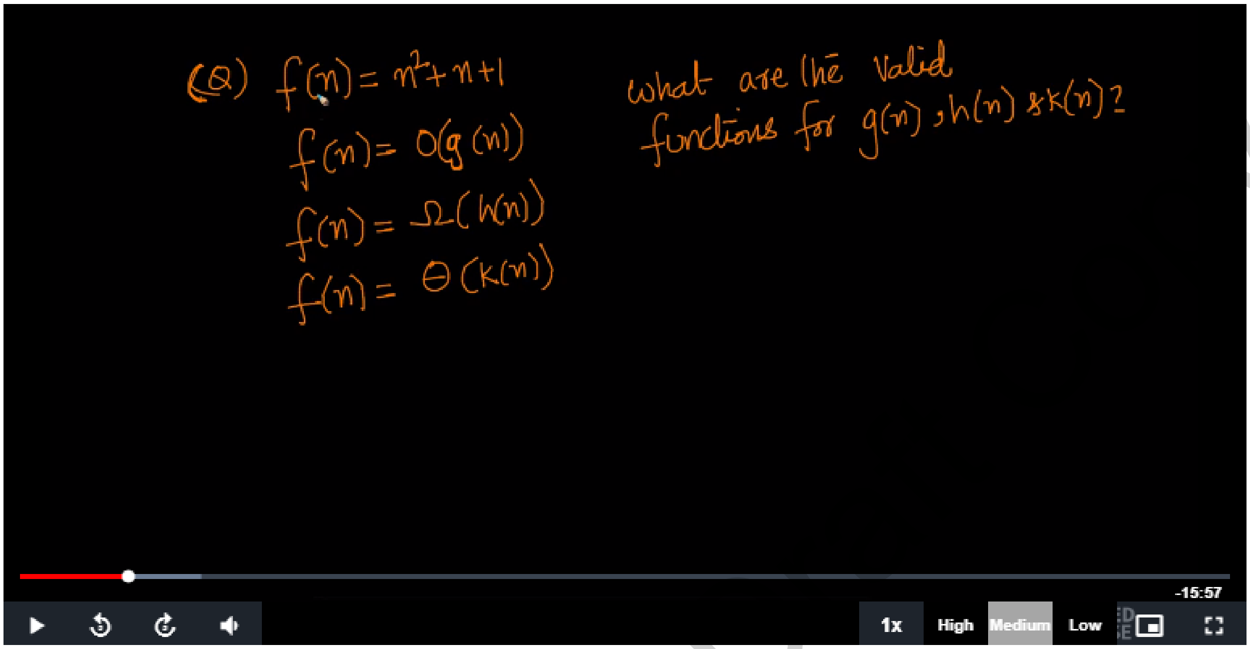
10.1 Solved Problem: Polynomials



Timestamp: 1:32

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

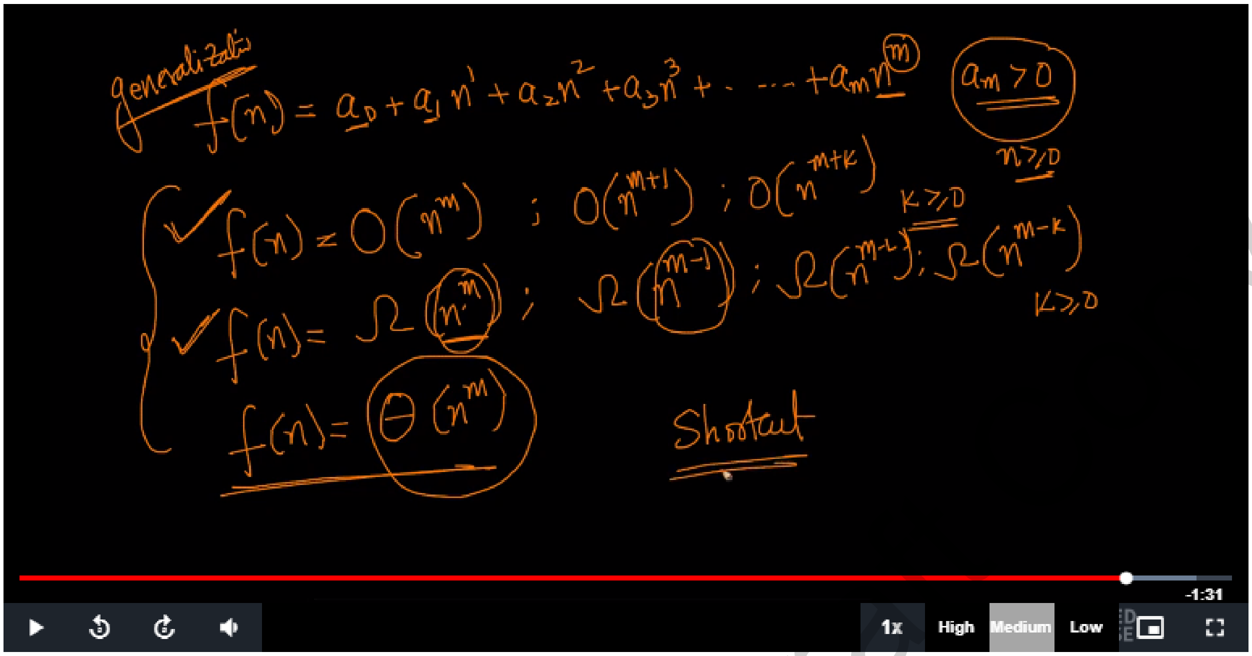
So in order for $f(n) = O(g(n))$, there should exists constants n_o, c such that $0<=f(n)<=c*g(n)$ for all $n>n_0$.

We know that $n^2 +n+1 <= n^2 +n^2 +n^2$ (since $n<=n^2$ and $1<=n^2$ for $n>=1$)
Hence $n^2+n+1 <= 3n^2$. Hence we are able to find $n_o=1$ 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 n2, n3,n4,etc.

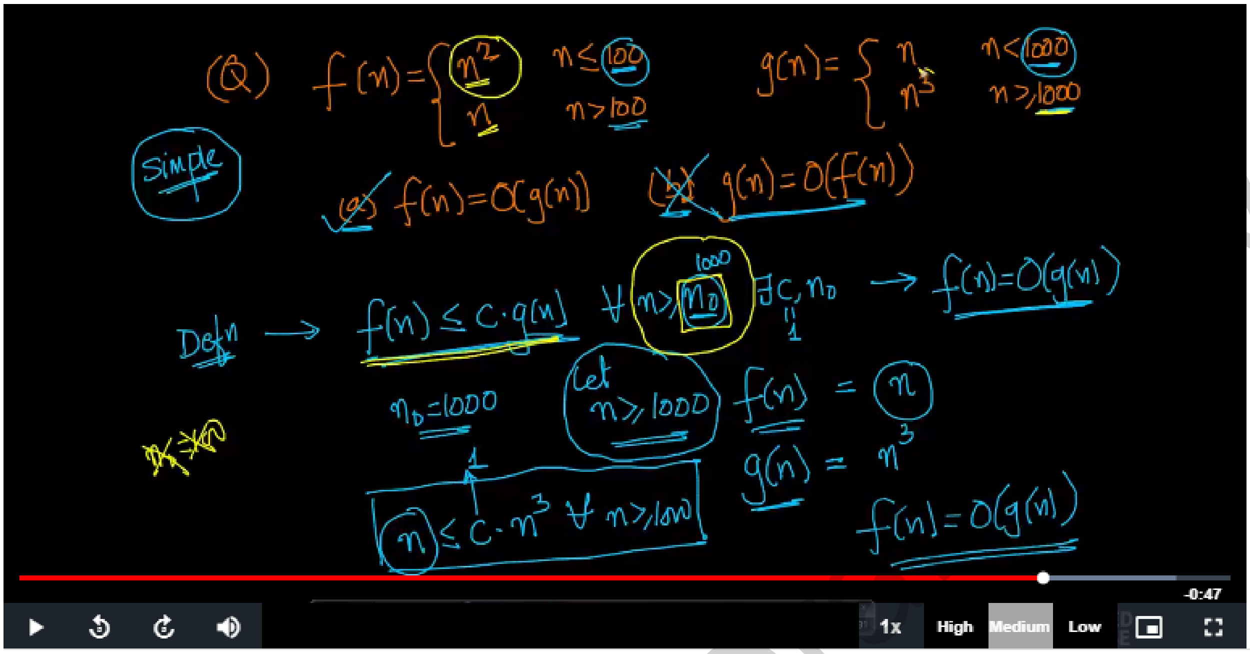
Similarly we can find h(n) and k(n) that satisfy $f(n)=\Omega(h(n))$ and $f(n)=\Theta(k(n))$ respectively.
But in case of Ω, we have to find constants c and n_0 such that $0<=c*h(n)<= f(n)$ for all $n>n_0$.
In case of Θ, it should satisfy both O and Ω.



Timestamp: 16:00

Please refer to the generalization solutions for O, Ω and Θ when we were given function f(n) of the above form.

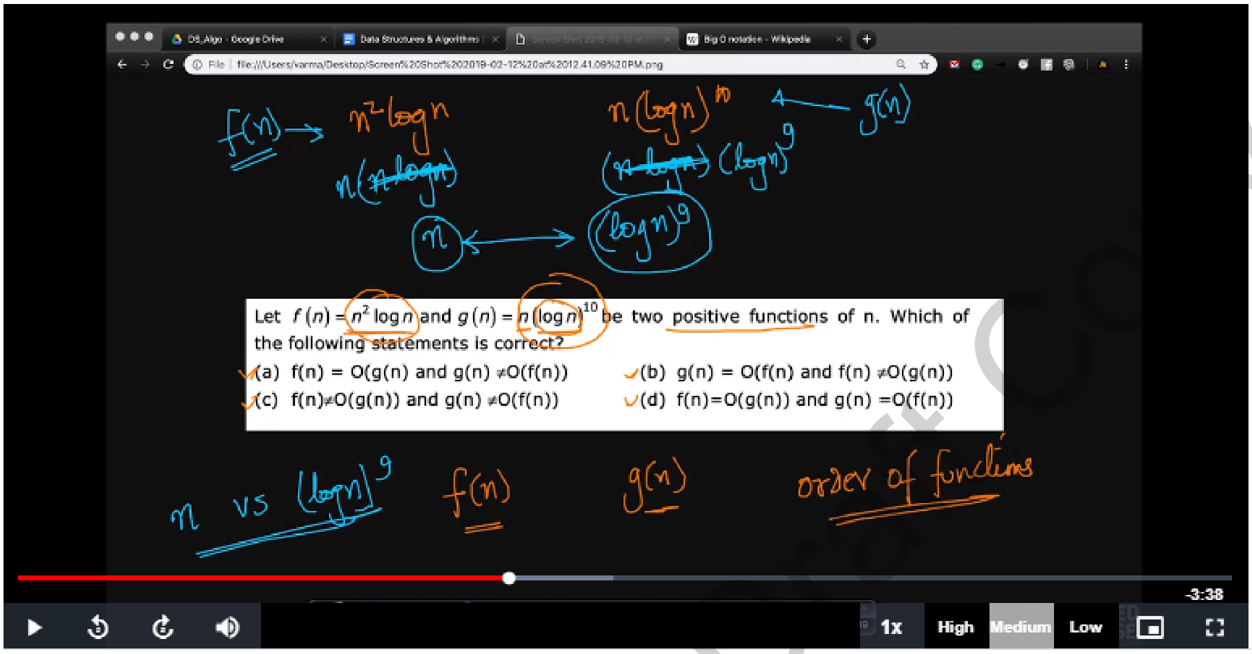
10.2 Solved Problem: n>n0 case



TImestamp: 4:18

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

10.3 Solved Problem-1



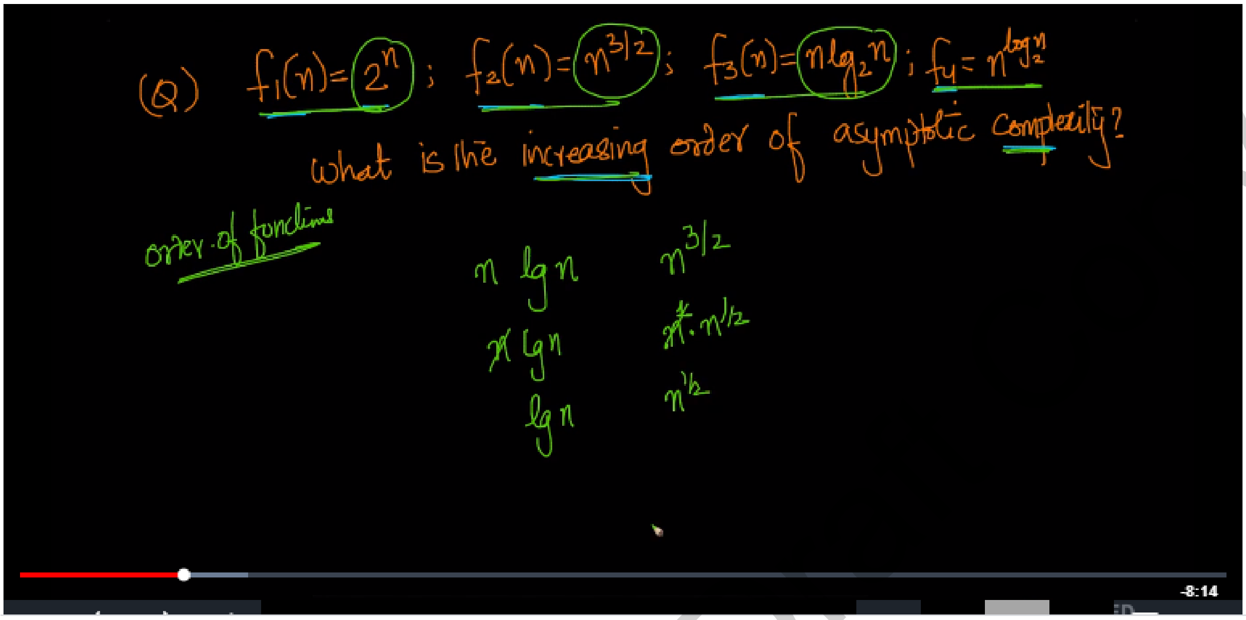
Timestamp 2:26

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

f(n)=n^2logn and g(n) = n$(\log n)^{10}$
-> f(n)=nlogn * n and g(n)= nlogn *$(\log n)^9$
-> n and $(\log n)^9$ (nlogn cancel on both sides)
-> 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)!=O(g(n))

10.4 Solved Problem-2



Timestamp: 1:13

As shown in the above figure when to compare

$f2(n) = n^{3/2}$ and $f3(n)=nlogn$
-> $n*n^{1/2}$ and n*logn
-> $n^{1/2}$ and logn
-> from the reference
https://en.wikipedia.org/wiki/Big_O_notation#Orders_of_common_functions
we know $n^{1/2}$ >=logn

Hence f2(n)>f3(n)

Similarly, coming to f4 and f2.

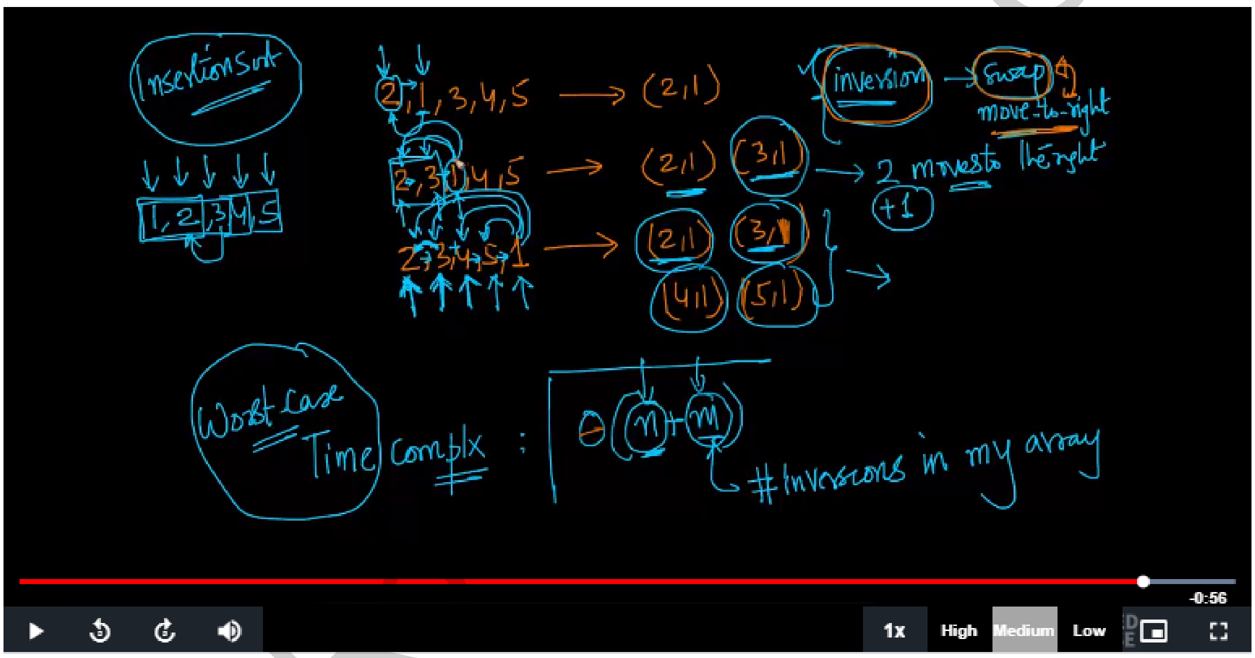
$f4(n) = n^{logn}$ and $f2(n) = n^{3/2}$
We know that $n^a > n^b$ if a>b
Hence we compare powers
-> logn and 3/2
We know that since 3/2 is a constant it doesn't increase with n but logn increases
Hence logn >3/2 -> f4(n) > f2(n).

Until now we got f4(n)>f2(n)>f3(n). Now let us compare f4(n) and f1(n).

$f4(n)=n^{\log n}$ and $f1(n) = 2^n$
Taking log on both of them since we know log a >log b if a>b
-> logn * logn and n
->$(\log n)^2$ and n
Again from our reference we know that $n>(\log n)^2$. Hence f1(n)>f4(n).

So the required order is f1(n)>f4(n)>f2(n)>f3(n)

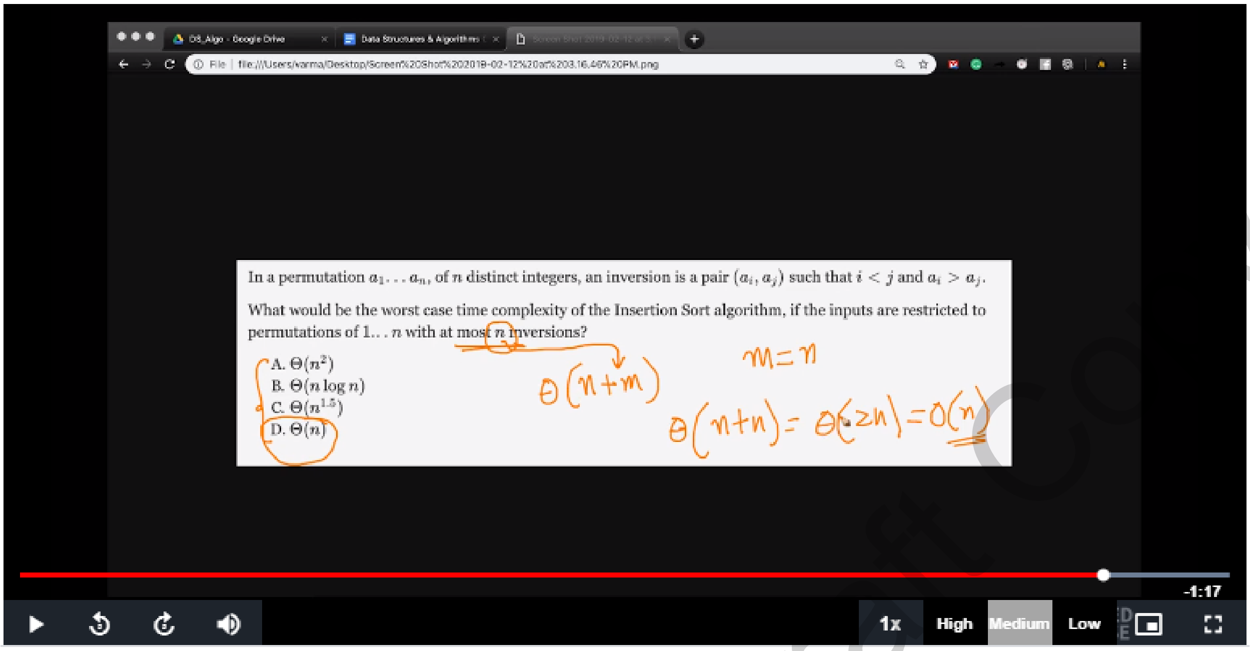
10.5 Solved Problem-3



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

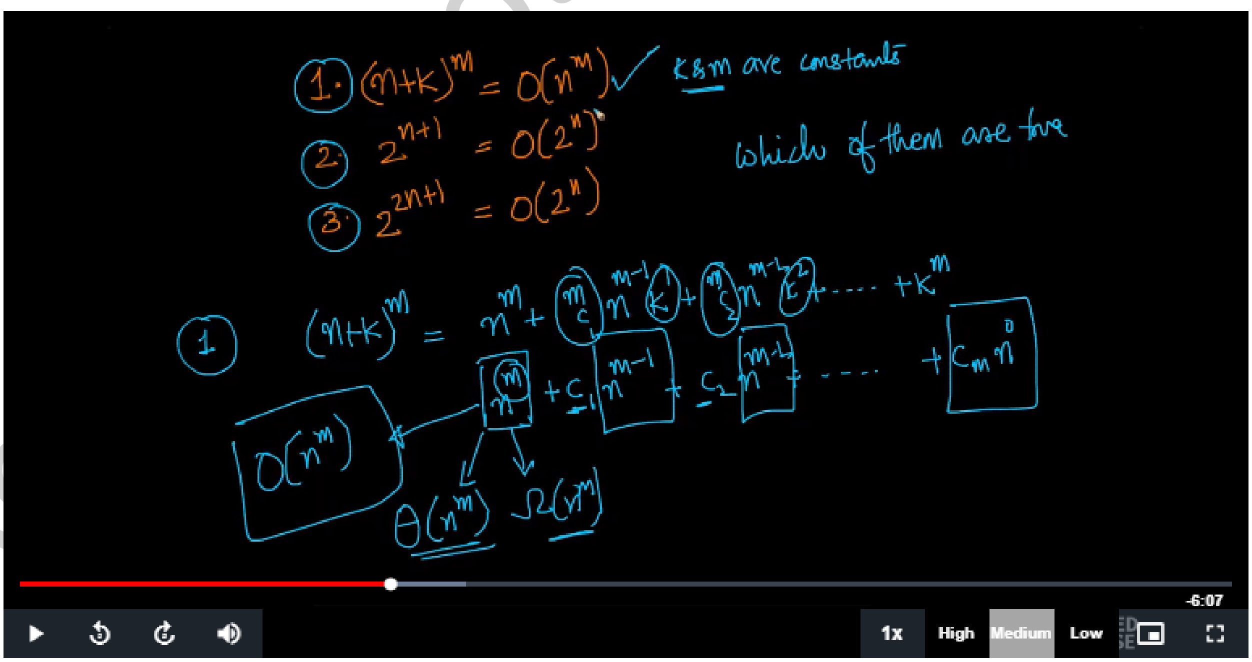
Hence for insertion sort the time complexity can also be thought of as Θ(n+m) where m is the number of inversions and n is the size of the input array.



Timestamp: 10:58

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

10.6 Solved Problem-4



Timestamp: 2:39

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

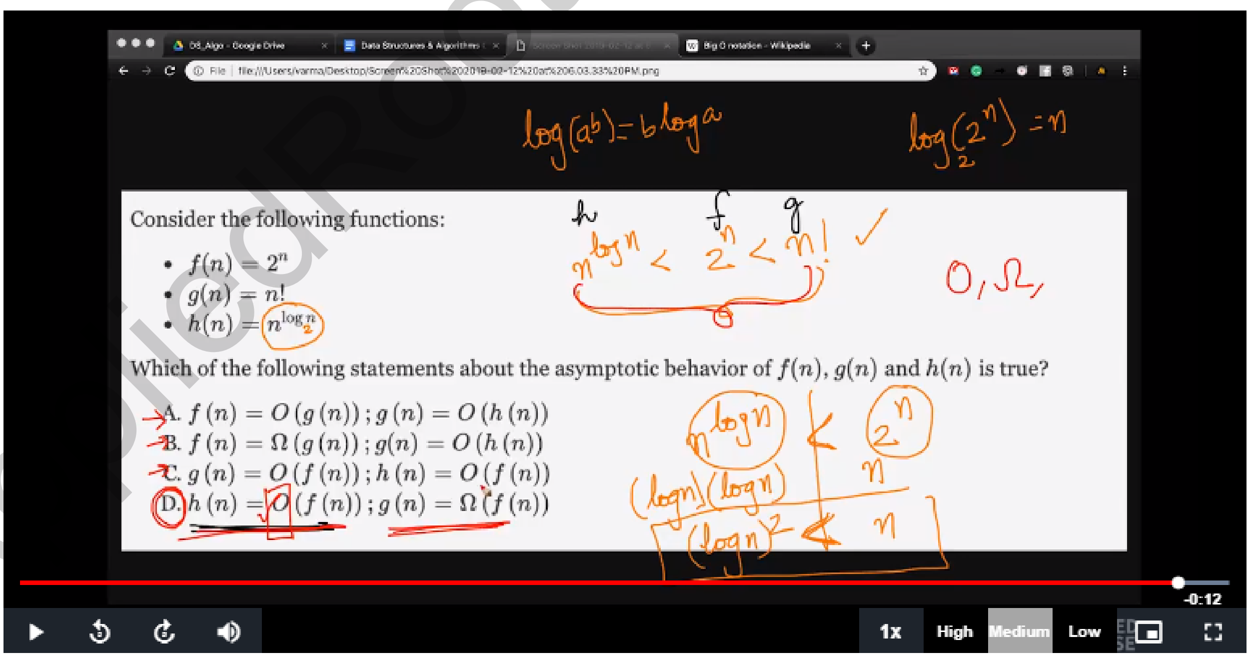
For problem 2, for 2^{n+1} to be $O(2^n)$, $2^{n+1} <= c*2^n$ for some c, since 2^{n+1} can be written as $2*2^n$, the equation satisfies for constant c=2. Hence $2^{n+1}=O(2^n)$.

For problem 3,

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

$2^{2n+1} <= c*2^n$
-> $2*2^{2n} <= c*2^n$
-> $2^{2n} <= 2^n$ (since constants doesn't matter)
-> $2n <= n$ (since log a<=logb iff a<=b)
Since $2n<=n$ is never true for $n>=1$, our assumption about $2^{2n+1}=O(2^n)$ cannot be true, hence the answer for problem3 is false.

10.7 Solved Problem-5



Timestamp: 4:30

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

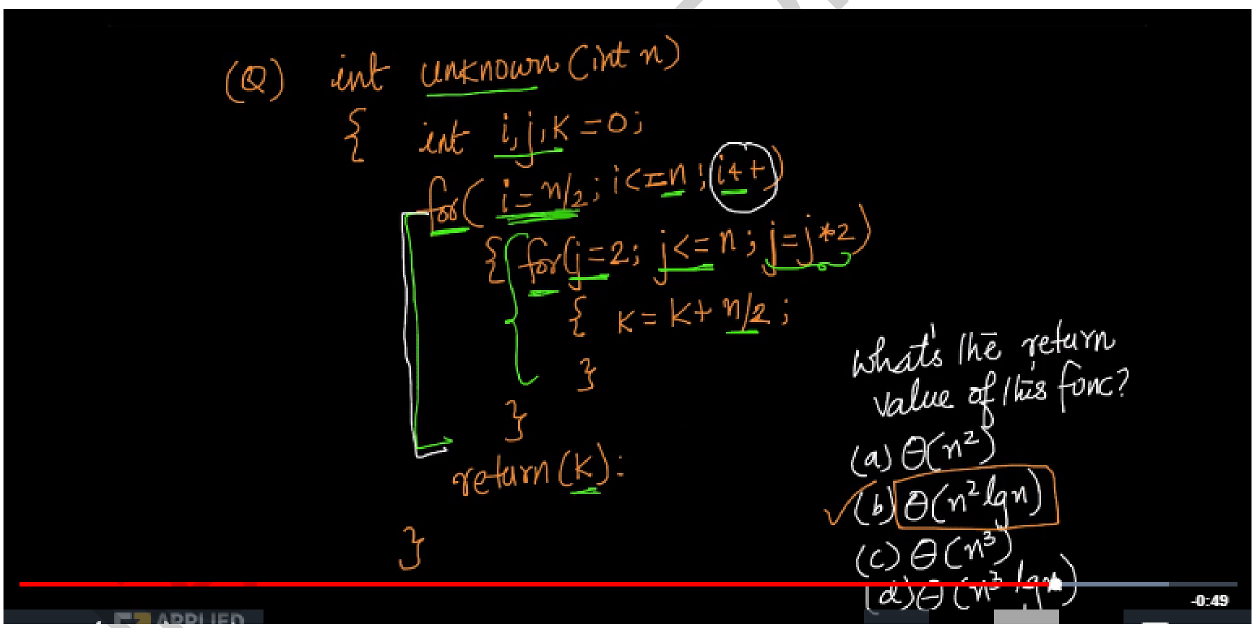
$n^{logn} < 2^n$ can be proved using $n^{logn} < 2^n$ -> $logn < n$.
$2^n < n!$ can be checked by running through few values of n.

Hence in the question above

a) f(n)=O(g(n)) is true and g(n)=O(h(n)) is false.
b) f(n)=Ω(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) = Ω(f(n)) is true.

Hence the answer is d.

10.8 Solved Problem-6

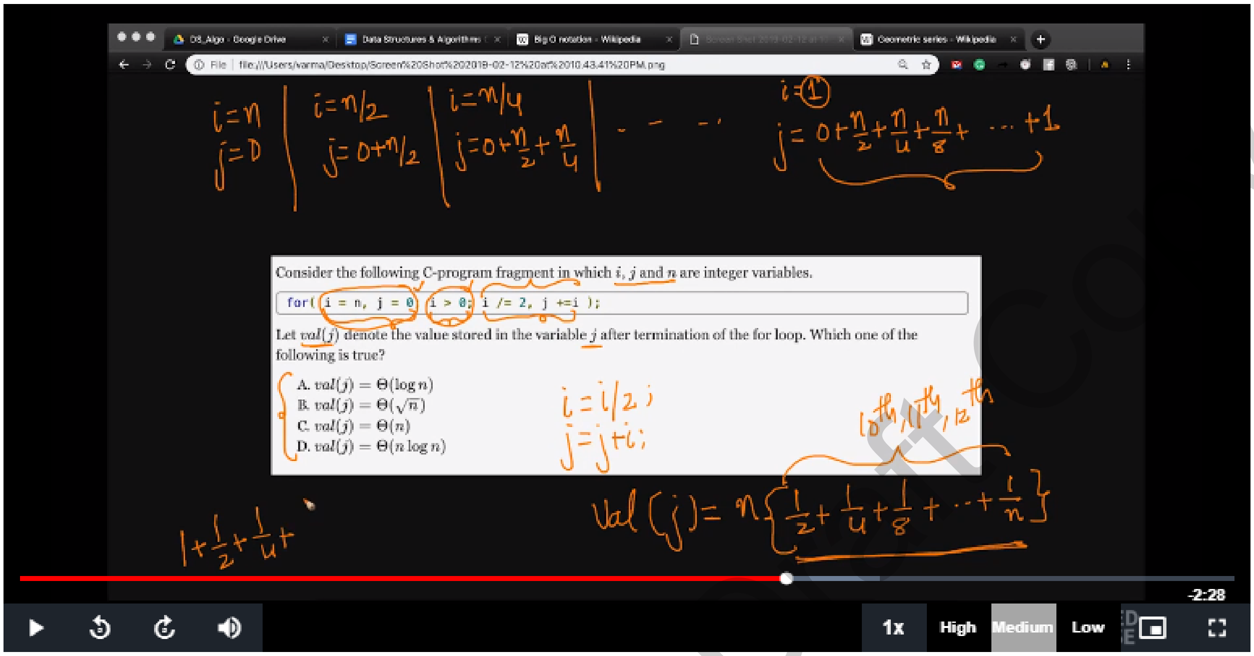


Timestamp: 4:41

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

Hence at the end the value of K=n/2*logn*n/2 which is $Θ(n^2logn)$.

10.9 Solved Problem-7



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+....1.

-> $j= n(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+..) = n (1-1 +\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+..)$
-> $j<=n(2-1)$ (due to geometric series $1+\frac{1}{2}+\frac{1}{4}+ \ldots =1/(1-\frac{1}{2})$ but since our series is finite, we are using <)
Hence val(j) =Θ(n)