11/3/24. Assignment - 1 Date: / / Page no:

1. What is ADA? What is the need to study Algorithm. Explain in detail?

ADA:- ADA is a branch of CS that focus on the study of algorithms, their design, analysis of their of efficiency and their application in solving computational problems.
Computational means which can be solved by an algorithm.

Need to study algorithm:-

- problem solving:- algorithm provide systematic approaches to solving various computational problems by studying algorithms, you learn how to crack complex problem.
- Efficiency:- Understanding algorithms helps in developing efficient solutions.
- Optimization:- Algorithm often provide multiple solutions to the same

Date: / / Page no:

problem, with varying trade off in terms of time space.

Foundation for computer science:- Algorithms forms the backbone of computer science. They are fundamental to various subfields such that a data structures.

Problem Domain independence algorithms Algorithms are not specific to any particular programming language or domain.

Competitive programming and interviews In fields like software engineering algorithms knowledge is often a key component of technical interviews.

Q.2 Give the divide and conquer solution for quicksort and analyse its complexity.

Divide and Conquer Technique.

Date: / / Page no:

DSC is a top down approach to solve a problem the algorithm which involves DSC techniques involves 3 steps.

Divide:- The original problems into a set of sub problems.

Conquer (or solve) :- every sub-problem individually recursive.

Combine:- The solutions of these sub-problems to get the solutions of original problem.

* Quicksort (For sorting)

- pivot element based
- Partition based.

It picks an element as pivot and partition the given array around the picked pivot. There are many different versions of quicksort that picks the pivot in different ways.

Date: / / Page no:

1) Always pick first elements as pivot
2) Pick last elements as Pivot.
3) Pick a random element as Pivot
4) Pick median as Pivot.

Ex:-

1	2	3	4	5
1	5	0	6	4

i=0 p=1 j=1

x is the point
A[r] = x
So, x = 4

→ // Partition Function:-

```
Partition (A, p, r)        // p=1, r=5
{
   x = A[r]
   i = p-1
for(j = p to r-1)          // j=1
if (A[j] ≤ x)
}
   i = i+1
   exchange
   A[i] & A[j]
}
```

Date: / / Page no:

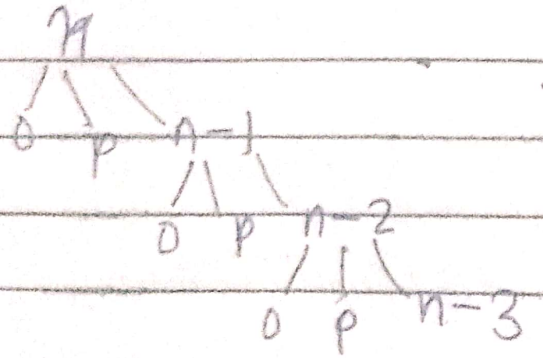
} exchange
A[i+1] & A[r]
return (i+1)
}

// Algorithm :-

quicksort (A, p, r)
{
if (p < r)
q = partion(A, p, r)
quicksort (A, p, q-1)
quicksort (A, q+1, r)
}

// complexity :-

- Best case :- O(n log n)
- Average case :- O(n log n)
- Worst Case :- $O(n^2)$ - { when the array is already sorted }



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

$O(n^2)$

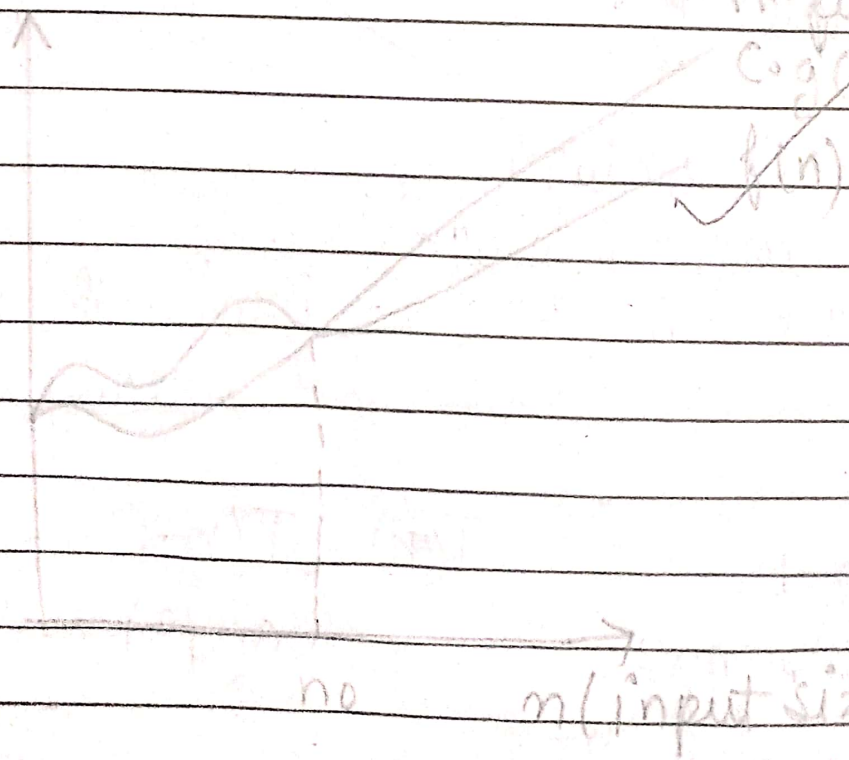
Date: / / Page no:

Q.3 Define asymptotic notations give different notations used to represent the complexity of algorithm.

Definition:- Asymptotic notations are mathematical representation used in computer science to describe the running time or space complexity of an algorithm as the input size approaches infinity.

Different types notations

(a) Big O Notation:-



Date: / / Page no:

$f(n) \geq c \cdot g(n)$, $n \geq n_0$, $c > 0$, $n_0 \geq 1$ $\Longrightarrow$ $f(n) = O(g(n))$

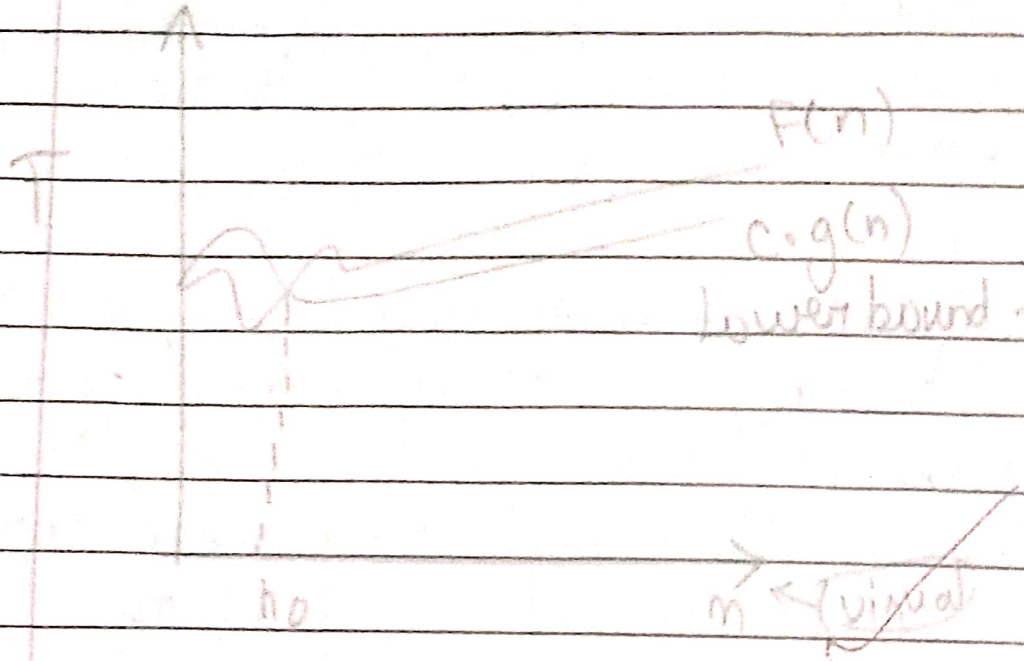
- $O(n)$ is the formal way to express the upper bound of an algorithm for running time.
- It measures the worst case time complexity or longest amount of time an algorithm can possibly take to complete.

(b) Omega Notation Ω (Big omega)

The $\Omega(n)$ 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.

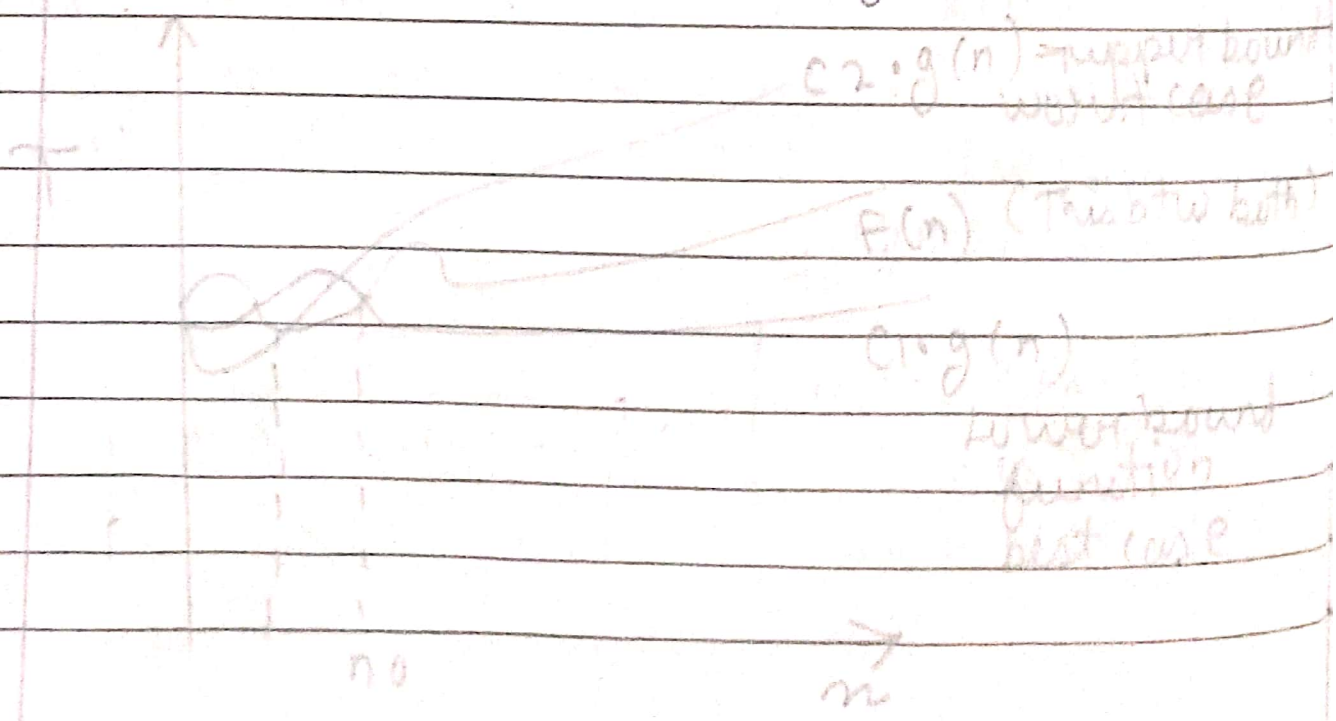
$f(n) \geq c \cdot g(n)$, $n \geq n_0$, $c > 0$, $n_0 \geq 1$ $\Longrightarrow$ $f(n) = \Omega(g(n))$

Date __/__/__ Page no: _____



(c) Big theta (Θ) Notation.

The Θ(n) is the formal way to express both the lower bound and upper bound of an algorithm's running time.



Date: / / Page no:

This graph represents the time growth of an algorithm w.r.t input size 'n'.

$c_2 \cdot g(n) \geq f(n) \geq c_1 \cdot g(n)$
$n \geq n_0$
$c_1, c_2 > 0 \cdot n_0 \geq 1$
$\Rightarrow f(n) = \Theta(g(n))$

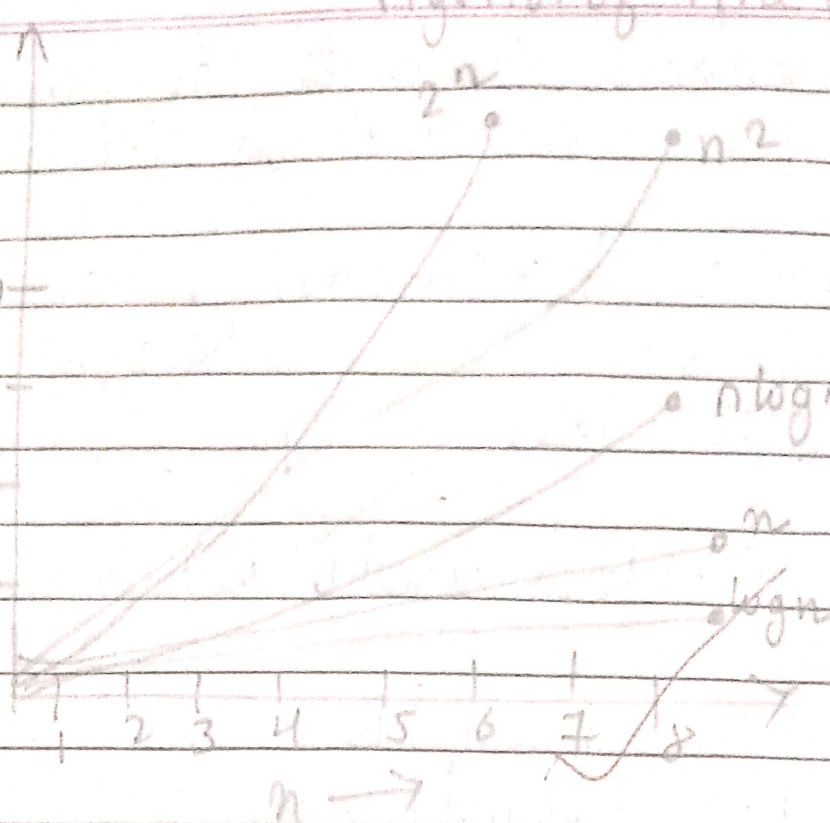
$\Theta =$ is used in average cases.

Ω	Lower bound	Best-case
O	upper bound	worst-case
Θ	Between	Averg-case

Complexity :-

$O(1)$: constant
$O(n)$: Linear
$O(n^2)$: quadratic
$O(n^3)$: cubic
$O(2^n)$: exponential.
$O(\log n)$ Logarithmic

Fig. Plot of five functions



Q.4. Write the three cases of master theorem for the equation:-

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

Master theorem:- For solving recurrence relation.

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

Check $a \geq 1$, $b > 1$, $k \geq 0$
p is a real number

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

Date: / / Page no:

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

a) If $p > -1$ then $T(n) = \Theta(n \log_b^a \log^{p+1} n)$
b) If $p = -1$ then $T(n) = \Theta(n \log_b^a \log \log n)$
c) If $p < -1$ then $T(n) = \Theta(n \log_b^a)$

Case-3 If $a < b^k$

a) If $p \geq 0$ then $T(n) = \Theta(n^k \log^p n)$
b) If $p < 0$ then $T(n) = O(n^k)$

5 How can we prove that strassen's matrise multiplication is advantages over ordinary matrise multiplication.

- It is used to solve the matrise multiplication problem.
- The usual matrise multiplication method multiplies each row with each column to achieve the product matrise. Time complexity taken by this approach is $O(n^3)$, since it takes three loops to multiply.

This method was introduced to reduce the time complexity from
(a) n^3 to $O(n\log^{-1})$.

Naive Method of matrix multiplication :-

Let X is a matrix of order p x q & y is a matrix of order q x r

Z = [X] p x q [Y] q x r

Z has the order p x 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 algo, the time consumption can be improved

a little bit.

It can be applied only on square matrices where 'n' is a power of 2 order of both the matrices are n×n.

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

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

$M_1 := (A+C) \times (E+F)$
$M_2 := (B+D) \times (G+H)$
$M_3 := (A-D) \times (E+H)$
$M_4 := A \times (F-H)$
$M_5 := (C+D) \times E$
$M_6 := (A+B) \times H$
$M_7 := D \times (G-E)$

Then

$I := M_2 + M_3 - M_6 - M_7$

$J := M_4 + M_6$ & $K := M_5 + M_7$

$L := M_1 - M_3 - M_4 - M_5$

Date: / / Page no:

Analysis :-

$$T(n) = \begin{cases} c & : \text{if } n = 1 \\ 7 \times T\left(\frac{n}{2}\right) + dxn^2 & : \text{otherwise} \end{cases}$$

where c and d are constants

we get $T(n) = O(n^{\log 7})$.

Analysis 11/3/24