm

Data structure is a logical or mathematical model of particular organization of data. The mathematical model explains how the data is stored in memory and which operations are possible on it. Both data structure and algorithm work together. When set of statements do not suggest basic steps to solve the problem, it is called as procedure. For example, a program is always an algorithm except a slight difference that a program need not to be terminate.

For example: Write an algorithm and draw a flowchart to find the largest of three numbers.



Algorithm:

Step 1: Start

Step 2: Input a, b, c.

Step 3: If $a > b$, goto step4. Otherwise goto step5.

Step 4: If $b > c$, goto step6. Otherwise goto step8.

Step 5: If $a > c$, goto step7. Otherwise goto step8.

Step 6: b is the largest, goto step9.

Step 7: a is the largest, goto step9.

Step 8: c is the largest, goto step9.

Step 9: Stop

1.2 ALGORITHM ANALYSIS

Analysing the algorithm means solving the problem statement of an algorithm with different approaches. It also means to check for the best algorithm. Analysis of an algorithm performed in two phases:

1. **Prior estimate:** It is concerned with theoretical mathematical calculation that uses asymptotic notations for its representation.
2. **Posterior estimate:** It is concerned with writing a program and therefore depends on operating system, compiler, etc.

1.3 RUNNING TIME ANALYSIS

The running time of an algorithm for particular input based upon the number of operations executed. The greater the number of operations performed, the longer the running time of an algorithm. When a user writes a program, he is concerned about the resource requirement of the program. Algorithm analysis is determination of time, amount of memory utilized and other resources by the system to execute a particular task. Running time analysis increases the execution time (number of computational steps) of an algorithm depending on the increase in its input size.

Consider the input size denoted as n , and then the running time of that problem is function $f(n)$. Running time of an algorithm measured using following steps:

1. **Computing primary functions:** Primary functions are instructions with fixed implementation time. These instructions include:
 - (a) Assignment of variables to objects
 - (b) Performing arithmetic operations like addition of numbers, subtraction of numbers, etc.
 - (c) Comparing two numbers
 - (d) Defining a function and calling that function
2. **Computing operation of input size n :** Growth rate represented in algorithm as function $f(n)$.

1.4 COMPARING ALGORITHMS

Algorithms compared based on its time complexity and space complexity. Algorithms with better time complexity and less memory space is better algorithm.

Performance analysis of an algorithm depends on two factors:

1. **Space Complexity:** The amount of memory space required by an algorithm to run to its completion.
2. **Time Complexity:** The amount of time required by an algorithm to run to its completion. Number of step count taken by algorithm to complete its task calculates time complexity of an algorithm.

There are three types of step count:

- (a) **Best case:** The best-case step count the minimum number of steps executed for given parameters.
- (b) **Worst case:** The worst-case step count the maximum number of steps executed for given parameters.
- (c) **Average case:** The average case step count the average number of steps executed for given parameters.

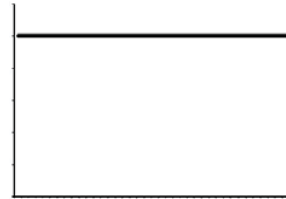
1.5 RATE OF GROWTH

Growth rate is used to analyse the algorithm. The greater the amount of data, larger the amount of resources required by an algorithm. Therefore, there is a resource growth rate for a piece of code in the form of function $f(n)$. Commonly used rates of growth to analyse the algorithm are:

1. **Constant function:** A constant resource need is one where the resource need does not grow. The simplest function with some fixed constant c such as $c = 10$ or $c = 2$, etc. The constant function is calculated as:

$$f(n) = c$$

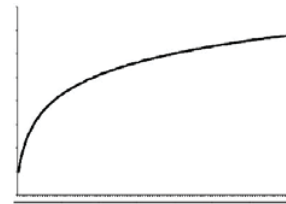
i.e., for any number n , the constant function $f(n)$ assigns the value c . The constant function is used to add two numbers, initialising value to a variable, comparing two numbers. The graph of such growth rate is represented by a horizontal line.



2. **Logarithmic function:** Logarithmic growth rate is a growth rate where resource need increases by one unit each time the data is doubled. The logarithmic function is calculated as:

$$f(n) = \log_b n \text{ for constant } b > 1$$

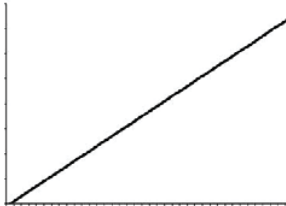
This function is defined as $x = \log_b n$ iff $b^x = n$, where value b is called base of the logarithm.



3. **Linear function:** Another simple important function is linear function calculated as:

$$f(n) = n$$

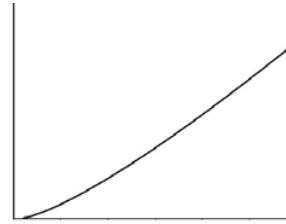
Given an input value n , the linear function f assigns the value n itself. A linear growth rate is a growth rate where the resource needs and amount of data directly proportional to each other. The growth rate of linear function is represented by a horizontal line. The example include comparing a number to each and every element of size n require n comparisons.



4. **N-Log-N function:** This type of function is calculated as:

$$f(n) = n \log n$$

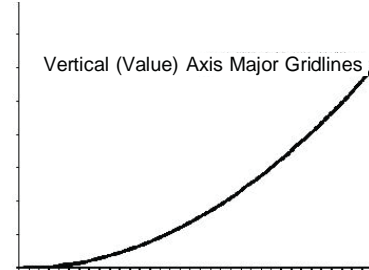
The function that assigns to an input n , the value of n times the logarithm base-two of n . A log linear growth rate is slightly curved, i.e., for lower values than higher ones. The example include possible algorithm for sorting n numbers.



5. **Quadratic function:** The quadratic function is calculated as:

$$f(n) = n^2$$

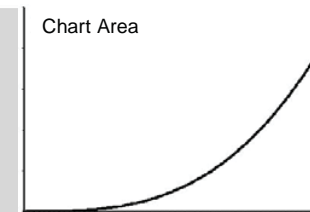
Given an input value n , the function f assigns the product of n with itself. The example includes algorithm with nested loops, where inner loop implements linear number of instructions and outer loop implemented linear number of times.



6. **Cubic function:** The cubic function is calculated as:

$$f(n) = n^3$$

which assigns to input value n the product of n with itself three times. The example includes Matrix Multiplication with three loops.



7. **Exponential function:** The exponential growth rate is one where extra unit of data requires double amount of resources. It is calculated as:

$$f(n) = b^n$$

where b is a constant called base and n is called exponent.



1.6 ASYMPTOTIC NOTATIONS

Asymptotic notations represent the limiting behaviour concerned with the increase in running time of algorithm with size of input as a limit. There are three types of asymptotic notations:

1. **Big-O Notation:** Let $f(n)$ and $g(n)$ be positive integers. This notation is represented as:

$$f(n) = O(g(n))$$

iff there exists two positive constants c and n_0 such that $f(n) \leq cg(n)$, i.e., $f(n)$ is asymptotically less than or equal to $g(n)$ for all values of $n \geq n_0$ where Big-O represents upper bound.

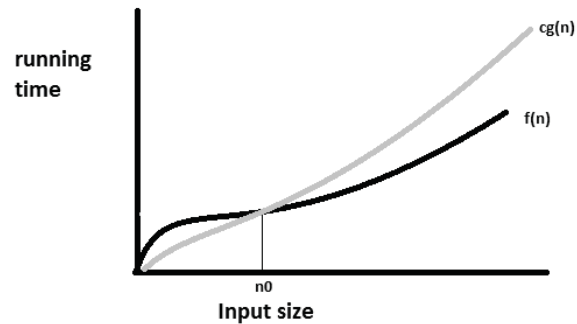


Fig. 1.3: Graph representing Big-O Notation where function $f(n) = O(g(n))$, since $f(n) \leq cg(n)$ where $n \geq n_0$

Examples Based on Big-O Notation

1.6.1 Let the function $f(n) = 3n + 3$. So, prove that $f(n) = O(n)$ and identify the function type.

Proof: Assume $n = g(n)$

Since Big-O notation formula includes,

$$f(n) \leq cg(n)$$

$$3n + 3 \leq c(n) \text{ for all } n \geq n_0.$$

Let $n = 1, c = 4$ $(3(1) + 3) \leq 4(1)$

$$6 \leq 4, \text{ so this condition fails for } n = 1.$$

Let $n = 2, c = 4$ $(3(2) + 3) \leq 4(2)$

$$9 \leq 8, \text{ so this condition again fails for } n = 2$$

Let $n = 3, c = 4$ $(3(3) + 3) \leq 4(3)$

$$12 \leq 12, \text{ so this condition pass for } n = 3$$

Let $n = 4, c = 4$ $(3(4) + 3) \leq 4(4)$

$$15 \leq 16, \text{ so this condition pass for } n = 4$$

Now, we can say that $3n + 3 \leq cn$ for all values of $n \geq 3$ and $c = 4$.

Hence, function $f(n) = O(n)$ for $n \geq 3$ is a linear function.

1.6.2 Let the function $f(n) = 10n^2 + 5n + 2$. So, prove that $f(n) = O(n^2)$ and identify the function type.

Proof: Assume $n^2 = g(n)$

Since Big-O notation formula includes,

$$f(n) \leq c \cdot g(n)$$

$$10n^2 + 5n + 2 \leq c(n^2) \text{ for all } n \geq n_0$$

Let $n = 1, c = 100$ $(10(1) + 5(1) + 2) \leq 100(1)$

$17 \leq 100$, so this condition pass for $n = 1$.

Let $n = 2$, $c = 100$ $(10(2^2) + 5(2) + 2) \leq 100(2^2)$

$52 \leq 200$, so this condition pass for $n = 2$.

Let $n = 3$, $c = 100$ $(10(3^2) + 5(3) + 2) \leq 100(3^2)$

$107 \leq 900$, so this condition pass for $n = 3$.

Now, we can say that $10n^2 + 5n + 2 \leq cn^2$ for all values of $n \geq 1$ and $c = 100$.

Hence, function $f(n) = O(n^2)$ for $n \geq 1$ is a quadratic function.

1.6.3 Let the function $f(n) = 5\log n + 6$. So, prove that $f(n) = O(\log n)$ and identify the function type.

Proof: Assume $\log n = g(n)$

Since Big-O notation formula includes,

$$f(n) \leq cg(n)$$

$$5 \log n + 6 \leq c(\log n) \text{ for all } n \geq n_0$$

Let $n = 1$, $c = 100$ $5 \log 1 + 6 \leq 100 \cdot \log 1$

$$5(0) + 6 \leq 100(0)$$

$6 \leq 0$, so this condition fails for $n = 1$.

Let $n = 2$, $c = 100$ $5 \log 2 + 6 \leq 100 \cdot \log 2$

$$5(0.30) + 6 \leq 100(0.30)$$

$7.5 \leq 30$, so this condition fails for $n = 2$.

Now, we can say that $5 \log n + 6 \leq c \cdot \log n$ for all values of $n \geq 2$ and $c = 100$ since, for $n = 1$ $\log n = 0$.

Hence, function $f(n) = O(\log n)$ for $n \geq 2$ is a logarithmic function.

2. Omega Notation: Let $f(n)$ and $g(n)$ be positive integers. This notation is represented as:

$$f(n) = \Omega(g(n))$$

iff there exists two positive constants c and n_0 such that $f(n) \geq cg(n)$, i.e., $f(n)$ is asymptotically greater than or equal to $g(n)$ for all values of $n \geq n_0$ where Ω represents lower bound.

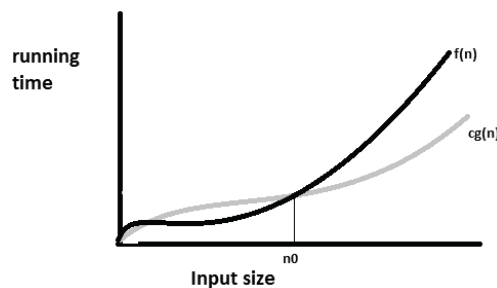


Fig. 1.4: Graph representing Omega Notation where function $f(n) = \Omega(g(n))$, since $f(n) \geq cg(n)$ where $n \geq n_0$

Examples Based on Omega Notation**1.6.4 Let the function $f(n) = 3n + 3$. So, prove that $f(n) = \Omega(n)$ and identify the function type.****Proof:** Assume $n = g(n)$

Since Omega notation formula includes,

$$f(n) \geq cg(n)$$

$$3n + 3 \geq c(n) \text{ for all } n \geq n_0$$

$$\text{Let } n = 1, c = 1 \quad (3(1) + 3) \geq 1(1)$$

$$6 \geq 1, \text{ so this condition pass for } n = 1.$$

$$\text{Let } n = 2, c = 1 \quad (3(2) + 3) \geq 1(2)$$

$$9 \geq 2, \text{ so this condition pass for } n = 2.$$

$$\text{Let } n = 3, c = 1 \quad (3(3) + 3) \geq 1(3)$$

$$12 \geq 3, \text{ so this condition pass for } n = 3.$$

Now, we can say that $3n + 3 \geq cn$ for all values of $n \geq 1$ and $c = 1$.Hence, function $f(n) = \Omega(n)$ for $n \geq 1$ is a linear function.**1.6.5 Let the function $f(n) = n^3 + 10n^2 + 6$. So, prove that $f(n) = \Omega(n^3)$ and identify the function type.****Proof:** Assume $n^3 = g(n)$

Since Omega notation formula includes,

$$f(n) \geq cg(n)$$

$$n^3 + 10n^2 + 6 \geq c(n^3) \text{ for all } n \geq n_0$$

$$\text{Let } n = 1, c = 1 \quad (1^3 + 10(1^2) + 6) \geq 1(1^3)$$

$$17 \geq 1, \text{ so this condition pass for } n = 1.$$

$$\text{Let } n = 2, c = 1 \quad (2^3 + 10(2^2) + 6) \geq 1(2^3)$$

$$54 \geq 8, \text{ so this condition pass for } n = 2.$$

$$\text{Let } n = 3, c = 1 \quad (3^3 + 10(3^2) + 6) \geq 1(3^3)$$

$$123 \geq 27, \text{ so this condition pass for } n = 3.$$

Now, we can say that $n^3 + 10n^2 + 6 \geq cn^3$ for all values of $n \geq 1$ and $c = 1$.Hence, function $f(n) = \Omega(n^3)$ for $n \geq 1$ is a cubic function.**3. Theta Notation:** Let $f(n)$ and $g(n)$ be positive integers. This notation is represented as:

$$f(n) = \Theta(g(n))$$

 $f(n)$ is a Big Theta of $g(n)$.

Iff $f(n)$ is Big O of $g(n)$ and $f(n)$ is Big Omega of $g(n)$, i.e., $c'g(n) \leq f(n) \leq c''g(n)$, i.e., $f(n)$ is asymptotically equal to $g(n)$ for all values of $n \geq n_0$.

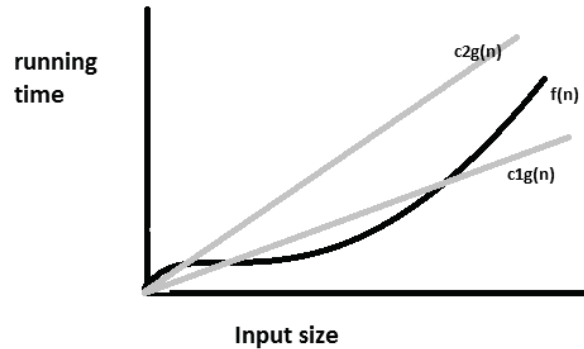


Fig. 1.5: Graph representing Theta Notation where function $f(n) = \Omega g(n)$, since $f(n) \geq cg(n)$ where $n \geq n_0$

Examples based on Theta Notation

1.6.6 Let the function $f(n) = 3n + 3$. So, prove that $f(n) = \theta(n)$ and identify the function type.

Proof: Let $g(n) = n$

$f(n) = \theta(g(n))$, i.e., $f(n)$ is a Big Theta of $g(n)$

Iff $f(n)$ is Big O of $g(n)$ and $f(n)$ is Big Omega of $g(n)$.

$$c'g(n) \leq f(n) \leq c''g(n) \text{ for all values of } n \geq n_0$$

$$\text{Let } n = 1, c' = 1, c'' = 100 \quad 1(1) \leq 3(1) + 3 \leq 100(1)$$

$$1 \leq 6 \leq 100, \text{ so this condition is passed for } n = 1.$$

$$\text{Let } n = 2, c' = 1, c'' = 100 \quad 1(2) \leq 3(2) + 3 \leq 100(2)$$

$$2 \leq 9 \leq 200, \text{ so this condition is passed for } n = 2.$$

As in above examples mentioned in 1.6.1 and 1.6.3, we have proved that

$f(n) = O(n)$ and $f(n) = \Omega(n)$, i.e., $3n + 3 = O(n)$ and $3n + 3 = \Omega(n)$

Hence, function $f(n) = \theta(n)$ for $n \geq 1$ which is the exact bound.

1.6.7 Let the function $f(n) = 10n^2 + 2n + 6$. So, prove that $f(n) = \theta(n^2)$ and identify the function type.

Proof: Let $g(n) = n^2$

$f(n) = \theta(g(n))$, i.e., $f(n)$ is a Big Theta of $g(n)$

Iff $f(n)$ is Big O of $g(n)$ and $f(n)$ is Big Omega of $g(n)$.

$$c'g(n) \leq f(n) \leq c''g(n) \text{ for all values of } n \geq n_0$$

$$\text{Let } n = 1, c' = 1, c'' = 100 \quad 1(1^2) \leq 10(1^2) + 2(1) + 6 \leq 100(1^2)$$

$$1 \leq 18 \leq 100, \text{ so this condition is passed for } n = 1.$$

$$\text{Let } n = 2, c' = 1, c'' = 100 \quad 1(2^2) \leq 10(2^2) + 2(2) + 6 \leq 100(2^2)$$

$4 \leq 50 \leq 400$, so this condition is passed for $n = 2$.

We have proved that,

$f(n) = O(n)$ and $f(n) = \Omega(n)$, i.e., $f(n) = 10n^2 + 2n + 6 = O(n)$ and $f(n) = 10n^2 + 2n + 6 = \Omega(n)$

Hence, function $f(n) = \Theta(n)$ for $n \geq 1$ which is the exact bound.

EXERCISE QUESTIONS

1. Let function $f(n) = 8n + 12$. So, prove that this function $f(n) = O(n)$ and also identify the function type?
2. What is an algorithm? How to analyse the algorithm?
3. Let function $f(n) = 2n^3 + 3n^2 + n + 10$. So, prove that this function $f(n) = O(n^3)$ and also identify the function type.
4. Write a note on asymptotic notations.
5. Let function $f(n) = 8n + 12$. So, prove that this function $f(n) = \Omega(n)$ and also identify the function type.
6. Explain growth rate in terms of algorithm.
7. Let function $f(n) = 9n^2 + 3n + 12$. So, prove that this function $f(n) = \Omega(n^2)$ and also identify the function type.
8. How to calculate running time of an algorithm?
9. Let function $f(n) = 7 \log n + 2$. So, prove that this function $f(n) = O(\log n)$ and also identify the function type.
10. Let function $f(n) = 8n \log n - 2$. So, prove that this function $f(n) = \Omega(n \log n)$ and also identify the function type.
11. Let $f(n) = 8n + 12$. So, prove that this function $f(n) = \Theta(n)$ and identify its function type.
12. Let $f(n) = 3n \log n + 2n$. So, prove that this function $f(n) = \Theta(n \log n)$ and identify its function type.

