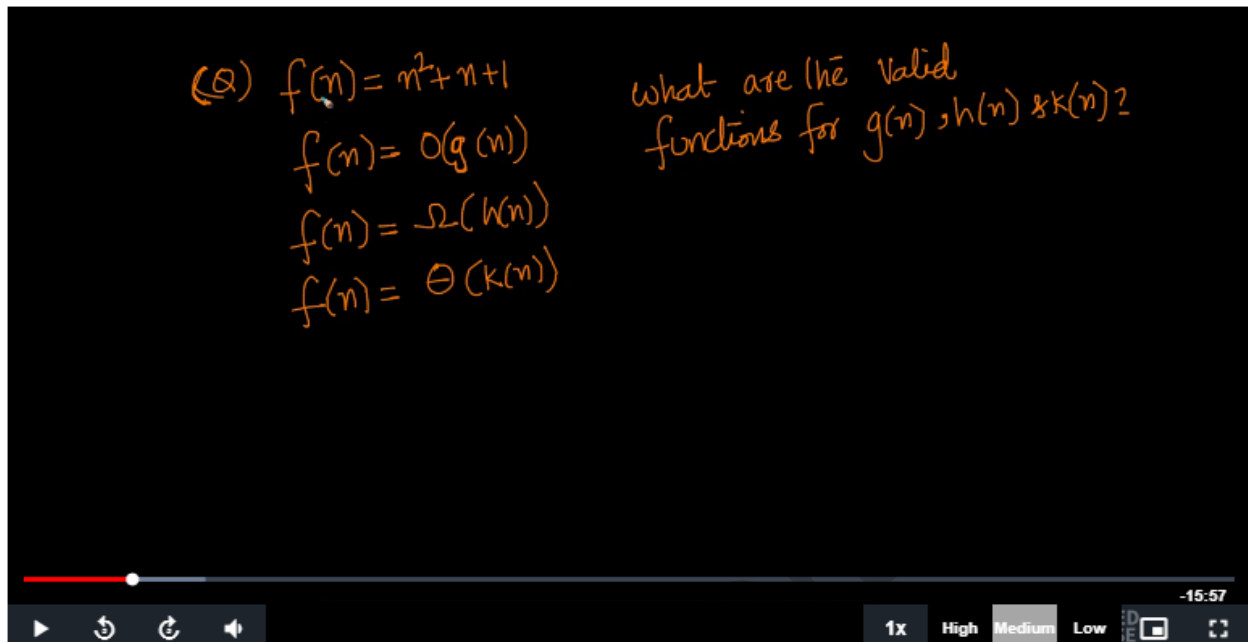


## 10.1 Solved Problem: Polynomials



Timestamp: 1:32

Let us say  $g(n) = n^2$ .

So in order for  $f(n) = O(g(n))$ , there should exist constants  $n_0$ ,  $c$  such that  $0 \leq f(n) \leq c \cdot g(n)$  for all  $n > n_0$ .

We know that  $n^2 + n + 1 \leq n^2 + n^2 + n^2$  (since  $n \leq n^2$  and  $1 \leq n^2$  for  $n \geq 1$ )

Hence  $n^2 + n + 1 \leq 3n^2$ . Hence we are able to find  $n_0 = 1$  and  $c = 3$  that satisfies the above condition. Hence  $f(n) = n^2 + n + 1$  is  $O(n^2)$ .

Similarly we can prove that  $f(n) = n^2 + n + 1$  is also  $O(n^3)$ .

Hence possible values of  $g(n)$  are  $n^2, n^3, n^4$ , etc.

Similarly we can find  $h(n)$  and  $k(n)$  that satisfy  $f(n) = \Omega(h(n))$  and  $f(n) = \Theta(k(n))$  respectively.

But in case of  $\Omega$ , we have to find constants  $c$  and  $n_0$  such that  $0 \leq c \cdot h(n) \leq f(n)$  for all  $n > n_0$ .


In case of  $\Theta$ , it should satisfy both  $O$  and  $\Omega$ .

generalization

$$f(n) = a_0 + a_1 n^1 + a_2 n^2 + a_3 n^3 + \dots + a_m n^m \quad \left( \begin{array}{l} a_m > 0 \\ n \geq n_0 \end{array} \right)$$

$$\left\{ \begin{array}{l} \checkmark f(n) = O(n^m) ; O(n^{m+1}) ; O(n^{m+k}) \quad k \geq 0 \\ \checkmark f(n) = \Omega(n^m) ; \Omega(n^{m-1}) ; \Omega(n^{m-k}) \quad k \geq 0 \\ f(n) = \Theta(n^m) \end{array} \right.$$

Shortcut



Timestamp: 16:00

Please refer to the generalization solutions for  $O$ ,  $\Omega$  and  $\Theta$  when we were given function  $f(n)$  of the above form.

## 10.2 Solved Problem: $n > n_0$ case

(Q)  $f(n) = \begin{cases} n^2 & n \leq 100 \\ n & n > 100 \end{cases}$        $g(n) = \begin{cases} n & n < 1000 \\ n^3 & n > 1000 \end{cases}$

Simple

~~(a)  $f(n) = O(g(n))$~~       ~~(b)  $g(n) = O(f(n))$~~

Defn  $\rightarrow f(n) \leq c \cdot g(n) \quad \forall n \geq n_0 \quad \exists c, n_0 \rightarrow f(n) = O(g(n))$

~~$n \geq n$~~

$n_0 = 1000$

Let  $n \geq 1000$

$f(n) = n$

$g(n) = n^3$

$n \leq c \cdot n^3 \quad \forall n \geq 1000$

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

-0:47

Timestamp: 4:18

As shown in the figure, according to definition of  $O$ ,  $f(n) = O(g(n))$  if and only if  $f(n) \leq c \cdot g(n)$  for all  $n \geq n_0$ . The equation should satisfy for all  $n \geq n_0$ . So as in the above case  $f(n) \leq g(n)$  for all  $n \geq n_0$  when  $c=1$  and  $n_0=1000$ . This fails to hold for  $g(n) \geq f(n)$  when  $n \geq 1000$  hence that is not true.

## 10.3 Solved Problem-1

The screenshot shows a video player with a dark background. At the top, there are browser tabs for 'DS Algo - Google Drive', 'Data Structures & Algorithms', 'Screen Shot 2019-02-12', and 'Big O notation - Wikipedia'. The main content area displays a problem statement and four multiple-choice options. Handwritten orange and blue annotations are present throughout the image.

Let  $f(n) = n^2 \log n$  and  $g(n) = n(\log n)^{10}$  be two positive functions of  $n$ . Which of the following statements is correct?

(a)  $f(n) = O(g(n))$  and  $g(n) \neq O(f(n))$       (b)  $g(n) = O(f(n))$  and  $f(n) \neq O(g(n))$   
(c)  $f(n) \neq O(g(n))$  and  $g(n) \neq O(f(n))$       (d)  $f(n) = O(g(n))$  and  $g(n) = O(f(n))$

Handwritten notes include:  
-  $f(n) \rightarrow n^2 \log n$   
-  $n(\log n)^9$   
-  $n$  and  $(\log n)^9$  circled and connected by arrows.  
-  $n \text{ vs } (\log n)^9$   
-  $f(n)$  and  $g(n)$  underlined.  
-  $\text{order of functions}$  written in orange.

Timestamp 2:26

As shown in the above figure, to compare  $f(n)$  and  $g(n)$

$$f(n) = n^2 \log n \text{ and } g(n) = n(\log n)^{10}$$

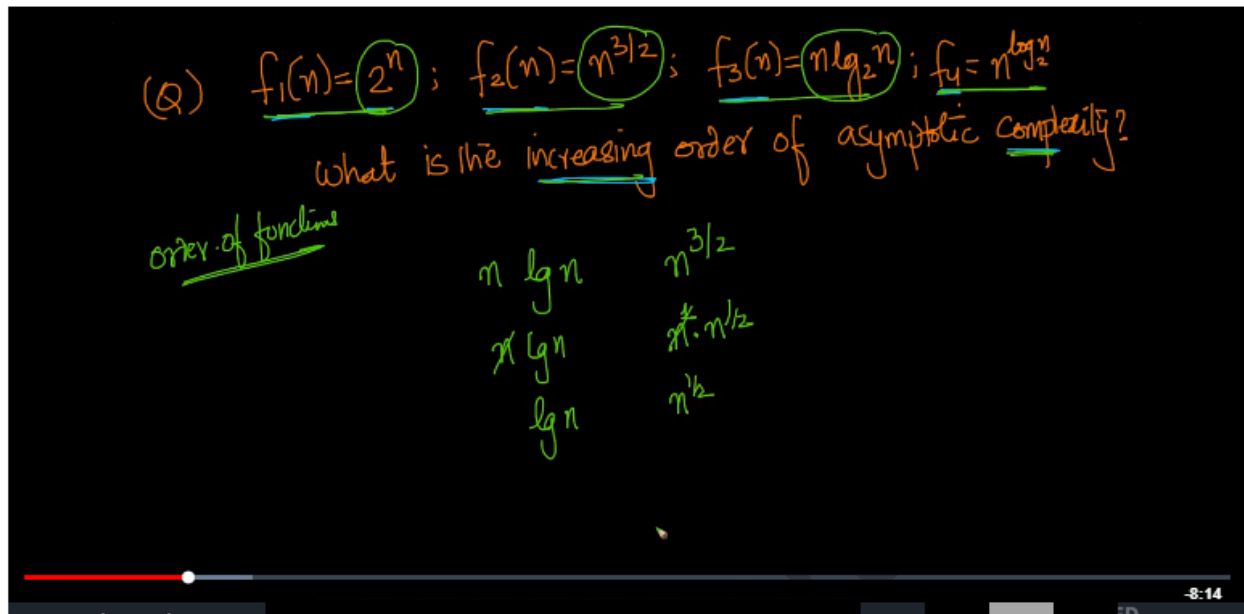
$$\rightarrow f(n) = n \log n * n \text{ and } g(n) = n \log n * (\log n)^9$$

$$\rightarrow n \text{ and } (\log n)^9 \text{ (} n \log n \text{ cancel on both sides)}$$

$$\rightarrow \text{from the reference table we know that } n > (\log n)^9$$

Hence  $f(n) > g(n)$ . Hence  $g(n) = O(f(n))$  and  $f(n) \neq O(g(n))$

## 10.4 Solved Problem-2



Timestamp: 1:13

As shown in the above figure when to compare

$$f_2(n) = n^{3/2} \text{ and } f_3(n) = n \lg n$$

$$\rightarrow n \cdot n^{1/2} \text{ and } n \cdot \lg n$$

$$\rightarrow n^{1/2} \text{ and } \lg n$$

$\rightarrow$  from the reference

[https://en.wikipedia.org/wiki/Big\\_O\\_notation#Orders\\_of\\_common\\_functions](https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions)

we know  $n^{1/2} \geq \lg n$

$$\text{Hence } f_2(n) > f_3(n)$$

Similarly, coming to  $f_4$  and  $f_2$ .

$$f_4(n) = n^{\lg n} \text{ and } f_2(n) = n^{3/2}$$

We know that  $n^a > n^b$  if  $a > b$

Hence we compare powers

$$\rightarrow \lg n \text{ and } 3/2$$

We know that since  $3/2$  is a constant it doesn't increase with  $n$  but  $\lg n$  increases

$$\text{Hence } \lg n > 3/2 \rightarrow f_4(n) > f_2(n).$$

Until now we got  $f_4(n) > f_2(n) > f_3(n)$ . Now let us compare  $f_4(n)$  and  $f_1(n)$ .

$$f_4(n) = n^{\log n} \text{ and } f_1(n) = 2^n$$

Taking log on both of them since we know  $\log a > \log b$  if  $a > b$

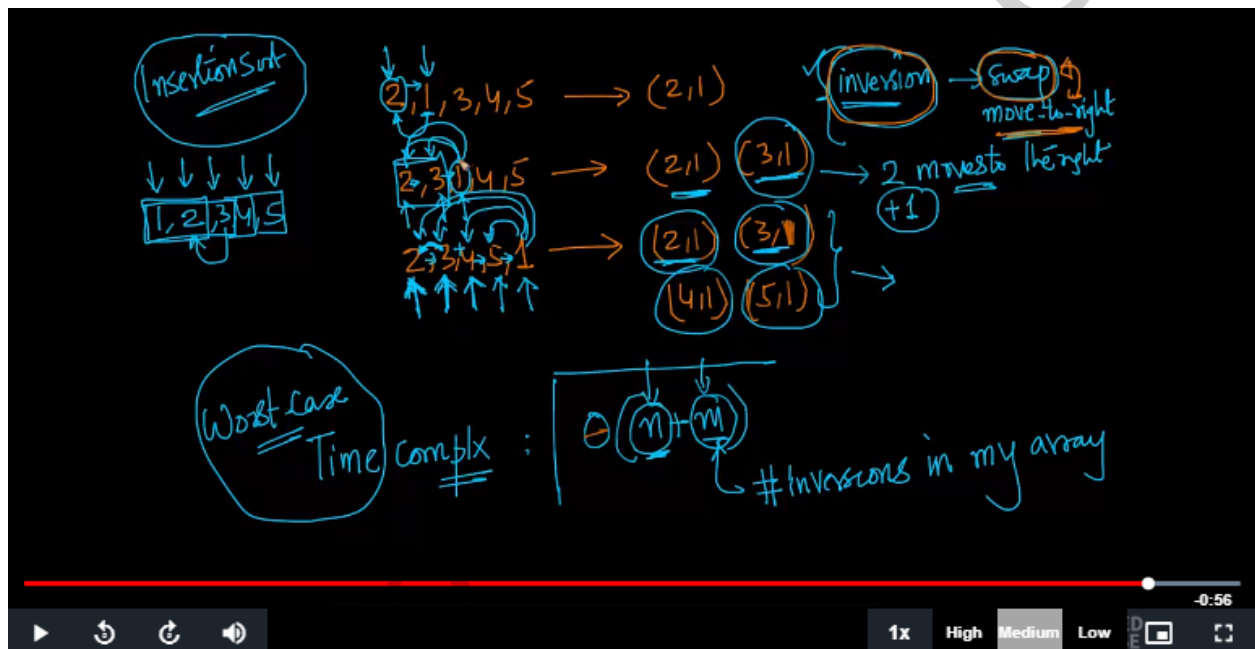
->  $\log n * \log n$  and  $n$

->  $(\log n)^2$  and  $n$

Again from our reference we know that  $n > (\log n)^2$ . Hence  $f_1(n) > f_4(n)$ .

So the required order is  $f_1(n) > f_4(n) > f_2(n) > f_3(n)$

## 10.5 Solved Problem-3



Timestamp: 11:18

For any pair of indices  $i, j$  in an array  $A$  if  $i < j$  and  $A[i] > A[j]$ , then it is called an inversion.

In insertion sort as shown in the above figure the number of inversions = no of swaps that need to be made

Hence for insertion sort the time complexity can also be thought of as  $O(n+m)$  where  $m$  is the number of inversions and  $n$  is the size of the input array.

In a permutation  $a_1 \dots a_n$ , of  $n$  distinct integers, an inversion is a pair  $(a_i, a_j)$  such that  $i < j$  and  $a_i > a_j$ .

What would be the worst case time complexity of the Insertion Sort algorithm, if the inputs are restricted to permutations of  $1 \dots n$  with at most  $n$  inversions?

A.  $\Theta(n^2)$   
 B.  $\Theta(n \log n)$   
 C.  $\Theta(n^{1.5})$   
 D.  $\Theta(n)$

$\Theta(n+m)$   $m=n$   
 $\Theta(n+n) = \Theta(2n) = \Theta(n)$

Timestamp: 10:58

Since the number of inversions given is atmost  $n$ , then the time complexity becomes  $\Theta(n+n)=\Theta(2n)$

## 10.6 Solved Problem-4

1.  $(n+k)^m = O(n^m)$  ✓  $k \& m$  are constants

2.  $2^{n+1} = O(2^n)$

3.  $2^{2n+1} = O(2^n)$

which of them are true

1.  $(n+k)^m = n^m + \binom{m}{1} n^{m-1} k + \binom{m}{2} n^{m-2} k^2 + \dots + k^m$

$O(n^m)$   $\Theta(n^m)$   $\Omega(n^m)$

Timestamp: 2:39

For problem 1, we know that a polynomial as shown in the above figure has a time complexity of  $O(n^m)$ .

For problem 2, for  $2^{n+1}$  to be  $O(2^n)$ ,  $2^{n+1} \leq c \cdot 2^n$  for some  $c$ , since  $2^{n+1}$  can be written as  $2 \cdot 2^n$ , the equation satisfies for constant  $c=2$ . Hence  $2^{n+1} = O(2^n)$ .

For problem 3,

Let us assume  $2^{2n+1} = O(2^n)$  then the below equation should hold true

$$2^{2n+1} \leq c \cdot 2^n$$

$$\rightarrow 2 \cdot 2^{2n} \leq c \cdot 2^n$$

$$\rightarrow 2^{2n} \leq 2^n \text{ (since constants doesn't matter)}$$

$$\rightarrow 2n \leq n \text{ (since } \log a \leq \log b \text{ iff } a \leq b)$$

Since  $2n \leq n$  is never true for  $n \geq 1$ , our assumption about  $2^{2n+1} = O(2^n)$  cannot be true, hence the answer for problem 3 is false.

## 10.7 Solved Problem-5

Consider the following functions:

- $f(n) = 2^n$
- $g(n) = n!$
- $h(n) = n^{\log n}$

Which of the following statements about the asymptotic behavior of  $f(n)$ ,  $g(n)$  and  $h(n)$  is true?

☒ A.  $f(n) = O(g(n))$ ;  $g(n) = O(h(n))$

☐ B.  $f(n) = \Omega(g(n))$ ;  $g(n) = O(h(n))$

☐ C.  $g(n) = O(f(n))$ ;  $h(n) = O(f(n))$

☒ D.  $h(n) = O(f(n))$ ;  $g(n) = \Omega(f(n))$

Handwritten notes and diagrams:

- $\log(ab) = b \log a$
- $\log_2(2^n) = n$
- $n^{\log n} < 2^n < n!$  (with a checkmark)
- $O, \Omega$
- Comparison diagram showing  $n^{\log n} < 2^n$  and  $2^n < n!$ .
- Diagram showing  $(\log n)(\log n) < n$  and  $(\log n)^2 < n$ .

Timestamp: 4:30



We know from the previous proofs and our reference table  $n^{\log n} < 2^n < n!$ .

$n^{\log n} < 2^n$  can be proved using  $n^{\log n} < 2^n \rightarrow \log n < n$ .

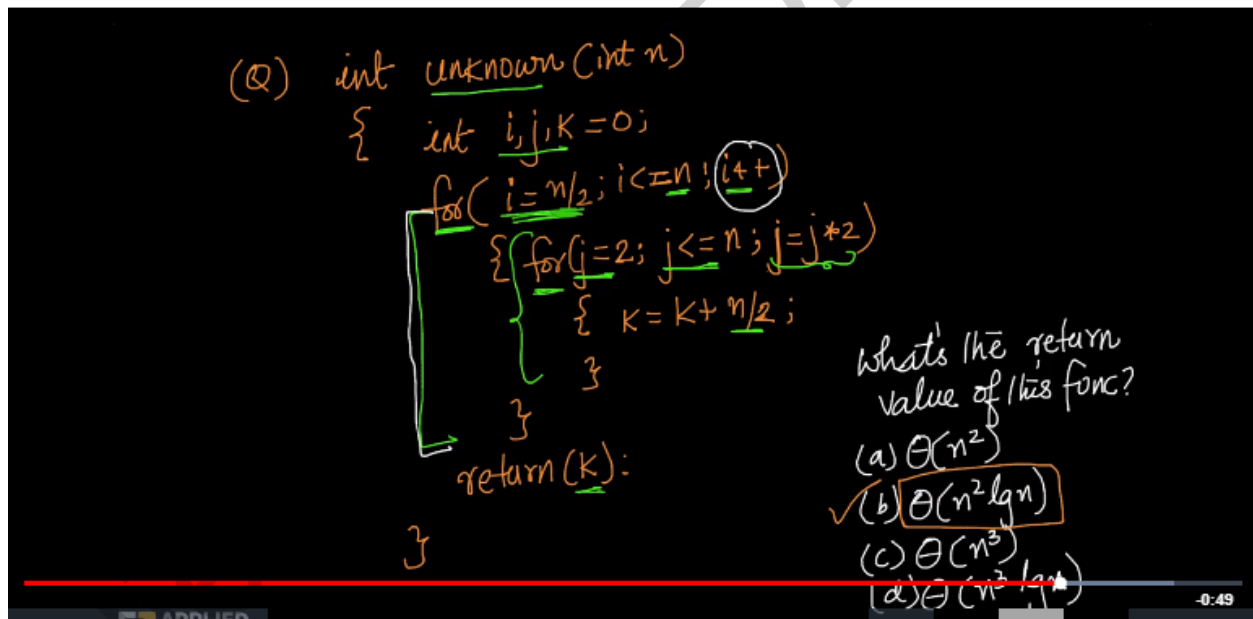
$2^n < n!$  can be checked by running through few values of  $n$ .

Hence in the question above

- a)  $f(n) = O(g(n))$  is true and  $g(n) = O(h(n))$  is false.
- b)  $f(n) = \Omega(g(n))$  is false and  $g(n) = O(h(n))$  is false.
- c)  $g(n) = O(f(n))$  is false and  $h(n) = O(f(n))$  is true.
- d)  $h(n) = O(f(n))$  is true and  $g(n) = \Omega(f(n))$  is true.

Hence the answer is d.

## 10.8 Solved Problem-6



Timestamp: 4:41

The outer for loop with iterator  $i$  runs for  $n - n/2 = n/2$  times. While the inner loop runs from  $j=2$  to  $n$  with increments of 2 times so  $j$  takes values  $2, 4, 8, \dots, n$ , so  $j$  takes  $\log n$  values hence inner loop repeats for  $\log n$  times, each time  $k$  is incremented by  $n/2$ .

Hence at the end the value of  $K = n/2 * \log n * n/2$  which is  $\Theta(n^2 \log n)$ .

## 10.9 Solved Problem-7

The screenshot shows a video player with a dark background. At the top, there are browser tabs for 'OS Algo - Google Drive', 'Data Structures & Algorithms', 'Big O notation - Wikipedia', and 'Geometric series - Wikipedia'. The main content area displays a C-program fragment with handwritten annotations in orange:

```

for( i = n, j = 0; i > 0; i /= 2, j += i );
    
```

Handwritten notes above the code show the progression of variables:

$$\begin{array}{l}
 i=n \quad j=0 \\
 i=n/2 \quad j=0+n/2 \\
 i=n/4 \quad j=0+n/2+n/4 \\
 \dots \\
 i=1 \quad j=0+\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\dots+1
 \end{array}$$

Below the code, a text box asks: "Consider the following C-program fragment in which  $i$ ,  $j$  and  $n$  are integer variables. Let  $val(j)$  denote the value stored in the variable  $j$  after termination of the for loop. Which one of the following is true?"

Options A, B, C, and D are listed, with handwritten notes next to them:

- A.  $val(j) = \Theta(\log n)$
- B.  $val(j) = \Theta(\sqrt{n})$
- C.  $val(j) = \Theta(n)$
- D.  $val(j) = \Theta(n \log n)$

Handwritten notes include:

- $i = i/2;$
- $j = j + i;$
- $1 + \frac{1}{2} + \frac{1}{4} + \dots$
- $val(j) = n \left\{ \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{n} \right\}$
- Annotations for the series terms:  $1^{st}, 1^{st}, 1^{st}, 1^{st}, 1^{st}$

The video player controls at the bottom show a progress bar at 2:28, with buttons for 1x, High, Medium, Low, and a full screen icon.

Timestamp: 4:10

As shown in the above figure, when  $i=n$   $j=0$  when  $i=n/2$   $j=0+n/2$  when  $i=n/4$   $j=0+n/2+n/4$  similarly when  $i=n$   $j=0+n/2+n/4+\dots+1$ .

->  $j = n(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots) = n(1 - \frac{1}{2} + \frac{1}{2} - \frac{1}{4} + \frac{1}{4} - \frac{1}{8} + \dots)$

->  $j <= n(2-1)$  (due to geometric series  $1 + \frac{1}{2} + \frac{1}{4} + \dots = 1/(1-\frac{1}{2})$  but since our series is finite, we are using  $<$ )

Hence  $val(j) = \Theta(n)$