Date:
P. No:

Assignment - 1

Q-1 What is ADA ? what is the need to study algorithms ? explain in detail ?

Ans: ADA stands for Analysis & design of Algorithms.

The Analysis is a process of estimating the efficiency of an algorithms. There are two fundamental parameters. ~~There~~ based on which we can analysis the algorithm.

Space Complexity :- it can be understand as the amount of space required by an algorithm to run to completion.

Time Complexity :- it is a function of I/P size n that refers to the amount of time needed by an algorithm to run the completion.

Generally we make three types of analys which is as follows:

- Worst - Case time Complexity
- Average - Case time Complexity
- Best - Case time Complexity.

Need of Algorithm:-

i) To understand the basic idea of the problem.

ii) To find an approach to Solve the problem.

iii) To improve the efficiency of existing techniques.

iv) To understand the basic principles of designing the algorithms

v) To understand the flow of problem

vi) To measure the behavior of the method in all cases.

vii) To understand the principle of designing.

Date:
P. No:

Q:) write all the three Cases of master theoream for the equation:-

○ $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

Ans:- $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$f(n) = (n^k \log^e n)$

$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^k \log^e n)$

$a \geq 1, b \geq 1, k \geq 0$

p is a real number

Case 1:- If $a > b^k$ then $T(n) = \Theta(n \log_b a)$

Case 2:- If $a = b^k$

a) if $p > -1$ than $T(n) = \Theta(n \log b^a \log^{p+1} n)$

b) if $p = -1$ than $T(n) = \Theta(n \log^a \log \log n)$

c) if $p < -1$ than $T(n) = \Theta(n \log b^a)$

Case 3: if $a < b^k$

a) if $p \geq a$ then $T(n) = \Theta(n^k \log^p n)$
b) if $p < a$ then $T(n) = \Theta(n^k)$.

Q3) What is an asymptatic notations? Give the different natations to used to represent the complexity of algorithms?

Ans:- Asymptatic notations are the mathematical natations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

There are mainly three asymptatic natations.

1) Big-oh (O) Natation

Date:
P. No:

ii) Omega (Ω) Notation

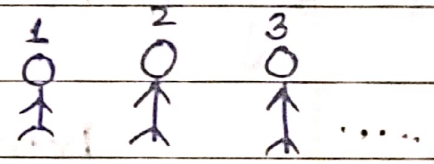
iii) Theta (Θ) Notation

1) Big-oh (O) Notation :-

It represents the upper bound of the running time of an algorithm, Therefore, it give the worst case Complexity of an algorithm

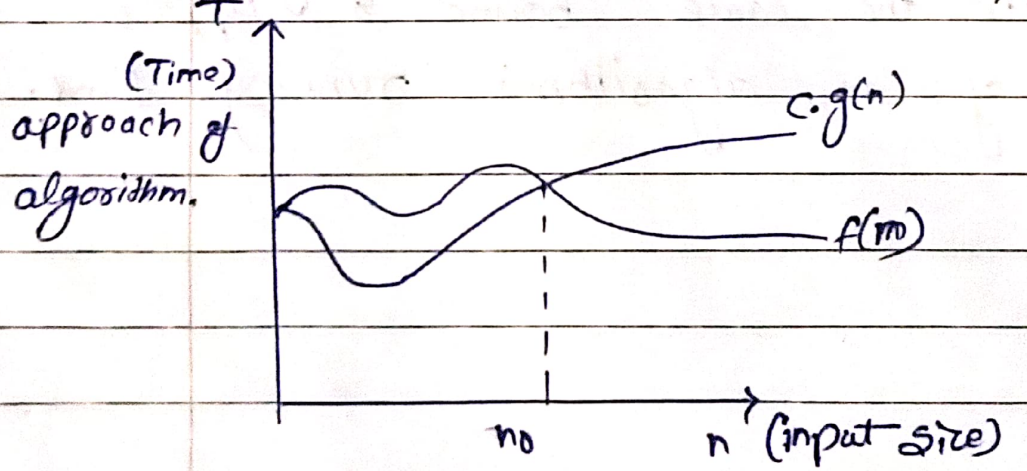
- it is most widly used notation for Asymaptatic analysis.
- it specifies the upper bound of the function.

The ~~maximum time~~



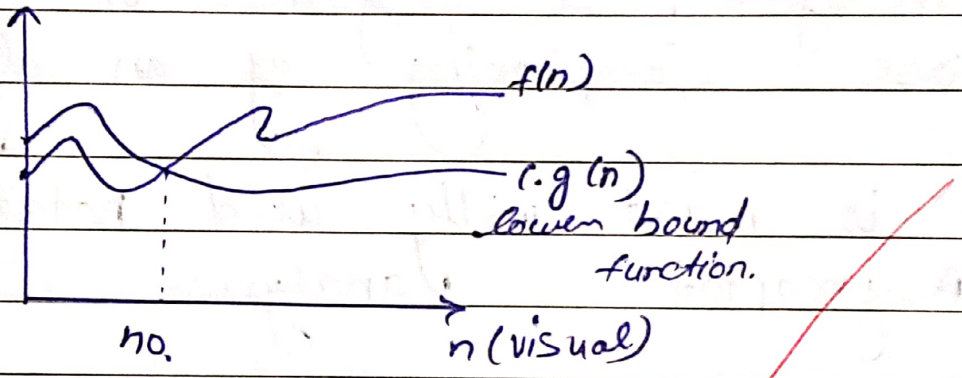
$O(n)$

- upper $C.g(n)$
- worst case



② omega Notation (Ω Big omega) =

The Ω(n) is the formal way to express the lower bound of an algorithms running time. It measures the best case time Complexity or best possible amount of time an algorithm can take to complete.

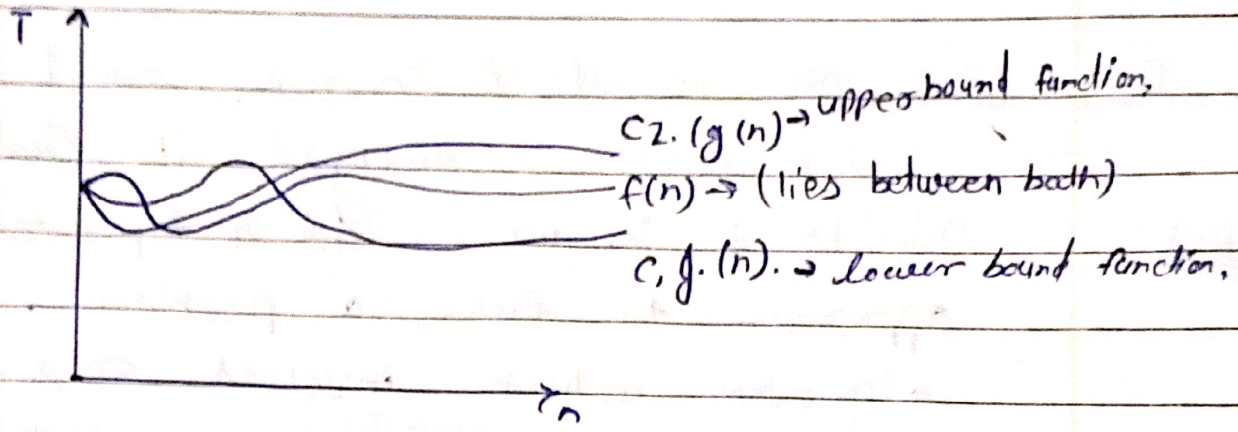


$f(n) \geq c \cdot g(n)$

$n > n_0 \quad c > 0, \; n_0 \geq 1. \Rightarrow f(n) = \Omega(g(n))$

③ Big Theta (Θ) Notation = The Θ(n) is the formal way to express both the lower bound & upper bound of an algorithm's running time.

Date:
P. No:



$c_2 \cdot g(n) \geq f(n)$ $c_1 \cdot g(n) \Rightarrow f(n) = O(g(n))$

$n \geq n_0$

$c_1, c_2 \geq 0$ $n_0 \geq 1$.

This graph represents the time growth of an algorithm with respect to input size (n)

- O → upper bound → worst case
- Ω → lower bound → Best - Case.
- Θ → Between (Average) → Average case

Complexity =

O(1) : constant O(n) : linear

$O(n^2)$: quadratic $O(n^3)$: cubic

$O(2^n)$: exponential O(logn).

Q4) Give the divide & conquer solution for quick sort & analyse its complexity?

Ans:- Divide & Conquer is a top-down approach to solve a problem. The algorithm which involves D&C technique involves 3 steps =>
Divide = The original problem into a set of sub problems
→ Conquer (or solve) every sub-problem individually recursively
→ Combine = The solution of these sub problems to get the solutions of original problem.

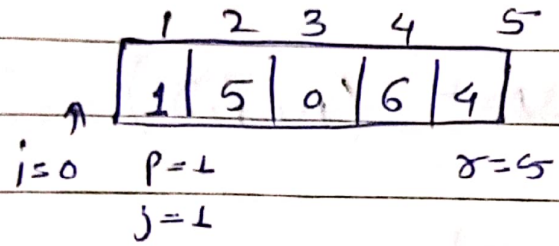
Quick Sort =
- pivot element based
- partition based.

It picks an elements as pivot & partitianal the given array around the picked pivot. There are many different varsions of quicks sort that picks the pivot in different ways.

Date:
P. No:

1) Always pick first element as pivat.
2) Always pick last element as pivat.
3) pick a random element as pivat.
4) pick medium as pivat.

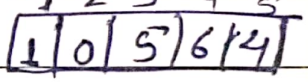
ex:-



x is the pivat
A[r] = x
So x=4

// partition function.
Partition (A, p^1, r^5)

{
x = A[r]
i = p-1
for (j = p^1 to r^4 - 1)



{
if (A[j] ≤ x) {
i = i+1
exchange.
A[i] & A[j] }}

find the index of i & j.

exchange A[i-1] & A[r]
return i+1 }
// Algorithm

quicksort (A, p, r)

{ if (p < r)
q = partition (A, p, r)

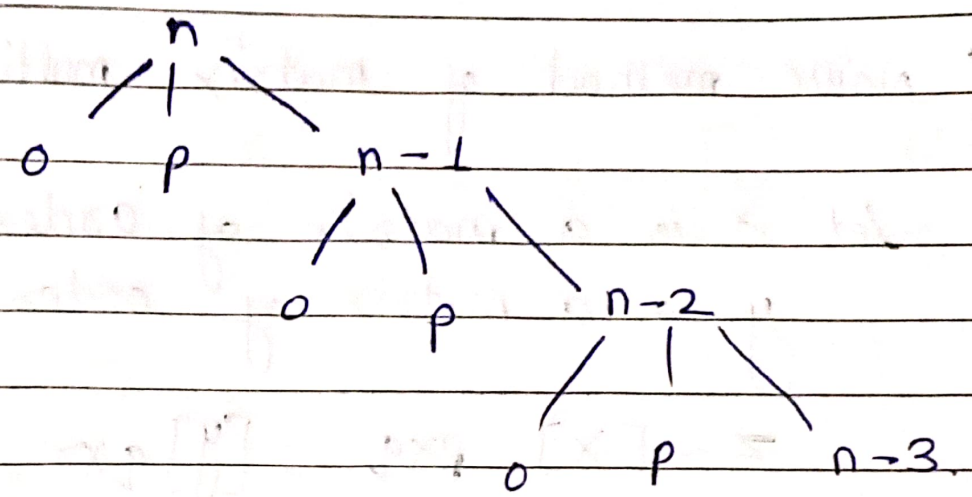
quick sort (A, p, q-1)
quick Sort (A, q+1, r)

}}

Complexity -

Big(Ω) Best case : O(n log n)
Big(Θ) Average case : O(n log n)
Big(O) worst case : O(n²)

Date:
P. No:



$$T(n) = T(n-1) + n$$

$$= O(n^2)$$ Ans

Q5 How can we prove that Strassen's matrix multiplication is advanges over ordinary matrix multiplication?

Ans

It is used to solve the matrix multiplication. each row with each column. to achive the product matrix time Complexity taken by this approach is $O(n^3)$ to $O(n \log -1)$.

• Naive method of matrix multiplication.

Let X is a matrix of order $p \times q$

y is a matrix of order $q \times r$.

$z = [X]_{p \times q} \quad [y]_{q \times r}$

z has the order $p \times r$.

Algorithm matrix - multiplication.
(x, y, z)

for i = 1 to p for j = 1 to r

z[i, j] = 0 for k = 1 to q do

z[i, j] = z[i, j] + x[i, k] x y [k, j]

Complexity = $O(n^3)$

Smm algorithm = with this algorithm the time consorption can be improved a little bit

Date:
P. No:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \qquad y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$Z = \begin{bmatrix} I & J \\ K & L \end{bmatrix}$$

$m_1 = (A+C) \times (E+F)$
$m_2 = (B+D) \times (G+H)$
$m_3 = (A-D) \times (E+H)$

$m_4 = [A \times (E-H)]$
$m_5 = (C+D) \times E$
$m_6 = (A+B) \times H$
$m_7 = D \times (G-E)$

Then,
$I = m_2 + m_3 - m_6 - m_7$
$J = m_4 + m_6$
$K = m_5 + m_7$
$L = m_1 - m_3 - m_4 - m_5$

22/3/24

Analysis

$T(n) = ?$

Date:
P. No:

Analysis =

$T(n) = ?$ c

$$\begin{cases} T \times T\left(\frac{n}{2}\right) \times d \sin n^2 & \text{if } n=1 \\ \text{otherwise} \end{cases}$$

where c and d are Constans

we get $T(n) = O(n \log)$.