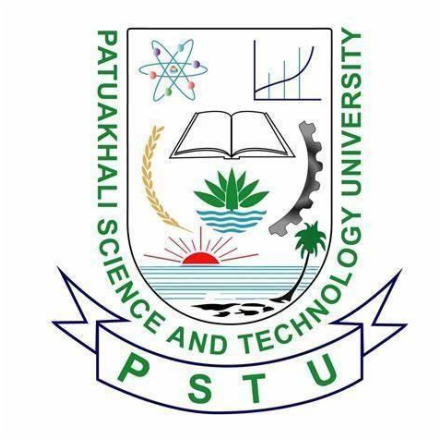


PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

COURSE CODE - CIT-121

SUBMITTED TO:

Prof. Dr. Md Abdul Masud
Department of Computer and Information Technology Faculty of Computer Science and Engineering

SUBMITTED BY:

Name: Shaid Ibna Sobhan
ID: 2102057
Registration No: 10184
Faculty of Computer Science and Engineering

Date of submission: **30th April,2024**

Algorithm

An algorithm is a set of commands that must be followed for a computer to perform calculations or other problem-solving operations.According to its formal definition, an algorithm is a finite set of instructions carried out in a specific order to perform a particular task.

- Properties of Algorithms :

Input: An algorithm has input values from a specified set.
Output: From the input values, the algorithm produces the output values from a specified set. The output values are the solution.

Correctness: An algorithm should produce the correct output values for each set of input values.

Finiteness: An algorithm should produce the output after a finite number of steps for any input.

Effectiveness: It must be possible to perform each step of the algorithm correctly and in a finite amount of time.

Generality: The algorithm should work for all problems of the desired form.

Algorithms are essential because of the large variety of applications in which they are used. Understanding how algorithms work is also crucial for developing problem-solving skills and building logical reasoning.

- Examples of common discrete mathematics algorithms include:

  Searching Algorithms to search for an item in a data set or data structure like a tree.

  Sorting Algorithms to sort items in a specific order.

  Insertion and Deletion Algorithms to insert or delete item in a data structure such as a tree or list.

  Division Algorithms such as a procedure for dividing two integers or the Euclidean Algorithm to determine the greatest common divisor between two integers.

  Optimization algorithms such as finding the line of best fit of set of points, or the problem of finding the nearest neighbor in a set of points to a given point (here close could mean most similar according to some mathematically defined measure of closeness or similarity)

Asymptotic Notations

Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

For example: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear i.e. the best case.

There are mainly three asymptotic notations:

Big-O notation

Omega notation

Theta notation

Big-O Notation (O-notation)

Big-O notation represents the upper bound of the running time of an algorithm. Thus, it gives the worst-case complexity of an algorithm.

This notation provides an upper bound on a function which ensures that the function never grows faster than the upper bound. So, it gives the least upper bound on a function so that the function never grows faster than this upper bound

For example:

If f(n) and g(n) are the two functions defined for positive integers,

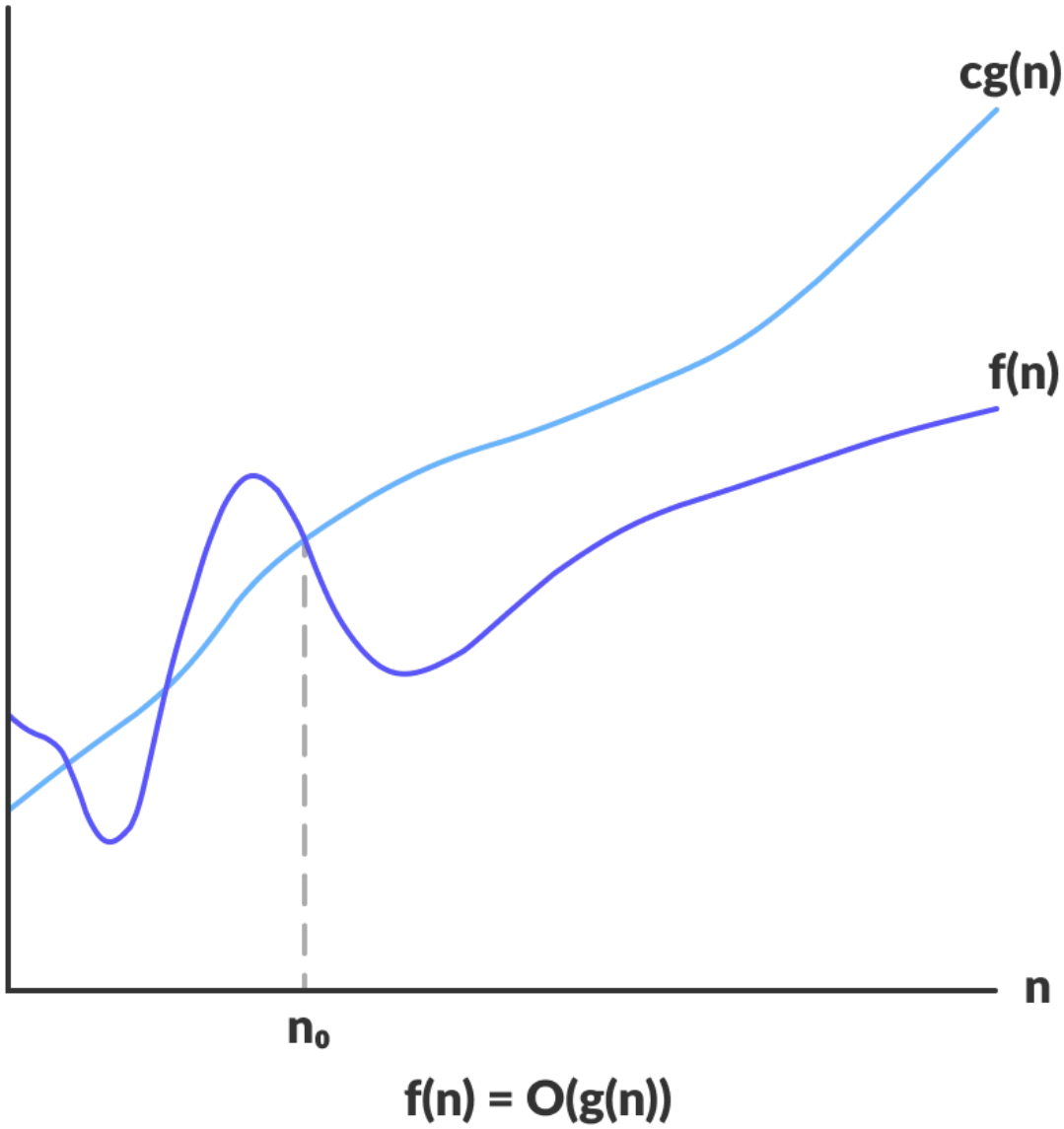
then f(n) = O(g(n)) as f(n) is big oh of g(n) or f(n) is on the order of g(n)) if there exists constants c and no such that:

$f(n) \leq c.g(n)$ for all $n \geq no$

This implies that f(n) does not grow faster than g(n), or g(n) is an upper bound on the function f(n). In this case, we are calculating the growth rate of the function which eventually calculates the worst time complexity of a function, i.e., how worst an algorithm can perform.

O(1)

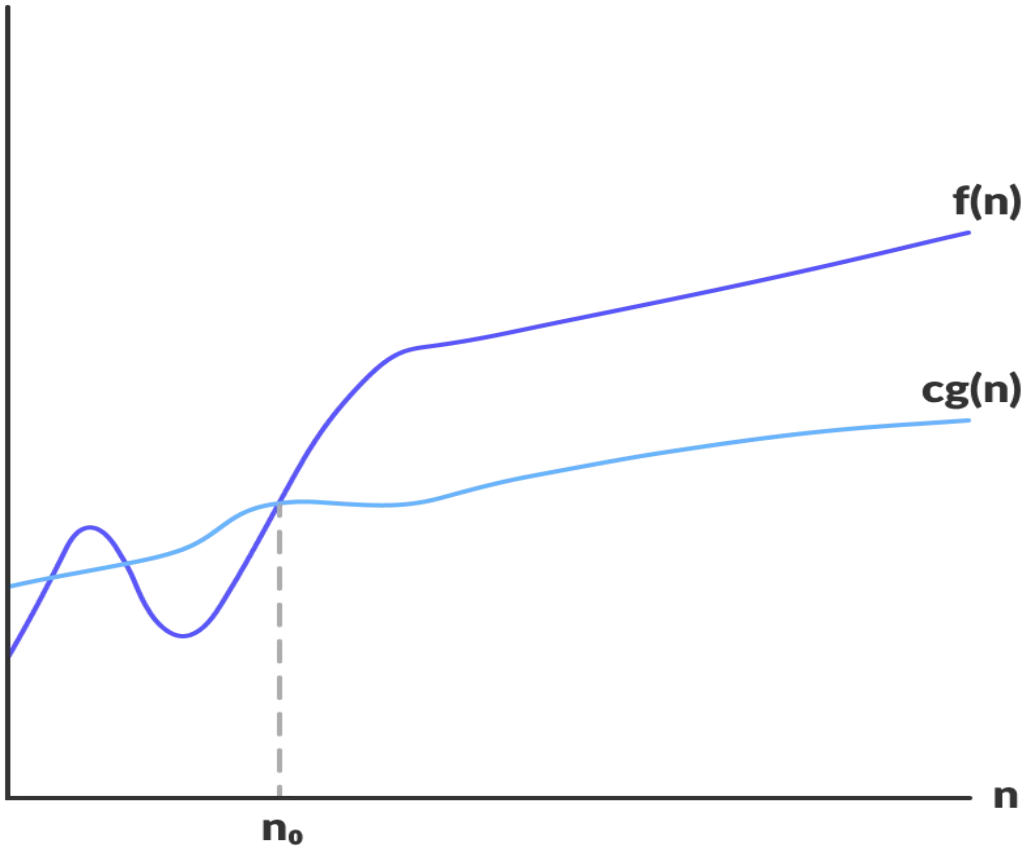
O(1) is the simplest of the O notations, and easily the best. It means that even when the action is scaled up, i.e. 2 elements vs 20,000 elements, the time spent to perform the action is constant.



Omega Notation (Ω)

Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best case complexity of an algorithm.

- It basically describes the best-case scenario which is opposite to the big o notation.
- It is the formal way to represent the lower bound of an algorithm's running time. It measures the best amount of time an algorithm can possibly take to complete or the best-case time complexity.
- It determines what is the fastest time that an algorithm can run.



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

The above expression can be described as a function f(n) belongs to the set Ω(g(n)) if there exists a positive constant c such that it lies above cg(n), for sufficiently large n.

For any value of n, the minimum time required by the algorithm is given by Omega Ω(g(n)).

Theta Notation (θ)

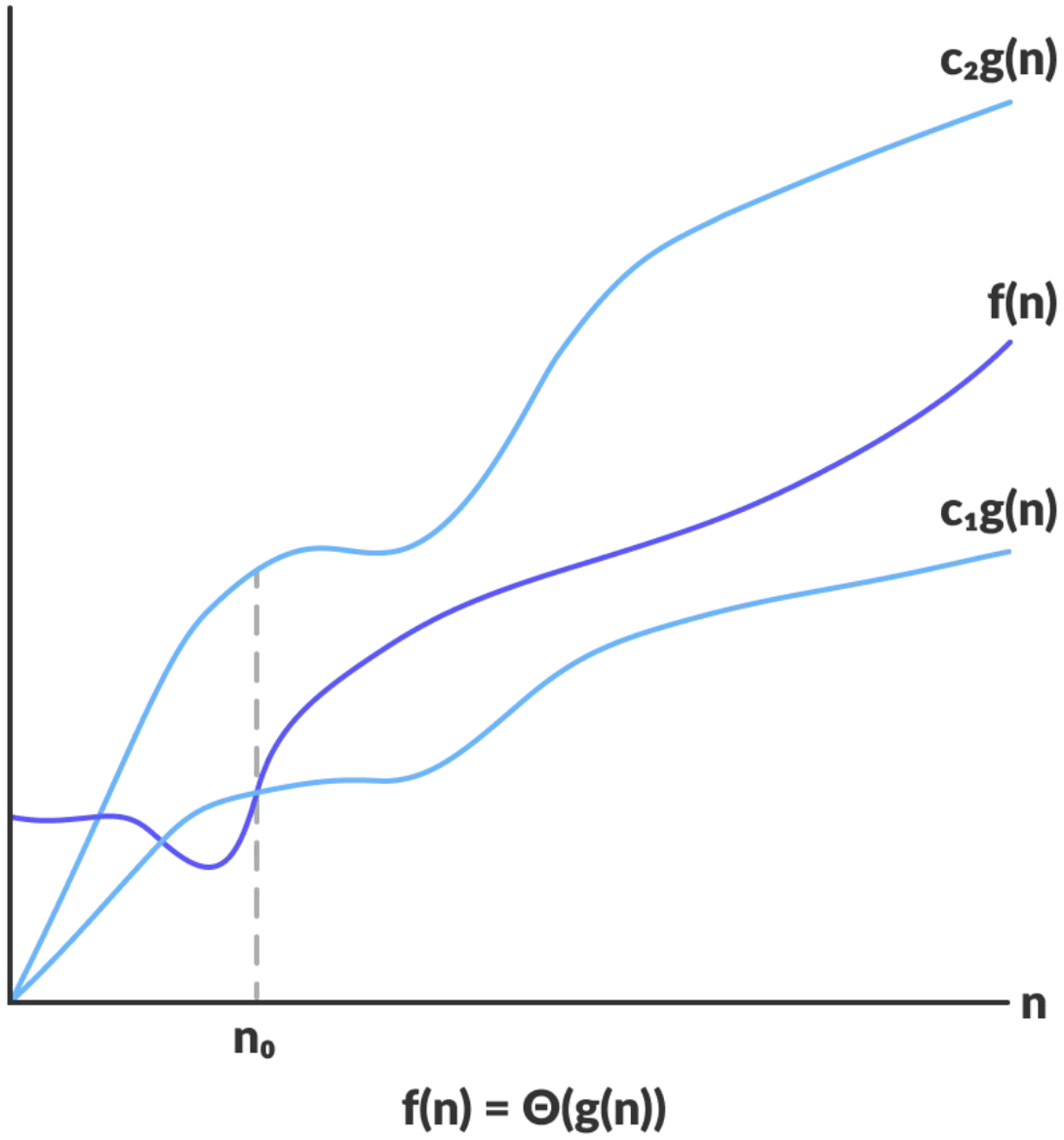
Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average-case complexity of an algorithm.

- The theta notation mainly describes the average case scenarios.
- It represents the realistic time complexity of an algorithm. Every time, an algorithm does not perform worst or best, in real-world problems, algorithms mainly fluctuate between the worst-case and best-case, and this gives us the average case of the algorithm.
- Big theta is mainly used when the value of worst-case and the best-case is same.
- It is the formal way to express both the upper bound and lower bound of an algorithm running time.

Definition: Let g and f be the function from the set of natural numbers to itself. The function f is said to be Θ(g), if there are constants c1, c2 > 0 and a natural number n0 such that c1* g(n) ≤ f(n) ≤ c2 * g(n) for all n ≥ n0.

Mathematical Representation:

Θ (g(n)) = {f(n): there exist positive constants c1, c2 and n0 such that 0 ≤ c1 * g(n) ≤ f(n) ≤ c2 * g(n) for all n ≥ n0}
Note: Θ(g) is a set



The above expression can be described as a function f(n) belongs to the set Θ(g(n)) if there exist positive constants c1 and c2 such that it can be sandwiched between c1g(n) and c2g(n), for sufficiently large n.