# CS & IT ENGINEERING

Algorithms



### Recap of Previous Lecture









## **Topics to be Covered**









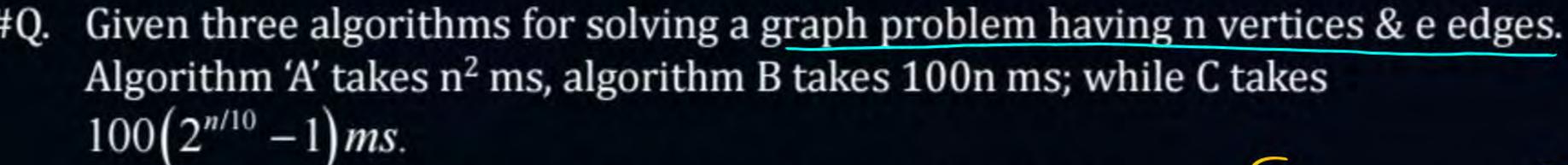
Topic

**Analysis of Algorithms** 

Topic

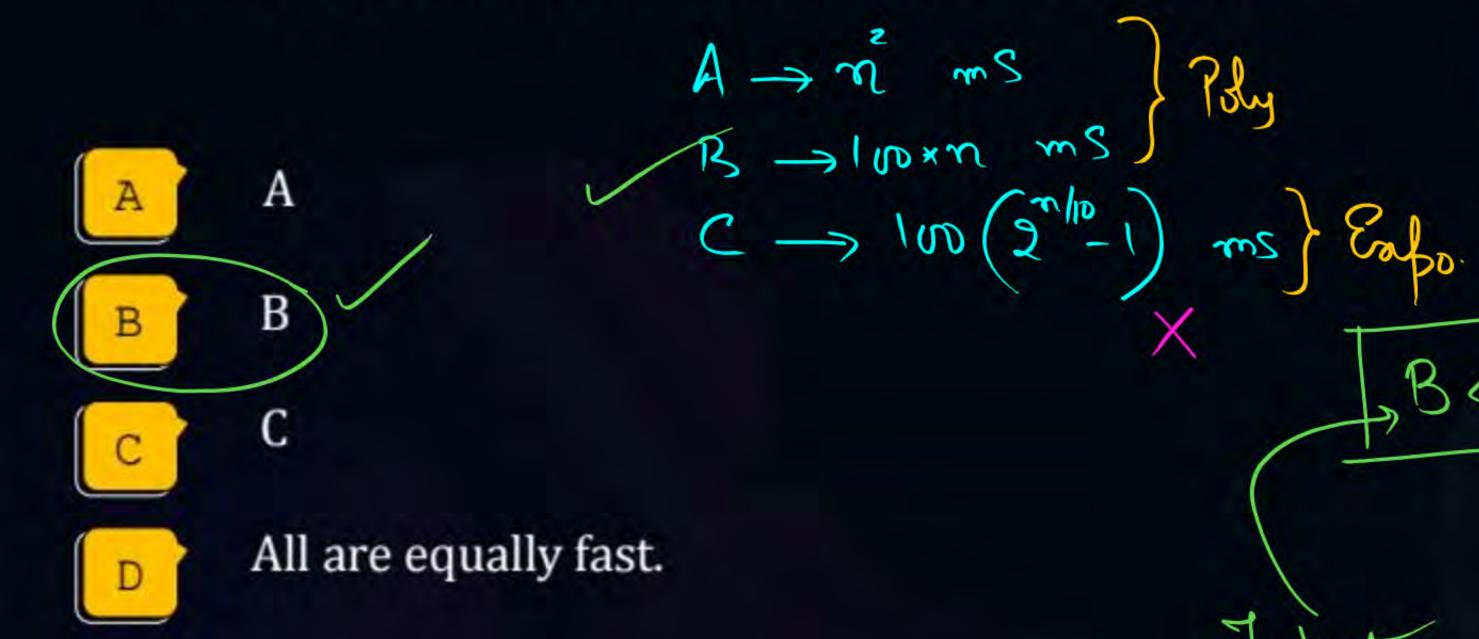
**Divide and Conquer** 

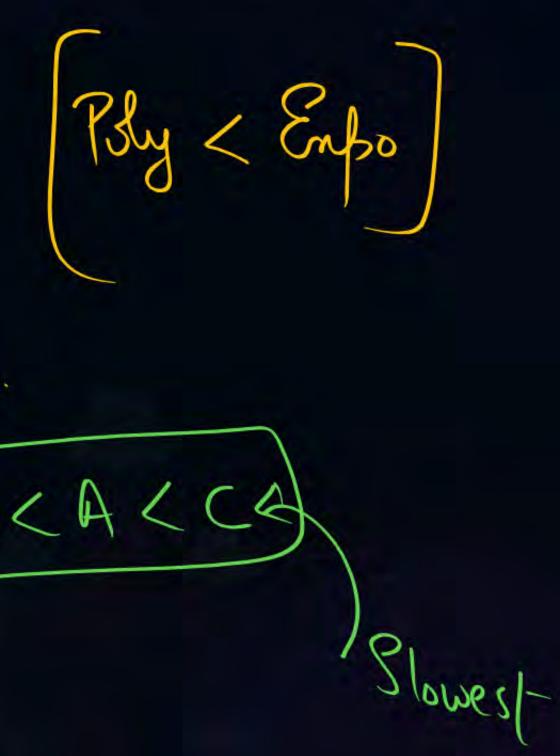
Math Background. 1) Analysis Jime Complenities > Recursive a Non-Recursive Alg's Lesp Complexition) Divide & Conquer Greedy Method Dynamic Programming 5) Graph Techniques 6) Heap Algorithmy 7) Soxting Tech's

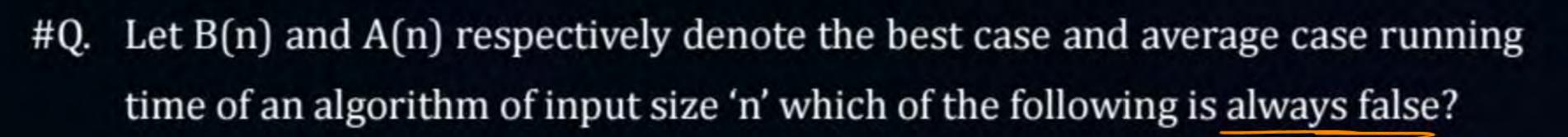




Which algorithm is fastest for all very large value of n?









B(n) = 
$$\Omega(A(n))$$
 May be True

B(n) =  $\theta(A(n))$  May be True

#Q. 
$$f(n) = \sum_{i=1}^{n} i^{\frac{1}{2}}$$
 is\_\_\_.

 $\theta(n)$ 

$$f(n) = \frac{3}{2}i^{2}$$

$$\int_{i=1}^{\infty} i^{2} \sqrt{2} = \left(\frac{3}{2}+1\right)^{n}$$





$$\frac{3/2}{2} \times \frac{2}{3}$$

$$e^{-c}$$
  $\theta(n\sqrt{n})$ 

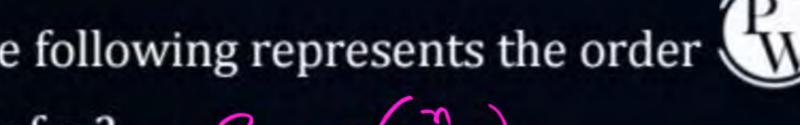
$$\theta(\sqrt{n})$$

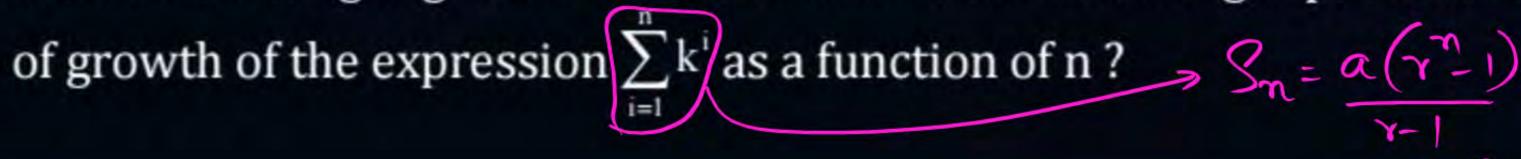
$$f(n) = \frac{2}{5}i^{1/2} = 0$$

$$= \frac{1}{\sqrt{|x|^2}} = \frac{1.5 \times \log n}{\log n}$$

$$= \frac{1}{\sqrt{|x|^2}} = \frac{1.5 \times \log n}{\log n}$$

#Q. Let k be an integer greater than 1. Which of the following represents the order





$$=\frac{1}{\sqrt{(\kappa^{n}-1)}}=O(\kappa^{n})$$

$$\Theta(k^n)$$



$$\Theta(n^{k+1})$$

#### Consider the following functions: #Q.

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

$$g(n) = n \log^{10} n$$

$$g(n) = n \log^{10} n$$

$$g(n) = n \log^{10} n$$



#### which of the following is incorrect?

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

 $h(n) = n^3$ 

$$g(n) = O(h(n)) \quad \Box \quad \Box$$

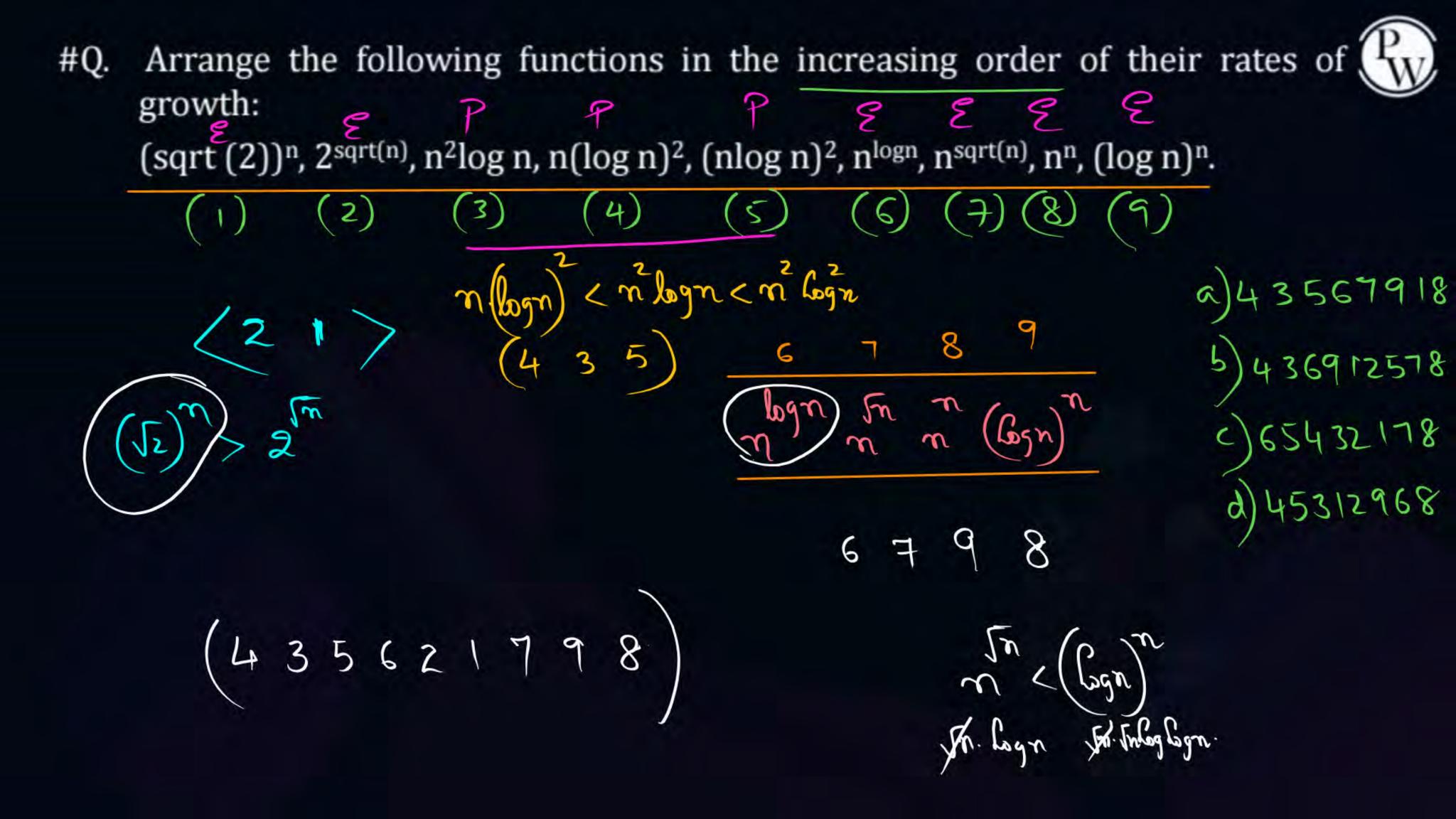
$$f(n) = O(h(n))$$

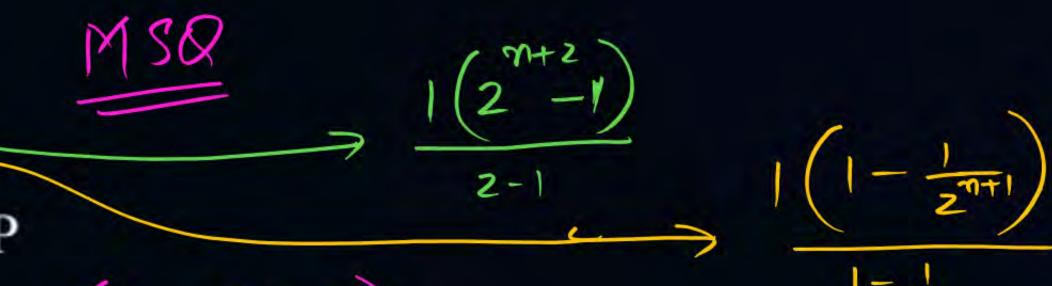
$$\frac{\log_2(n!), \frac{n}{2^n}, \frac{n}{\log_2 n}, (\log_2 n)!, \log_2 n}{\binom{n \log_2 n}{2}}$$

Arrange them in the decreasing order of rates of growth.

$$x! = O(x^{x})$$
 $(\log n)! = (\log n)$ 
 $\log n$ 
 $\log n$ 

$$(2)$$
  $(3)$ 







Which of the following notation asymptotically is / are remains correct, with respect to P?

$$\theta(2^{n+2})$$

$$\left(\frac{2n+2}{2}\right)\left(\frac{1-\frac{1}{2n+1}}{2}\right)$$

$$= 2 - 2 - 1 + \frac{1}{m}$$

$$= \left( \frac{2}{2} - 3 + \frac{1}{2} \right)^{2}$$

$$\theta(2^{2n})$$





```
main()
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         O(n \log n)
                                                                                                                                                                                   for (i = 1; i ≤ n; ++i) : m
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       O(n²logn)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         O(n^2)
                                                                                                                                                                                                                                                                                        2. while (j \le n) for k = 2*j; for k = 1; k \le n; k \ge n; k \ge n; k \le n; k \ge n; 
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         None of the above
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           \mathcal{M} + \mathcal{M} - \mathcal{M} + \mathcal{M} = \mathcal{M} + \mathcal{M} - \mathcal{M} -
```

The time complexity of above algorithm will be?

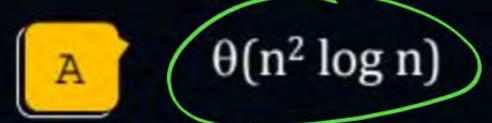
The Keturn (Final) value 9 K is

Pw

#### #Q. Consider the following function:

int unknown (int n)

```
int i, j, k = 0;
for (i = n/2; i < n; i++)
for (j = 2; j \le n; j = j * 2)
return (k);
```

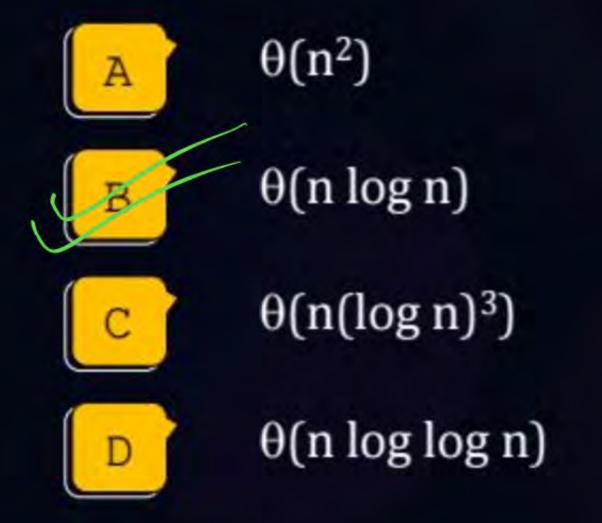


$$\theta(n \log n)$$

$$\theta(n^2)$$

What will be the time complexity of above C code?

What is the running time complexity of the above code?



#Q. If the given program takes 10 milliseconds to run n = 50, how long, in milliseconds, will it take to run for n = 500? \_\_\_.

#Q. Determine the Time Complexity of the following iterative function: int f (int A[SIZE] [SIZE], int n) Jotal = (m-1)+ 2 + (n-3) + 4 + (n-5) + 6 -- + · 1+n Count =  $2 + 4 + 6 + \dots + m + m \times m - (1 + 3 + 5 + \dots + \frac{m-1}{2})$ int i, j, sum = 0; for  $(i = 1; i \le n; ++i)$  $=\frac{m(n+1)}{3}(n+1)+m+--=0(n)$ if (i % 2 ==0) for  $(j = 1; j \le i; j = j + 1)$  sum = sum + A[i] [j]; else for (j = n; j > i; j = j - 1) sum = sum - A[i] [j];

```
#Q. What does the following function compute?

int F (int n, int *Fprev)

{

int Fn_1, Fn_2;

if (n == 0) {

*Fprev = 1;

return (0);
}
```

Recursion \*Fprev = 0; Recursion return (1); Stack }

if (n==1) {

```
Fn_1 = F(n -1, & Fn_2);

*Fprev = Fn_1;

return (Fn_1+Fn_2);
```



```
Return not Fibonacii Number
Jime: O(n)
```

Jime Complexity Recurrence of  

$$F(n) = C, n \le 1$$
  
 $= \alpha + T(n-1) + b$   
 $= T(n-1) + d, m > 1$ 



#### 2 mins Summary



Topic One Analysis of Algorithms

**Divide and Conquer** 

Topic Two

Topic Three

Topic Four

Topic Five



# THANK - YOU