

# CS & IT ENGINEERING

## Algorithms

Analysis of Algorithm

Lecture No. - 09

By- Dr. Khaleel Khan  
Sir



# Recap of Previous Lecture



Topic

Problem Solving with ASNs

Topic

Analysis of Recursive Algorithms

Topic

Topic

Topic



# Topics to be Covered



## Topics

Framework for Analysing Recursive Algo

Loop Complexities







## Topic: Asymptotic Notations

$$T(n) = 8T(n/2) + C, \quad n > 1$$
$$= a, \quad n = 1$$

$$T(n) = 8T(n/2) + C \quad \text{--- (1)}$$

$$T(n/2) = 8T(n/4) + C \quad \text{--- (2)}$$

$$T(n) = 8(8T(n/4) + C) + C$$

$$= 64T(n/4) + 9C \quad \text{--- (3)}$$

$$= 8^2 T(n/2^2) + (2^3 + 1)C \quad \text{--- (4)}$$

$$= 8^k T(n/2^k) + (2^{k+1} + 1)C$$

$$T(n) = 8^{\log_2 n} T(1) + \dots$$

$$= n^{\log_2 8} C + \dots$$

$$= n^3 + \dots$$

$$O(n^3) \checkmark$$



$$T(n) = 2 \cdot T(n/2) + n \cdot \log n \quad \text{--- (1)}$$

$$T(n/2) = 2 \cdot T(n/4) + n/2 \cdot \log n/2 \quad \text{--- (2)}$$

$$T(n) = 2 \left[ 2 \cdot T(n/4) + n/2 \cdot \log n/2 \right] + n \cdot \log n \quad \text{--- (3)}$$

$$= 4 \cdot T(n/4) + n \cdot \log_2 n/2 + n \cdot \log n/2$$

$$= 2^2 \cdot T(n/2^2) + n \cdot \log_2 n/2 + n \cdot \log n/2$$

$$\vdots$$

$$= 2^k \cdot T(n/2^k) + n \cdot \log_2 n/2^{k-1} + \dots + n \cdot \log_2 n/2^0 \quad \text{--- (4)}$$

$$= 2^k \cdot T(n/2^k) + n \left[ \log_2 n/2^{k-1} + \log_2 n/2^{k-2} + \dots + \log_2 n/2^0 \right]$$

$$= 2^k \cdot T(1) + n \cdot \left[ \log_2 \left( \frac{n^k}{2^{0+1+2+\dots+k-1}} \right) \right]$$

$$= 2^k \cdot T(1) + n \left[ \log_2 n^k - \log_2 2^{\frac{k(k-1)}{2}} \right]$$

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

$$= 2^k \cdot c + n \left[ k \cdot \log n - \frac{k(k-1)}{2} \right]$$

$$= 2^k \cdot c + n \left[ 2k \cdot \log n - k^2 + k \right]$$

$$= n \cdot c + \frac{n}{2} \left[ 2 \log^2 n - \log^2 n + \log n \right]$$

$$= cn + \frac{n}{2} \left[ \log^2 n + \log n \right]$$

$$= cn + \frac{1}{2} n \log^2 n + \frac{n \cdot \log n}{2}$$

$$O(n \cdot \log^2 n) \checkmark$$

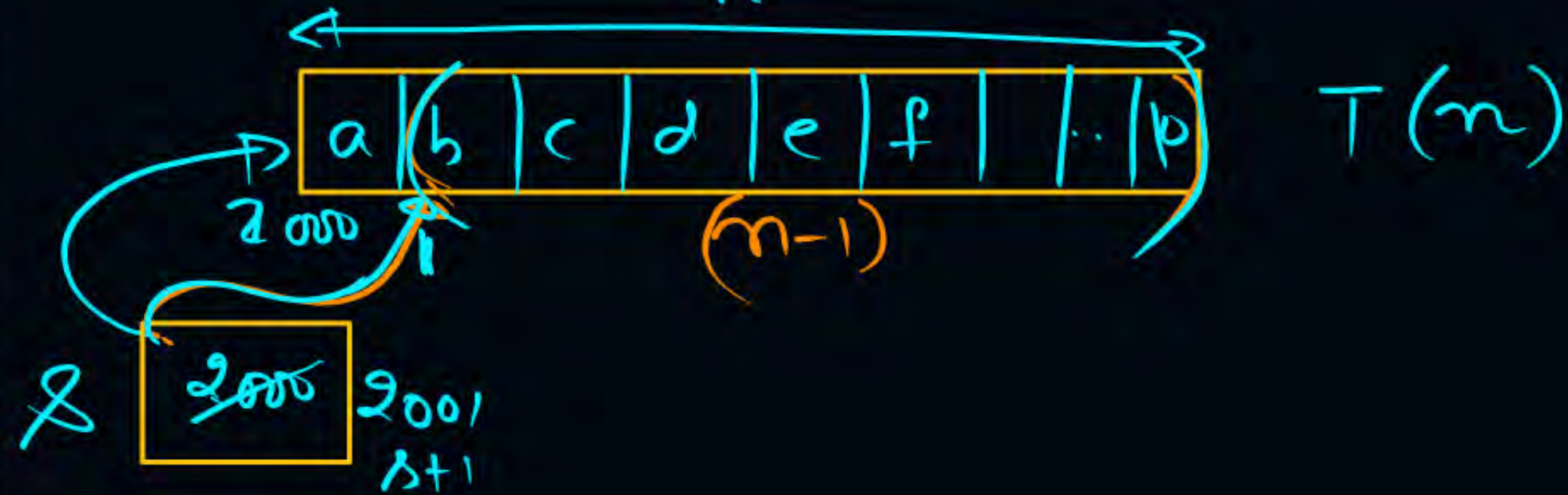


```

8. void abc(char *s)
{
    if (*s != '\0')
    {
        'a' printf("%c", *s);
        'b' abc(s + 1);  $T(n-1)$ 
        abc(s+1);  $T(n-1)$ 
    }
}

```

Let 's' be a pointer to a String of length 'n' char's  
What is the Time Complexity of this Program;



$$T(n) = C, n=1$$

$$= a + b + 2.T(n-1), n > 1$$

$$T(n) = 2.T(n-1) + d, n > 1 \quad d > 0$$

$$\rightarrow \underline{\underline{O(2^n)}}$$



integer  $A[n]$ ;

Algorithm  $RSum(A, n)$   
{

if  $(n=1)$  return  $(A[1])$ ;

else return  $(RSum(A, \underline{n-1}) + \underline{A[n]})$ ;

}

$T(n-1)$

$a$

$T(n)$

$$T(n) = c, \quad n=1$$

$$= T(n-1) + d, \quad n > 1$$

$$\rightarrow \underline{\underline{O(n)}} \checkmark$$

Algorithm DO-it( $n$ )

{

if ( $n=1$ ) return,

else

return (DO-it( $n-1$ ) +  $n$ );

}

val<sub>1</sub>

val<sub>2</sub>

Const

$$T(n) = c, \quad n = 1$$

$$= a + T(n-1) + b, \quad n > 1$$

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

$$O(n)$$



for  $i \leftarrow 1$  to  $n$

for ( $i=1; i \leq n; ++i$ )

$\left\{ \begin{array}{l} s_1 \\ s_2 \end{array} \right\} \Rightarrow$  Total times the loop repeats: ' $n$ '

1) for  $i \leftarrow 1$  to  $n$  :  $n$

$c=0;$   
 $c=c+1; \quad O(1)$

$$T(n) = \sum_{i=1}^n O(1)$$

Time:

$i=1$	$O(1)$
$i=2$	$O(1)$
$i=3$	$O(1)$
$i=4$	$O(1)$
$\vdots$	
$i=n$	$O(1)$
	$\underline{\underline{(n)}}$

Time Complexity

depends on No. of times the loop is repeat ( $n$ ) & the complexity of the statements in the body of loop



for  $i \leftarrow 1$  to  $n$  }  $(n+1)$   $\underline{n+c} \Rightarrow O(n)$   
 $c = c+1;$  }  $O(n)$   
 $c = n$

---

$c = 0$   
 for  $i \leftarrow 1$  to  $n$

$c = c+i;$   $c = (1+2+3+4+\dots+n) = \frac{n(n+1)}{2} = O(n^2)$

1) Time:  $O(n)$

2) Value( $c$ ):  $O(n^2)$

for  $i \leftarrow 1$  to  $n/2$  : Time:  $O(n)$   
 $c = c+1$  Value:  $O(n)$



for  $i \leftarrow 1$  to  $n$   
break;  $\therefore O(1)$

Memd(x);  
for  $i \leftarrow 2$  to  $n$

if  $(x \% i == 0)$   
break;

Body

Worst Case :  $O(n)$

Best Case :  $O(1)$

for  $i \leftarrow 1$  to  $n$   
 $B(n)$ ;

Time :  $O(n * O(B(n)))$

(i) if  $B(n) = O(1) \Rightarrow O(n)$

(ii) if  $B(n) = O(\log n) \Rightarrow O(n \cdot \log n)$

(iii) if  $B(n) = O(n) \Rightarrow O(n^2)$



$c=0$   
for ( $i=1; i \leq n; ++i$ );  $i=n+1$

$c=c+i; \text{"1"}$

Time:  $O(n)$

Value of  $c = \underline{n+1}$

$c=0;$   
while (1) {  
    Time:  $\infty$   
     $c=c+1;$   
}

while-loop:

while (condition)  
{

$s_1;$

$s_2;$

$s_k;$   
}

$i=1;$

while ( $i \leq n$ )  
{

$i=i+1;$  } 'n'

$O(n)$



## Nested loops:

(i) *Independent*

for ( $i=1; i \leq n; ++i$ )

{  
  for ( $j=1; j \leq n; ++j$ )  
  {  
     $c=c+1$ ;  
  }  
}

$i=1 \quad j: 1 \text{ to } n : n$   
 $i=2 \quad j: 1 \text{ to } n : n$   
 $i=3 \quad j: 1 \text{ to } n : n$   
  :  
 $i=n \quad \quad \quad n$   
}  $n \times n = n^2$

## Mutually Exclusive loops

{

1. for  $i \leftarrow 1 \text{ to } n : n$   
    $c=c+1$ ;

2. for  $j \leftarrow 1 \text{ to } m : m$   
    $k=k*2$ ;

}

Time:  $O(n+m)$   
       $= O(\max(n, m))$



$\rightarrow O(n^2)$   
for  $i \leftarrow 1$  to  $n$  :  $n$   
{

a)  $B(n) : \underline{\log n}$

b) for  $j \leftarrow 1$  to  $n/2$  }  $O(n)$   
     $c = c + 1;$

$k = 0$   
    c) while  $(k \leq n)$  }  $O(n)$   
         $++k;$

}

if  $B(n) = O(\log n)$

Time :  $O($

Per It of 'i' :  $(\log n + O(n) + O(n))$   
                  :  $O(n)$

Total time =  $n * (\log n + n + n)$   
              =  $n \cdot \log n + n^2 + n^2 = O(n^2)$





## Topic : Running Times of Program Segments with Loops:

Loop-Complexities \*

1. for i  $\leftarrow$  1 to n  
    c = c + 1;

$O(n)$

2. for i  $\leftarrow$  1 to n  
    for j  $\leftarrow$  1 to n/2  
        c = c + 1;

$n$

$n/2$

$(n * \frac{n}{2}) = O(n^2)$

3. for i  $\leftarrow$  1 to n :  $n$

for j  $\leftarrow$  1 to n/4 :  $n/4$

for k  $\leftarrow$  1 to n

break;

: 1

Time:  $n * \frac{n}{4} * 1$

$O(n^2)$



**THANK - YOU**