9-3-2024

Date / /
Page

Assignment - 1

Q What is ADA? What is the need to Study Algorithem. explain in detail?

⇒ ADA:-

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

- Computational means which can be Solved by on algorithm.

* Need to Study Algorithem:-

- Problem-Solving:- Algorithem provide Systemtic approaches to Solving Various computional problems By Studying algorithms, you learn how to Break Complex problem.

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

- **Scalability:-** As the size of data and computional tasks continue to grow, the importance of scalable algorithms becomes paramount.

- **Optimization:-** Algorithem often provide multiple solutions to the same problem, with varying trade-offs in terms of time speed.

- **Foundation for Computer Science:-**

Algorithms forms the backbone of Computer Science. They are fudamental 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 techinal interviews.

Q) Give the Divide and Conquer for QuickSort and analyse is Complexcity?

⇒ **Divide and Conquer Technique:-**

- DSC is a top down approach to solve a problem the algorithem which involes
- DSC tenchiques involes 3 steps:-

- **Divide:-** The Original problems into a set of sub problems.

- Conquer (or Solve):- every sub-problem is solved by recursive.
- Combine:- The solutions of these sub-problems to get the solutions of original problem.

* :- QUICK SORT (For Sorting)

- First 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

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 pivot
A[r] = x
So, x = 4

→ // 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)
}
```

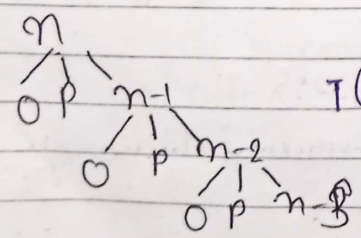
→ // Algorithem :-

```
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 :- $O(n \cdot \log n)$
- Worst-Case :- $O(n^2)$ - (when the array is already sorted sorted)

↓



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

$O(n^2)$

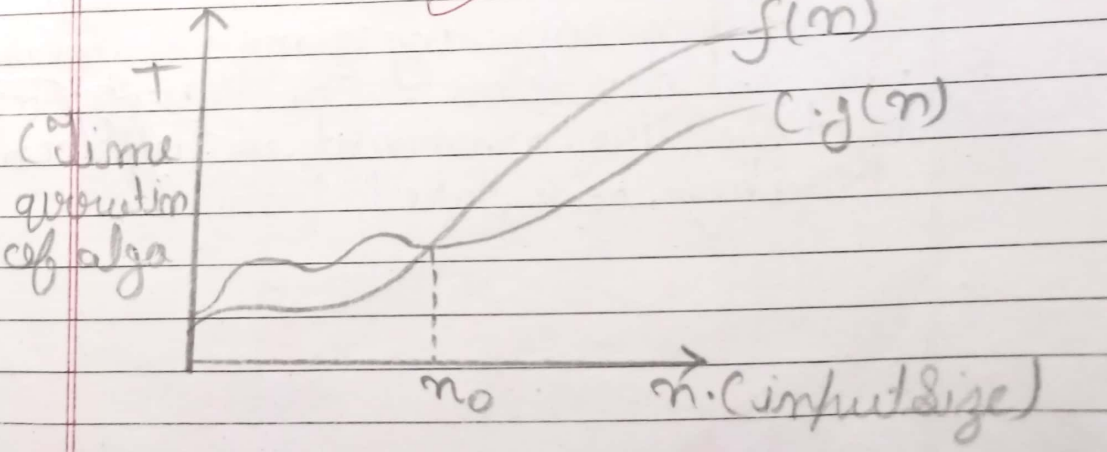
Q Define asymotic notions. Give different notations used to represent the Complexcity of algorithem.

⇒ Defination :-

- Asymption notations are Mathematical representation used in computer Science to describe the running time or Space Complexcity of an algorithem as the input size approches infinity.

* Different types notations :-

(i) Big O Notation :-



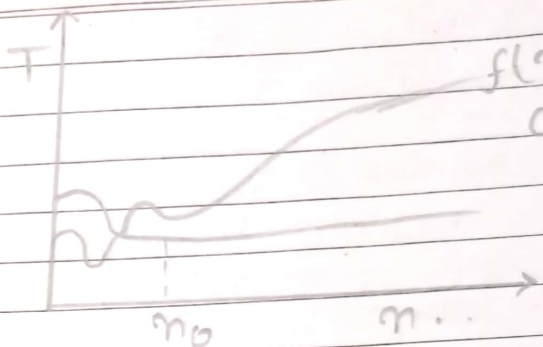
$f(n) \leq C \cdot g(n)$
$n \geq n_0$
$C > 0,\ n_0 \geq 1$
$\Rightarrow$ $f(n) = O(g(n))$

- $O(n)$ is formal way tenhores the upper bound of an algorithem running time.

- It measures the worst case time Comperity or longest ammount of time an algorithem can possible take to Complete.

(2) Omega Notation, Ω (Big omega) :-

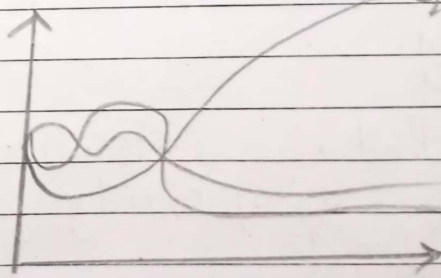
The $\Omega(n)$ is the lower bond of an algorithem running time. It measures the best case time Complexicity or best possible ammount of time an algorithem can take to Comple.



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

(3) Big theta (Θ) Notation:-

The $\Theta(n)$ the formal way to enforces both the lower bound and upper bound of an algorithem running time



- This graph represents the time growth of an algorithem

$$C_2\, g(n) \geq f(n) \geq C_1 \cdot g(n) \quad n \geq n_0 \quad C_1, C_2 > 0 \; n_0 \geq 1 \Rightarrow f(n) = \Theta g(n)$$

Q5) How can we prove that Strassen's Mardix multiplication is adurantages over ordinary Mardix multiplication.

⇒

- It is used to Solve the mardix multiplicaⁿ problem.
- The usual mardix multiplication method multiplies each row with each column to achieve the product mardix. 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 $O(n^3)$ to $O(n^{\log_2 7})$

* Naive Method of Mardix Multiplication:-

Let X is a mardix of ordes $p \times q$
Y is a matix of order $q \times r$

$Z = [X]_{p \times q} \cdot [Y]_{q \times r}$

Z has the order $p \times r$

```
Algorithem Mardix - 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 Algorithem:- With this algo. the time consarsption can be improved a little bit.

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

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

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

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

Then,
$I := M_2 + M_3 - M_6 - M_7$
$J := M_4 + M_6$
$K := M_5 + M_7$
$L := M_1 - M_3 - M_4 - M_5$

Analysis :-

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

↳ where c and d are Constants

We get $T(n) = O(n^{\log_2 7})$

Q) 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

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

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

Case 1:⇒ If $a > b^k$ then $T(n) = O(n^{\log_b a})$

Case 2:⇒ If $a = b^k$

If $P > -1$ then $f(n) = \Theta(n^{\log_b^a} \log^{P+1} n)$

If $P = -1$ then $f(n) = \Theta(n \log_b^a \log \log n)$

If $P < -1$ then $f(n) = \Theta(n \log_b^a)$

If $a < b^k$

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

If $p < 0$ then $T(n) = O(n^k)$

BA Answered
11/3/24