Prepared By Abhiyan Dev Rimal
https://github.com/abhirimal/be-software-notes

classmate
Date
Page

Chapter-1 (1 or 2 Q)

1) Define algorithms. What are the criteria's than an algorithm must satisfy?

→ Algorithm is defined as the finite set of steps to solve a problem.

→ Algorithm is the sequence of instruction designed in such a way that if the instructions are executed in the specific sequence, the desired result is obtained.

The criteria's than an algorithm must satisfy are:-
(features/characteristics/Properties of algorithm)

1) Finiteness/Termination: An algorithm must terminate after a finite number of steps.

2) Definiteness: Each step of the algorithm must be precisely defined.

3) Input: An algorithm can have zero or more inputs. Input can be given initially or as program runs.

4) Output: An algorithm can give one or more result. After an algorithm terminates, algorithm must give desired result.

5) Effectiveness: An algorithm should be more effective among different ways of problem solving.

Scanned with CamScanner

11 2) What is algorithm analysis? Why is it necessary to analyze an algorithm?

or

Why is performance analysis of an algorithm important.

or

Ans: Algorithm analysis is an important part of of computational complexity theory, which provides theoritical estimation for the required resources of an algorithm to solve a specific computational problem.

→ Analysis of the algorithm is the process of analyzing the problem-solving capability of the algorithm in terms of the time and ~~space~~ space required (the size of memory for storage while implementation). However, the main concern of the analysis of the algorithm is the required time or performance.

→ To go from city "A" to city "B", there can be many ways of accomplishing this: by flight, by bus, by train and also by bicycle. Depending on the availability and convenience, we choose the one that suits us. Similarly, in computer science, multiple algorithms are available for solving the same problem (for example, a sorting problem has many algorithms, like insertion sort, selection sort, quick sort and many more).

Scanned with CamScanner

→ Algorithm analysis helps us to determine which algorithm is most efficient in terms of time and space consumed.

→ Generally, we perform the following types of analysis.

1) Worst-case → the maximum number of steps taken on any instance of size n.
2) Best-case → the minimum number of steps taken on any instance of size n.
3) Average-case → an average number of steps taken on any instance of size n.

→ Performance analysis of an algorithm depends upon two factors i.e amount of memory used and amount of compute time consumed on any CPU. Formally they are notified as complexities in terms of:

1) Space Complexity
2) Time Complexity

Scanned with CamScanner

(a) What are the measures that we take to measure the efficiency of an algorithm → Time & Space Com.

or

1111 3) What do you understand by complexity of an algorithm.

or

Explain space & time complexity of an algorithm using an example.

or.

Significance of space and time complexity.

→ The complexity of an algorithm computes the amount of time and spaces required by an algorithm for an input of size (n).

→ The complexity of an algorithm can be divided into two types. i.e Time Complexity and Space Complexity

→ Time is important because we need our programs to run as fast as possible to deliver the results quickly.

→ Space is important because machines have only a limited amount of space to spare for programs.

→ The best algorithm is the one that completes its execution in the least amount of time using the least amount of space.

→ But often, in reality, algorithms have to tradeoff between saving space or time.

→ If we need our algorithm to run as fast as possible despite the memory usage, we can pick the most time-efficient algorithm as the best algorithm and vice-versa.

Scanned with CamScanner

1) Time Complexity

→ Time complexity of an algorithm is the amount of computer time it needs to run to completion.

Defn → The time complexity is defined as the process of determining a formula for total time required towards the execution of that algorithm.

→ This calculation is totally independent of implementation and programming language.

2) Space complexity

→ Space complexity of an algorithm is the amount of memory it needs to run to completion i.e from start of execution to its termination.

Defn → Space complexity is defined as the process of determining a formula for prediction of how much memory space is required for the successful execution of the algorithm.

→ The memory space is generally considered as the primary memory.

⇒ Time and space complexity depends on lots of things like hardware, operating system, processors, etc. However we don't consider any of these factors while analyzing the algorithm. We will only consider the execution time of an algorithm.

Scanned with CamScanner

Date
Page

⇒ Which is more important: space or time complexity

→ Both matters, but sometimes ~~time~~ space complexity matters more. Your space is fixed for any set of hardware. If you don't have enough, you just can't run the algorithm.

→ On the other hand, programming contests are about time complexity. Space/memory can be reused easily. Available space can be expanded easily. Whereas, time needed for computing cannot be shorten that easily.

~~⇒ What are the measures that we take to measure the efficiency of an algorithm?~~

⇒ What do you mean by Best case, Worst case and average case complexity of an algorithm.

→ To analyze the given algorithm, we need to know with which inputs the algorithm takes less time (performing well) and with which inputs the algorithm takes a long time.

→ An algorithm can be represented in the form of an expression. That means we represent the algorithm with multiple expressions: one for the ~~time~~ case where it takes less time, ~~and~~ one for the case where it takes more time and another for the case where it takes average time.

Scanned with CamScanner

→ There are three types of analysis

1) Worst case

→ maximum time required for program execution.

→ worst case scenario occurs when the combination of inputs ~~that~~ that takes the maximum amount of time for completion is passed to the algorithm.

→ input is the one for which the algorithm was the slowest.

→ The execution time of the worst case acts as an upper bound to the time complexity of the algorithm.

→ Eg: In linear search, the worst case occurs when k is not present in the list. This takes the algorithm n+1 iterations to figure out that the number is not in the list.

nums = [1,2,3,4,5,6]

n = 6

k = 7

2) Best case

→ minimum time required for program execution.

→ we consider the combination of inputs that allows the algorithm to complete its execution in the minimum amount of time as the best-case scenario in terms of time complexity.

→ input is the one for which the algorithm ~~is the~~ was the fastest.

→ The execution time in this case acts as a lower bound to the time complexity of the algorithm.

Scanned with CamScanner

→ Eg: In linear search, the best-case scenario occurs when k is stored at the 0th index of the list. In this case, the algorithm can complete execution after only one ~~B~~ iteration of the for loop.

nums = [1, 2, 3, 4, 5, 6]

n = 6

k = 1

3) Average Case

→ Average time required for program execution.

→ To find the average case, we get the sum of running times of the algorithm for every possible input combination and take their average.

→ Assumes that the input is random.

~~→ In linear search, the number of iterations the~~

~~at~~

Asymptotic notation can be defined as the mathematical way of representing the time complexity.

Scanned with CamScanner

→ it is a method of describing the limiting behaviour of a function.

5) What is asymptotic notation.

→ 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.

→ Using asymptotic analysis we can very well conclude the best case, average case, and worst case scenario of an algorithm.

→ Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis.

→ For eg: the running time of one operation is computed as $f(n)$ and may be for another operation it is computed as $g(n^2)$. This means the first operation running time will increase linearly with the increase in n and the running time of the second operation will increase exponentially when n increases. Similarly, the running time of both operations will be nearly the same if n is significantly small.

→ The following three asymptotic notations are mostly used to represent the time complexity of algorithms.

i) Big-O notation (Big-O)

ii) Big Omega notation (Big-Ω)

iii) Big-Theta notation (Big-Θ)

Scanned with CamScanner

V.Imp

IIIII 6) Explain Big O, Big Θ & Big Ω notation for performance analysis of an algorithm.

or,

Explain the asymptotic notations Big O, Big Θ & Big Ω

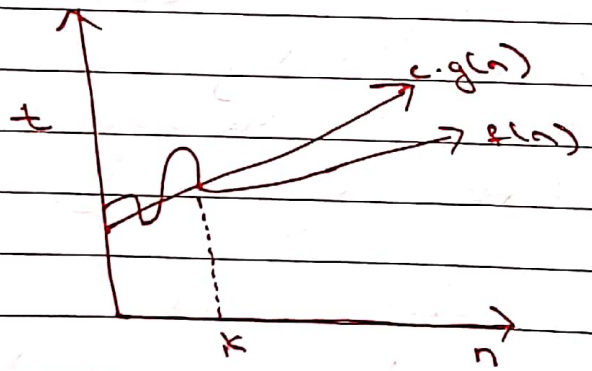
→ The asymptotic notations are explained below:-

→ Big-O Notation

→ It describes the worst-case running time of an algorithm.

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

→ We typically consult the Big-O because we must always plan for the worst case.



Here, $f(n) = O(g(n))$ ∴ $c > 0$, $n \geq k$, $k \geq 0$

then ∴ $f(n) \leq c \cdot g(n)$

→ A function $f(n) = O(g(n))$ (read as f(n) is bigoh of g(n)) if there exists two positive constants c and k such that for all $n \geq k$.

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

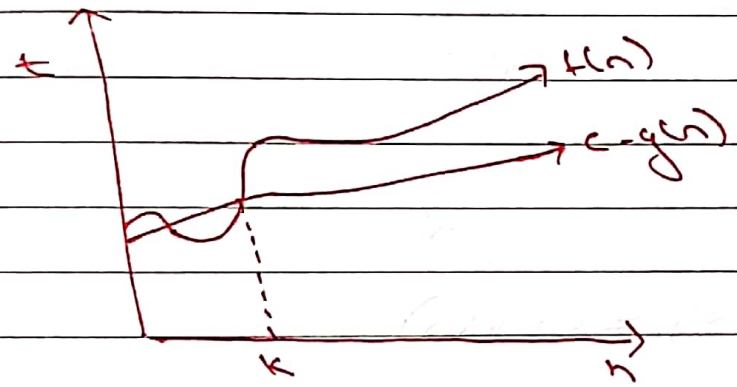
Scanned with CamScanner

→ The above relation says that $g(n)$ is an upper bound of $f(n)$.

2) Big Ω notation (Omega)

→ It describes the best running time of an algorithm.

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



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

then $\therefore f(n) \geq c.g(n)$

→ A function $f(n) = \Omega g(n)$ {read as $f(n)$ is big Ω of $g(n)$} if there exists two positive constants c and k such that for all $n \geq k$

$$\therefore c.g(n) \leq f(n)$$

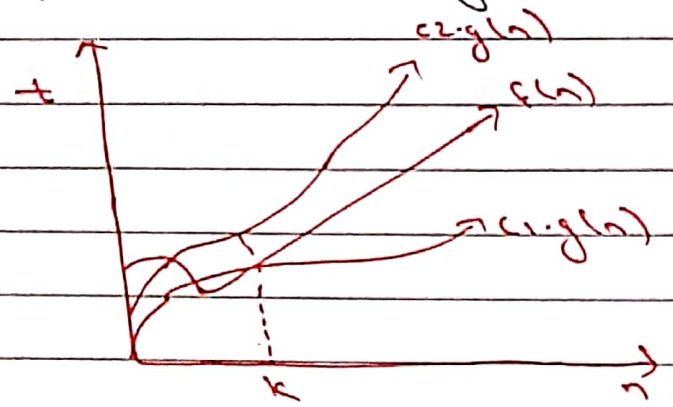
→ The above relation says that $g(n)$ is lower bound of $f(n)$.

P.T.O →

Scanned with CamScanner

3. Big theta notation (θ)

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



$$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$$

→ A function $f(n) = \theta g(n)$ {read as f(n) is big theta of g(n)} if there exists 3 positive constants c_1, c_2 & k such that for all $n \ge k$

$$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n).$$

→ The above relation says that f(n) is order of g(n)

Scanned with CamScanner

Common asymptotic Notations

constant	$O(1)$
logarithmic	$O(\log n)$
linear	$O(n)$
$n \log n$	$O(n \log n)$
quadratic	$O(n^2)$
cubic	$O(n^3)$
polynomial	$n^{O(1)}$
exponential	$2^{O(n)}$

Examples of asymptotic notation:

1) Given, $f(n) = 3n^2 + 4n + 7$ and $g(n) = n^2$, then prove that $f(n) = O(g(n))$.

Proof: let us choose c and k values as 14 and 1, respectively, then we can have

$$f(n) \leq c \cdot g(n) \quad , n > k$$

or, $3n^2 + 4n + 7 \leq 14 \cdot n^2$ for all $n >= 1$

(true)

$\therefore$ Hence, the above inequality is true. Hence $f(n) = O(g(n))$.

Scanned with CamScanner

2) $f(n) = 3n^2+4n+7$ and $g(n) = n^2$, then prove that $f(n) = \Omega(g(n))$.

Proof: let us choose c and k values as 1 and 1, respectively then we can have.

$$\text{or, } f(n) \geq c \cdot g(n) \quad , n \geq k \text{ as}$$

$$\therefore 3n^2+4n+7 \geq n^2 \quad \text{for all } n >= 1$$

$\therefore$ The above inequality is trivially true. Hence, $f(n) = \Omega(g(n))$.

3) Example: $f(n) = 3n^2+4n+7$ and $g(n) = n^2$, then prove that $f(n) = \Theta(g(n))$

Proof: let us choose c_1, c_2 and k values as 1, 14 and 1 respectively then we can ~~write~~ have,

~~or f(n) <= c1 g(n)~~

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

$$1 \cdot n^2 \leq 3n^2+4n+7 \leq 14 \cdot n^2 \quad \text{for } n >= 1$$

So, the above inequality is trivially true. Hence, $f(n) = \Theta(g(n))$.

Scanned with CamScanner

Randomized Algorithms

= An algorithm that uses random numbers to decide what to do next anywhere in its logic is called randomized algorithm.

for eg: In randomized quick sort, we use random numbers to pick the next pivot. Typically, this randomness is used to reduce time complexity or space complexity in other standard algorithms.

Two main type of randomized algorithms:

i) Las Vegas Algorithm
ii) Monte-Carlo algorithms

Las Vegas Algorithm: Eg: Randomized Quick sort

→ Algorithm may use randomness to speed up the computation, but algorithm must always return correct answer to input.

Monte Carlo Algorithm Eg: finding value of π.

→ They are allowed to give wrong return values.

However, returning a wrong return value must have a small probability, otherwise that monte-carlo algorithm would not be of any use.

Scanned with CamScanner

Deterministic	Non-deterministic
1) For a particular input the computer will give same output.	1) For particular input the computer will give different output on different execution.
2) Can solve the problem in polynomial time.	2) Can't solve the problem in polynomial time.
3) Can determine the next step of execution.	3) Cannot determine next step as algorithm can take more than one path.

Scanned with CamScanner