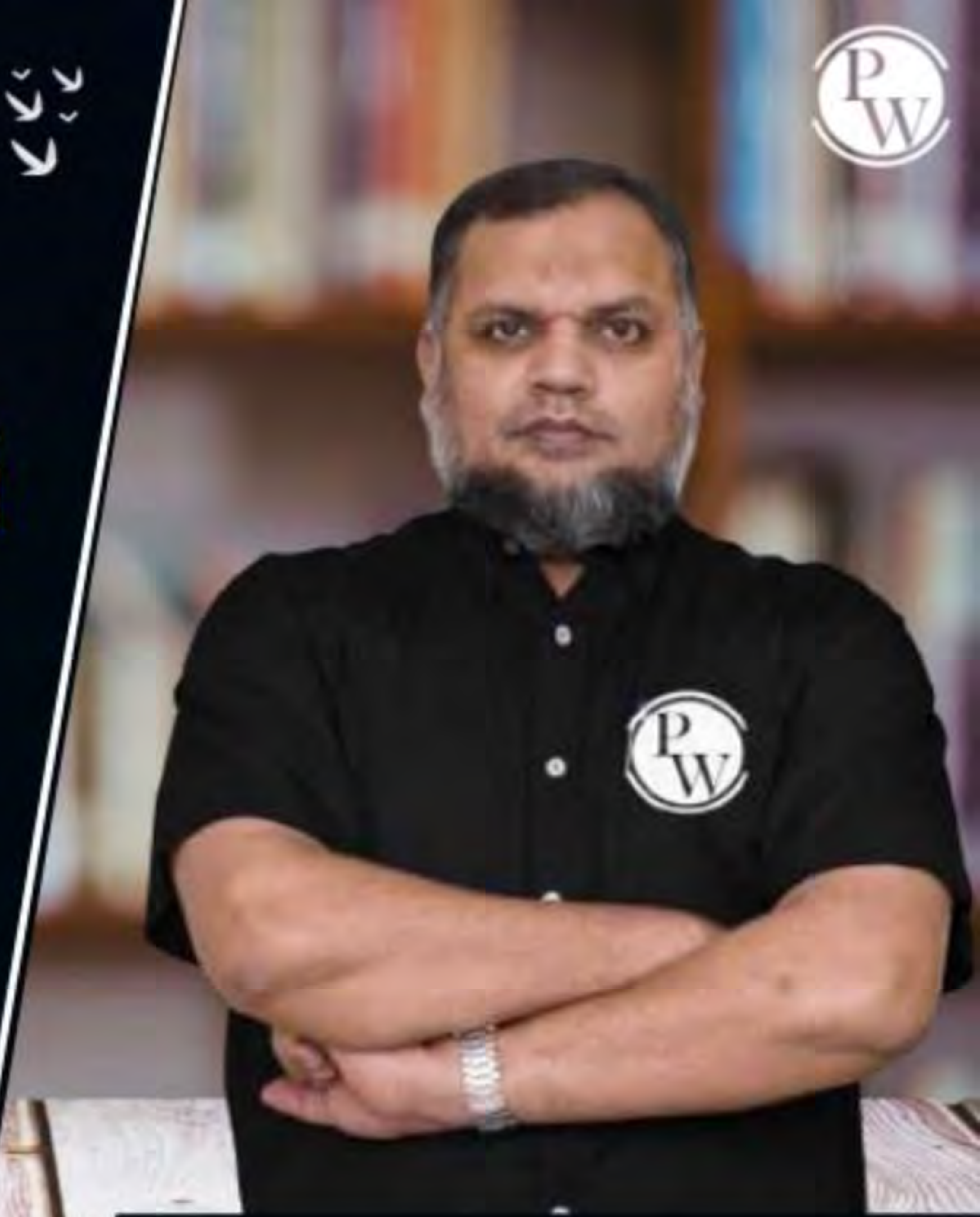
CS & IT ENGINEERING

Algorithms

[redacted]

Lecture No.- 02

1500 Series



By- Dr. Khaleel Khan sir

Recap of Previous Lecture

Topic ALGORITHMS 01

Topics to be Covered

Topic Analysis of Algorithms

Topic Divide and Conquer

#Q. Let f(n) and g(n) be asymptotically positive functions then consider the following statements

$\langle a+b \rangle$

(a) $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$ ×

(b) $f(n) = O((f(n))^2)$ ×

A. a is false ✓

B. b is false ✓

C. a, b are true

D. a is true

1) $f(n) = n \; ; \; g(n) = n^2$

$2^n = O(2^{n^2})$

2) $f(n) = 2n \; ; \; g(n) = n$

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

2^{2n} is it $O(2^n)$ ×

3) $f(n) = n$

n is it $O(n^2)$ ✓

4) $f(n) = 2^n$

2^n is it $O((2^n)^2)$ ✓

5) $f(n) = \frac{1}{n}$

$1/n$ is it $O(1/n^2)$ ×

#Q.

```
for (int i = 2; i ≤ n; i *=i)
{
    for (int j = 2; j ≤ i; j*=2)    log i
    {
        print( " * " );
    }
}
```

The Time Complexity of the code snippet is ________________.

$i=2 \quad i=2*2 \quad i=4\times4 \quad i=16\times16 \quad \ldots$

$$\text{Time} = \left[\log_2 2^{2^0} \quad \log_2 2^{2^1} \quad \log_2 2^{2^2} \quad \log_2 2^{2^3} \quad \ldots \quad \log_2 2^{2^k}\right]$$

$$= \log_2 2^{2^0+2^1+2^2+\cdots+2^k} = \log_2 2^{(2^{k+1}-1)} = (2^{k+1}-1) = (2^{\log\log n+1}-1) = O(\log n)$$

$$2^{2^k} = n \Rightarrow k = \log\log n$$

a) $\sqrt{n}\cdot\log n$

b) $\log n$

c) $n\log n$

d) n^2

e) $(\log n)^2$

f) $n\cdot\log\log n$

g) $\log n * \log\log n$

#Q. The following function recursively determines whether a given string is a Palindrome. Determine its Time Complexity.

```
int isPalindrome (char A[ ], int n)
{
    if (n ≤ 1) return 1;
    if (A[0] != A[n–1]) return 0;
    return isPalindrome (&A[1], n–2);
}
```

Let $T(n)$ Time complexity of isPalindrome(n)

$$T(n) = c, \quad n=1$$
$$= T(n-2)+a,$$

$$T(n) = T(n-2)+a \quad -1$$
$$= T(n-4)+2a \quad -2$$
$$= T(n-6)+3a \quad -3$$
$$= T(n-2\cdot 3)$$
$$= T(n-2\cdot k)+k\cdot a$$
$$= T(1)+\frac{a(n-1)}{2}$$
$$= O(n)$$

$n-2k=1$

$\frac{n-1}{2}=k$

#Q. The running time of the following procedure

procedure foo(n)

{

if (n < =2) return (1)

else

return $\left(\underbrace{foo(\sqrt{n})}_{\text{Value}_1} + \underbrace{n}_{\text{Value}_2}\right)$

}

is best described as

A 0 (loglogn)

B 0(1)

C 0(log n)

D $0(2^n)$

```
main()
{
  int n, m;
  n=10; m=20;
  return (n+m);
}
```

$O(1)$

$O(\log\log n)$

Let $T(n)$ repr. T.C of foo(n)

$T(n) = c, \; n \leq 2$

$= a + T(\sqrt{n}) + b, \; n > 2$

$= T(\sqrt{n}) + d, \; n > 2$

```
Procedure Foo(n)
{
    if (n ≤ 2) return(1);
    else
        return(Foo(√n) + B(n));
}
```

$O(B(n))$

$T(n) = c \quad , \quad n \le 2$

$= a + T(\sqrt{n}) + O(B(n)) + a$

#Q. Let T(n) be a recurrence relation such that $T(n) = 4T(\sqrt{n}) + c$ $n > 2$ ~~0~~ 2 otherwise

(Time Complexity)

What is the maximum depth of recursion required?

A $\lceil \log_4 n \rceil$

B $\lceil \log_2 \log_2 n \rceil$

C $\lceil \log_4 \log_4 n \rceil$

D $\lceil \log_4 n \rceil$

$$T(n) = 4 \cdot T(n^{1/2}) + c$$

$$T(n^{1/2}) = 4 \cdot T(n^{1/2^2}) + c$$

$$= 4 \cdot T(n^{1/2^k})$$

$$n^{1/2^k} = 2$$

$$K = \lceil \log_2 \log_2 n \rceil$$

$$T(2^k) = 4 \cdot T(2^{k/2}) + c$$

$$\boxed{S(k) = 4 \cdot S(k/2) + c}$$

$$c \text{ is } O(k^{2-\epsilon})$$

$$\therefore T(k) = O(k^2)$$

$$= O((\log n)^2)$$

1) $T(n) = T(n/2) + c \rightarrow$ Binary Search,

2) $T(n) = 2 \cdot T(n/2) + \underline{\underline{f(n)}} \rightarrow$ M.S / Q.S
n

$\Big\}$ $\log n$

$$T(n) = T(n/2) + c$$

$$T(n/2) = T(n/4) + c$$

$$= T(n/4) + 2c$$

$$\vdots$$

$$= T(n/2^k) + k \cdot c$$

k = depth of stack

$$\frac{n}{2^k} = 1$$

$$\Rightarrow n = 2^k$$

$$\boxed{k = \log n}$$

#Q. Consider the following recursive program

```
void abc(int n)
{
int x = 0;
if (n>1)
{
    abc(n - 1);
    abc(n - 1);
}
for (a = 0; a <n; a++)
printf("$");
}
```

$T(n) = 2 \cdot T(n-1) + n$

What is the recurrence relation for the given program ?

A. $T(n) = 2T(n-1)+1$

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

C. $T(n) = 2T(n - 1) + n$

D. None of these

#Q. Match the Pairs

A	$O(\log n)$	(P)	Merge sort
B	$O(n)$	(Q)	Binary Search
C	$O(n \log n)$	(R)	Insertion sort
D	$O(n^2)$	(S)	Partitioning / Pivoting

#Q. An array 'A' contains n integers ~~in increasing order~~. The Algorithm to find two indices 'i' & 'j' such that A[i] + A[j] = 'M' a given integer, if they exists, will take:

(Most efficient Algorithm)

A. Constant time

B. Logarithmic time

C. Linear Polynomial time

D. Exponential time

1) Brute_Force $= O(n^2)$

2) using Bin-Srch $= O(n \cdot \log n)$

3) Two-Pointer [Partition-Proc] $O(n)$

#Q. Which of the following has the lowest worst-case complexity

[Priority-Q] Heap : log n

A Shortest job first

B [Dijkstra's Banker's Algorithm] $O(n^2)$

C Round Robin : $O(1)$

↳ circular

L.L

D [Shortest seek time first] Heap

Evaluatis

$\frac{n}{2^k} = 1 \Rightarrow k = \log_2 n$

#Q. The solution to the recurrence equation T(n) = 3T (n/2) +1, T(1) =1 is-

A $2^{\log 3} . \log n$

B $3^{\log_2 n + 1} - 2$

C $3^{\log_2 n}$

D None of the above

$$T(n) = 3 \cdot T(n/2) + 1 \quad - (1)$$

$$T(n/2) = 3 \cdot T(n/4) + 1 \quad - (2)$$

$$T(n) = 3\left(3 \cdot T(n/4) + 1\right) + 1$$

$$= 9 \cdot T(n/4) + 3 + 1 \quad - (3)$$

$$= 3^2 \cdot T(n/2^2) + \sum_{i=0}^{1} 3^i$$

$$= 3^3 \cdot T(n/2^3) + \sum_{i=0}^{2} 3^i$$

$$= 3^k \cdot T(n/2^k) + \boxed{\sum_{i=0}^{k-1} 3^i}$$

$$= 3^k \cdot 1 + \frac{3^k}{2} - \frac{1}{2}$$

$$= \frac{3}{2} \cdot 3^k - \frac{1}{2}$$

$$= \frac{3^{k+1}}{2} - \frac{1}{2}$$

$$= \frac{1}{2}\left[3^{\log_2 n + 1} - 1\right]$$

$$O\left(n^{\log_2 3}\right)$$

$$S_n = \frac{a(r^n - 1)}{r - 1} = \frac{1(3^k - 1)}{3 - 1} = \left(\frac{3^k - 1}{2}\right)$$

#Q. Let T(n) be defined by $T(1) = 7$ and $T(n + 1) = 3n + T(n)$ for all integers $n \geq 1$. Which of the following represents the order of growth of T(n) as a function of n?

$T(n+1) = 3n + T(n)$

$$\boxed{T(n) = T(n-1) + 3(n-1)}$$

$T(1) = 7$

A $\Theta(n)$

B $\Theta(n \log n)$

C $\Theta(n^2)$

D $\Theta(n^2 \log n)$

E $\Theta(2^n)$

$$\boxed{T(n) = T(n-1) + n}$$

$\rightarrow O(n^2)$

#Q. Let $T(n)$ be defined by $T(0) = T(1) = 4$ and $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + cn$ for all integers $n \geq 2$, where c is a positive constant. What is the asymptotic growth of $T(n)$?

Cost

A $\Theta(\log n)$

B $\Theta(n)$

C $\Theta(n \log n)$

D $\Theta(n^2)$

E $\Theta\left(n^{\log_3 4}\right)$

$T(n) = T(n/2) + T(n/4) + n$

⟨Recursion Tree⟩

$T(n) \longrightarrow cn$

$T(n/2) \quad T(n/4) \longrightarrow \left(\frac{n}{2} + \frac{n}{4}\right) = \frac{3n}{4}$

Max

$\boxed{T(n/4)} \quad T(n/8) \quad T(n/8) \quad \boxed{T(n/16)} \longrightarrow \left(\frac{3}{4}\right)^2 n$

$T(n/2^k)$

Min

$T(n/4^k)$

#Q. Consider a recursive algorithm for sorting an array of $n \geq 2$ integers that works as follows.

(a) If there are only 2 elements to be sorted, compare them and swap them if they are out of order.

(b) Otherwise, do the following steps in order.

(1) Recursively sort the first $n-1$ elements of the array.

(2) In the resulting array, recursively sort the last $n-1$ elements.

(3) In the resulting array, recursively sort the first 2 elements of the array.

What is the asymptotic running time complexity of this algorithm measured in terms of the number of comparisons made?

A $\Theta(n \log n)$

B $\Theta(n^2)$

C $\Theta(n^3)$

D $\Theta(2^n)$

E $\Theta(3^n)$

$T(n) = c \; , \; n = 2$

$= T(n-1) + T(n-1) + a \; , \; n > 2$

$= 2 \cdot T(n-1) + a$ — 1

$O(2^n)$

#Q. Mergesort works by splitting a list of n numbers in half, sorting each half recursively, and merging the two halves, Which of the following data structures will allow mergesort to work in O(n log n) time?

I. A singly linked list

II. A doubly linked list

III. An array

A. None

B. III only

C. I and II only

D. II and III only

E. I, II and III

#Q. The time complexity to multiply two long integers of size n- digit represented in an array of size n, using divide & conquer strategy with 3-way split is-

A $O(n)$

B $O(n\log n)$

C $O(n^{1.32})$

D $O(n^2)$

2 mins Summary

Topic One

Topic Two

Topic Three

Topic Four

Topic Five

THANK - YOU