MADHVI KELOTRA

Analysis Design of Algorithm
CS-402

En. Roll. :- 0829CS221101
Class - CS-B
Sem - IV

Date: / / Page no:

Assignment - 1

What is ADA? What is the need to Study Algorithms? Explain in detail.

ADA :-

ADA is a branch of CS that focuses on the study of algorithms, their design, analysis of their efficiency and their application in Solving computational problems.

Computational means which can be Solved by an algorithm.

The need to Study algorithm are :-

- **Problem Solving :-** Algorithms provide Systematic approaches to Solving Various computational problems.

Date: / / Page no:

By studing algorithms, you learn how to break down complex problems into smaller, more manageable steps, leading to efficient and effective solutions.

- Efficiency :- Understanding algorithms helps in developing efficient solutions.

- Scalability :- As the size of data and computational tasks continues to grow, the importance of scalable algorithms becomes paramount.

- Optimization :- Algorithm often provide multiple solutions to the same problem, with varying trade-offs in terms of time and space complexity.

Date: / / Page no:

- **Foundation for Computer Science:**

Algorithms form the backbone of computer science they are fundamental to various subfields such as data structures, artifical intelligence.

- **Problem Domain independence:**

Algorithms are not specific to any particular programming language or domain.

- **Competitive programming and Interviews:**

In field like Software Engineering, algorithmic knowledge is often a key component of technical interviews.

Date: / / Page no:

Q2 Give the Divide and conquer Solution for quicksort and analyze its complexity.

Ans

Divide And conquer Technique (D&C)

D & C is a top-down approach to solve a problem. The algorithm which involves D & C technique involves 3 step →

- Divide :- The original problem into a set of sub problems.
- Conquer (or Solve) every sub-problem individually, recursive.
- combine the solution of these sub problems to get the solution of original problem.

Date: / / Page no:

Quick Sort.

- Pivot element based
- Partition based

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

Always pick first element as pivot.
Always pick last element as pivot.
Pick a random element as pivot.
Pick median as pivot.

Example :-

1	2	3	4	5
1	5	0	6	(4)

i = 0, j = 1, p = 1, r = 5

x is the pivot

A[r] = x

So, $\boxed{x = 4}$

Date: / / Page no:

// Partition function

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

1	2	3	4	5
1	0	5	6	4

find the index of i & j

→ // Algorithm

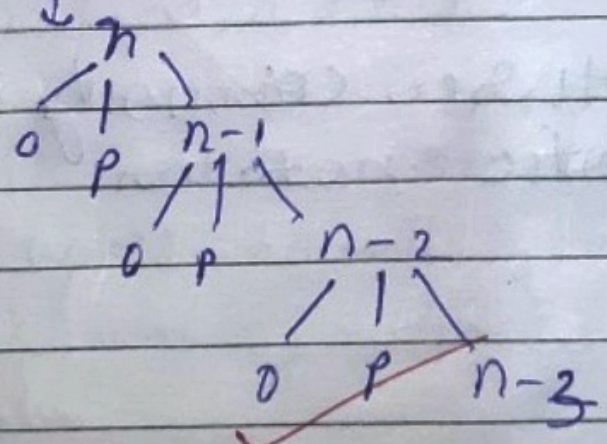
```
quicksort (A, p, r)
{
  if (p < r)
  {
    q = partition (A, p, r)
    quicksort (A, p, q-1)
    quicksort (A, q+1, r)
  }
}
```

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:

Q3 Define asymptotic notations. Give different notations used to represent the complexity of algorithms.

Ans →

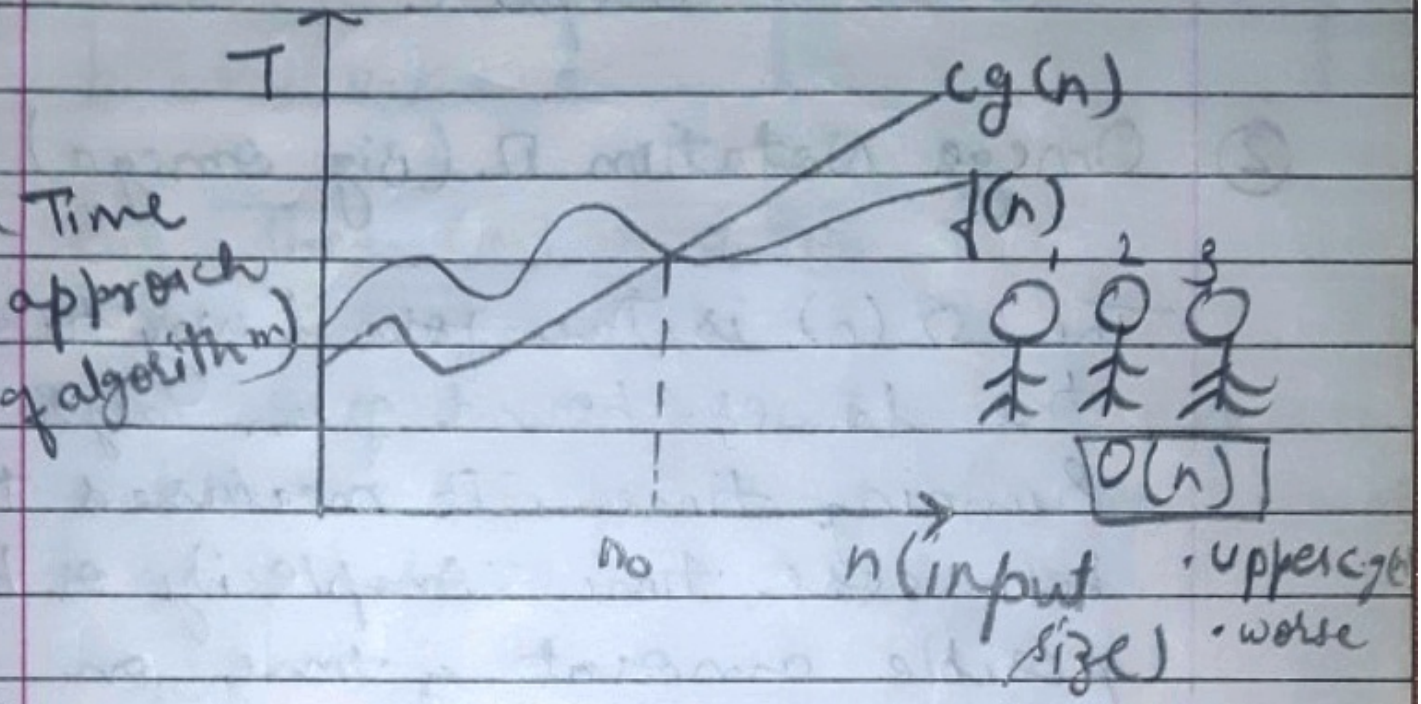
Asymptotic notations:-

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

There are three commonly used asymptotic notation.

Date: / / Page no:

1 Big O notation (- upper bound function of $f(n)$)



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

$n \geq n_0$

$c > 0, n_0 \geq 1$

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

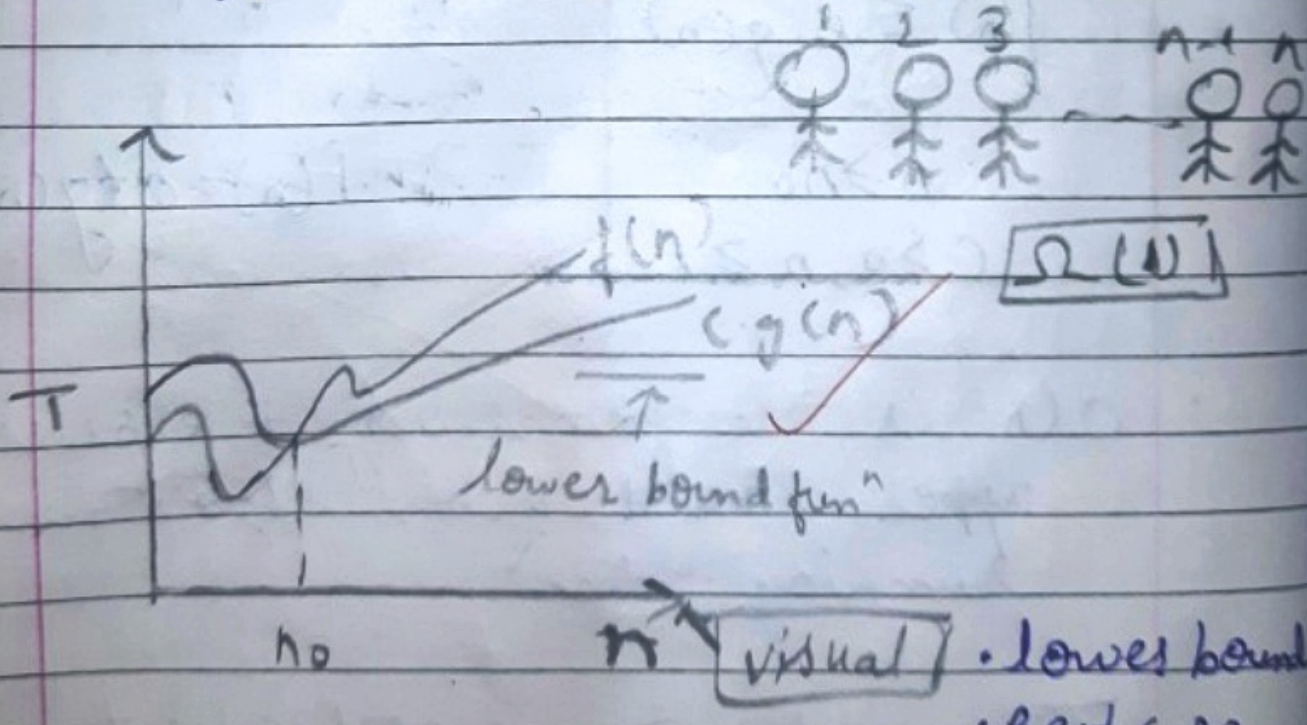
- $O(n)$ is the formal way to express the upper bound of an algorithm running time.
- It measures the worst case time complexity or longest amount of

Date: / / Page no:

time an algorithm can possibly take to complete

② Omega Notation Ω (Big Omega) :-

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



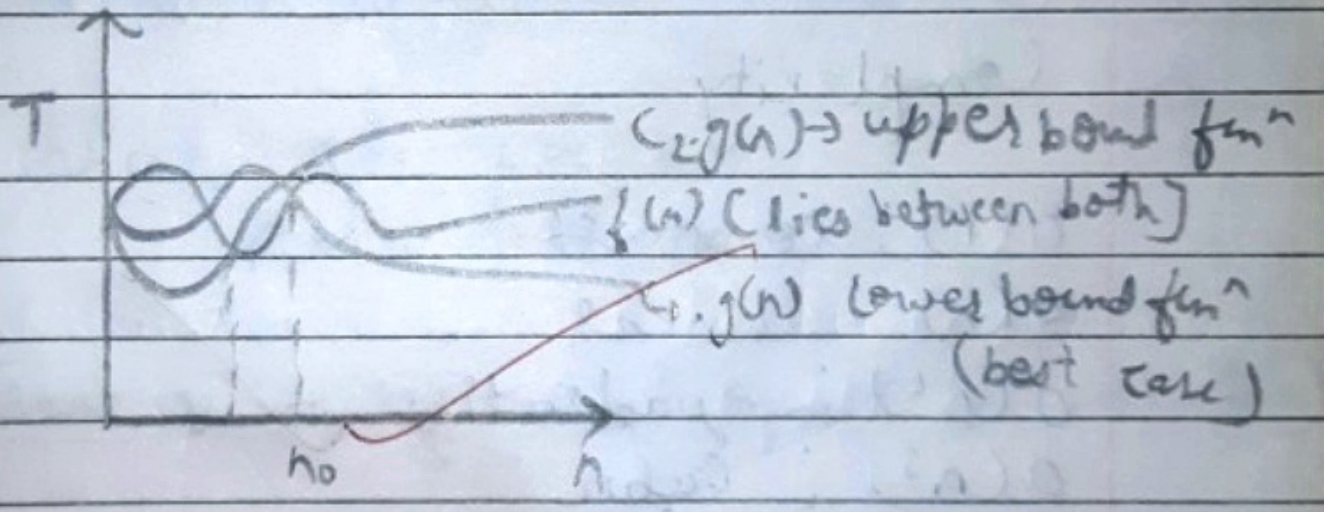
- lowest bound
- Best case

Date: / / Page no:

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

(3) Big Theta (Θ) Notation

The $\Theta(n)$ is the formal way to express both the lower bound and upperbound of an algorithm's running time



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

Date: / / Page no:

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

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

O	upper bound	worst case
Ω	lower bound	Best case
Θ	Between	Average case

Complexity

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

Date: / / Page no:

Q4 write all the three cases of master Theorem for the equation

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

Sol:- Master Theorem :- for solving recurrence relation.

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

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

p is a real number

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

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 $T(n) = \Theta(n^{\log_b^a} \log \log n)$

Date: / / Page no:

(c) If $p < -1$ then $T(n) = \theta(n \log_b^p)$

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 $T(n) = \theta(n^k)$

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

Ans. It is used to solve the matrix multiplication problem.

- The usual matrix multiplication method multipiles each row vast each column to achieve the product matrix time complexity taken by this approach is $O(n^3)$ Since it takes three loops to

Date: / / Page no:

multiply.

This method was introduced to reduce the time complexity from $O(n^3)$ to $O(n^{\log 7})$

* Naive method of martix multiplication

Let x is a martix of order pxq
y is a martix of order qxr.

$$Z = [x]_{pxq} [y]_{qxr}$$

Z = has the order pxr

Algorithm ~~martix~~ - multiplication (x, y, z)

```
for i = 1 to p
    j = 1 to r
    z[i, j] = 0
```

for $k = 1$ to q do

$z[i,j] = z[i,j] + x[i,k] \times y[k,j]$

complexity $= O(n^3)$

SMM Algorithm :-

with this algorithm the time consersption can be improved a little bit.

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

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

$M_1 := (A+C) \times (E+F)$

$M_2 := (B+D) \times (G+H)$

$M_3 = (A-D) \times (E+H)$

Date: / / Page no:

$M_4 = A \times (E - H)$

$M_5 = (C + D) \times E$

$M_6 = (A + B) \times H$

$M_7 = D \times (G - E)$

Then,

$I = M_2 + M_3 - M_6 - M_7$

$J = M_4 + M_6$

$K = M_5 + M_7$

$L = M_1 - M_3 - M_4 - M_5$

Analysis :-

$T(n) = ?c$, if $n = 1$

$(T \times T(\frac{n}{2}) \times d \times n^2$, otherwise

where c and d are constants

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

Wednesday
11/3/24.