

# Data Structure and Algorithms

Session-15

Dr. Subhra Rani Patra SCOPE, VIT Chennai

## Time Complexity

for (i = 0; i < 0; itt) Itatement; for Ci = 7; 1.70; 1--) statement;

3 for (i=1, i < n; its) statement; 9 for (i=0; i<n; i+) for cj =0; j <1; j ++) statement;

5) for (i -0; i <1; 1th)

/par (5=0; j <i; jth)

i	j	roof times	
0	O	0	
1	0		$1+2+3+\dots+n$ $= 2 + n$ $= 2 + n$ $= 2$ $0 + n$ $= 2$
2	07	2	
3	2 0 7 2	3	
1: 0	3	)	

# Time Complexity

fr(i =1; i <7; i=i\*2)

Statement;

Assume i >1,7

$$2 \times 2 = 2^{2}$$
 $2 \times 2 = 2^{3}$ 
 $i = 2^{4}$ 
 $2^{1} \times 2^{1}$ 
 $2^{1} \times 2^{2}$ 
 $2^{2} \times 2$ 

# Time Complexity

**Formal Definition:** f(n) = O(g(n)) means there are positive constants c and  $n_0$ , such that  $0 \le f(n) \le cg(n)$  for all  $n \ge n_0$ .

1 < logn< 
$$\frac{1}{3}$$
 < n < n < n < n < n < n^2 < n^3 - - < 2^2 < 3^2 - < 1

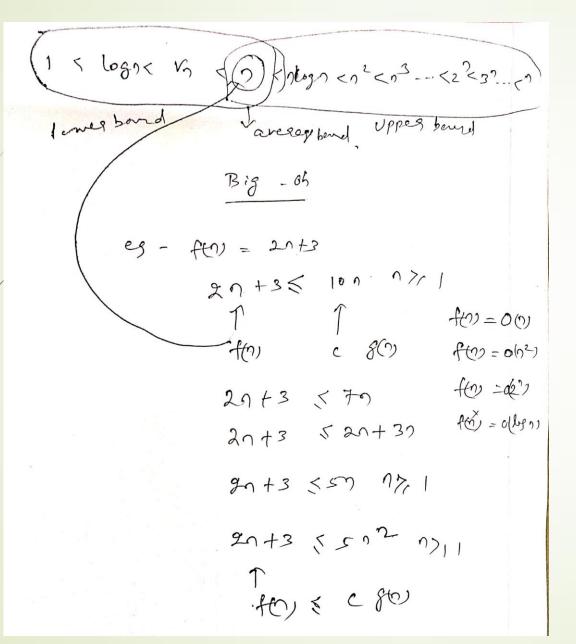
Big - 6h

eg - ftn = 2n + 3

2n + 3 < 100 n > 1

$$e_3 - f(n) = 2n + 3$$
 $2n + 3 \le 10n n > 1$ 

1 
$$f(0) = 00$$
  
 $f(0) = 00$   
 $2n + 3 < 70$   
 $2n + 3 < 50 + 30$   
 $2n + 3 < 50 + 30$ 



**Formal Definition:**  $f(n) = \Omega(g(n))$  means there are positive constants c and  $n_0$ , such that f(n) >= cg(n) for all  $n \ge n_0$ .

$$f(n) = \mathcal{N}(n)$$

$$f(n) = \mathcal{N}(\log n)$$

**Formal Definition:**  $f(n) = \theta(g(n))$  means there are positive constants  $c_1, c_2$  and  $n_0$ , such that  $c_1g(n) <= f(n) <= c_2g(n)$  for all  $n \ge n_0$ .

eg: 
$$f(0) = 2n + 3$$
  
 $(x_1 < 2n + 3 \le 5 \times 7)$   
 $(x_2 < 2n + 3 \le 5 \times 7)$   
 $(x_3 < 2n + 3 \le 5 \times 7)$   
 $(x_3 < 2n + 3 \le 5 \times 7)$ 

Thank,