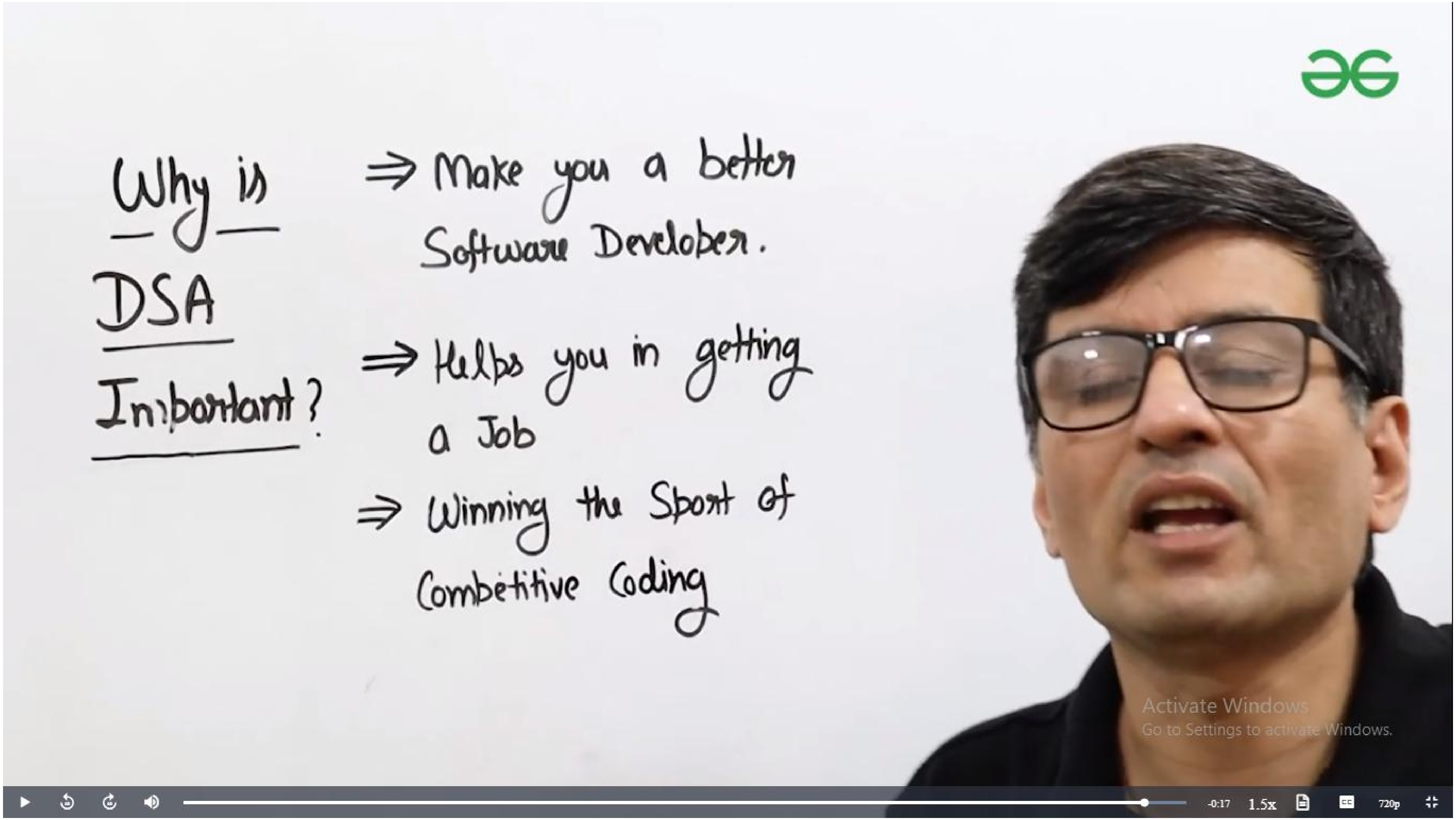
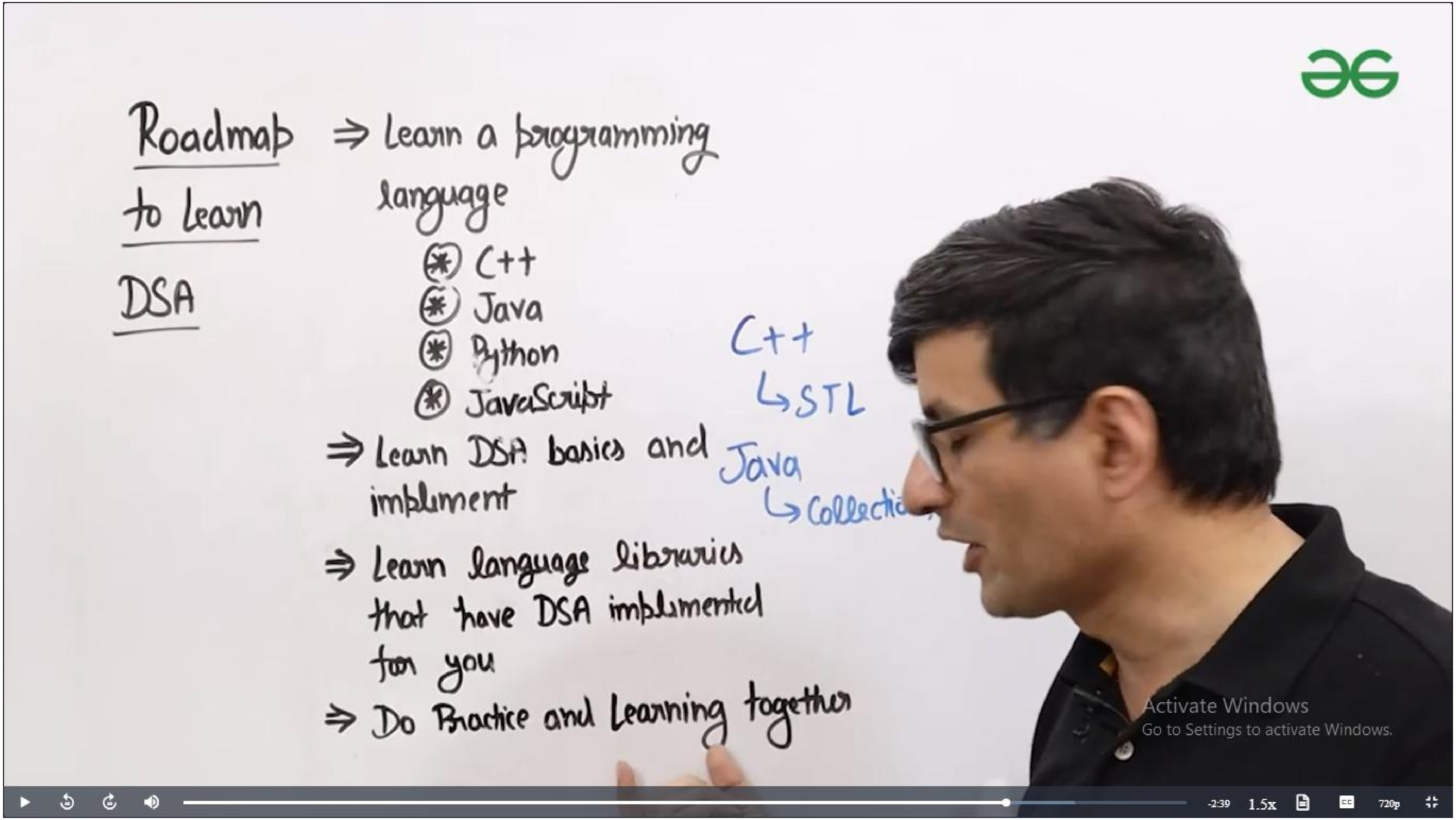
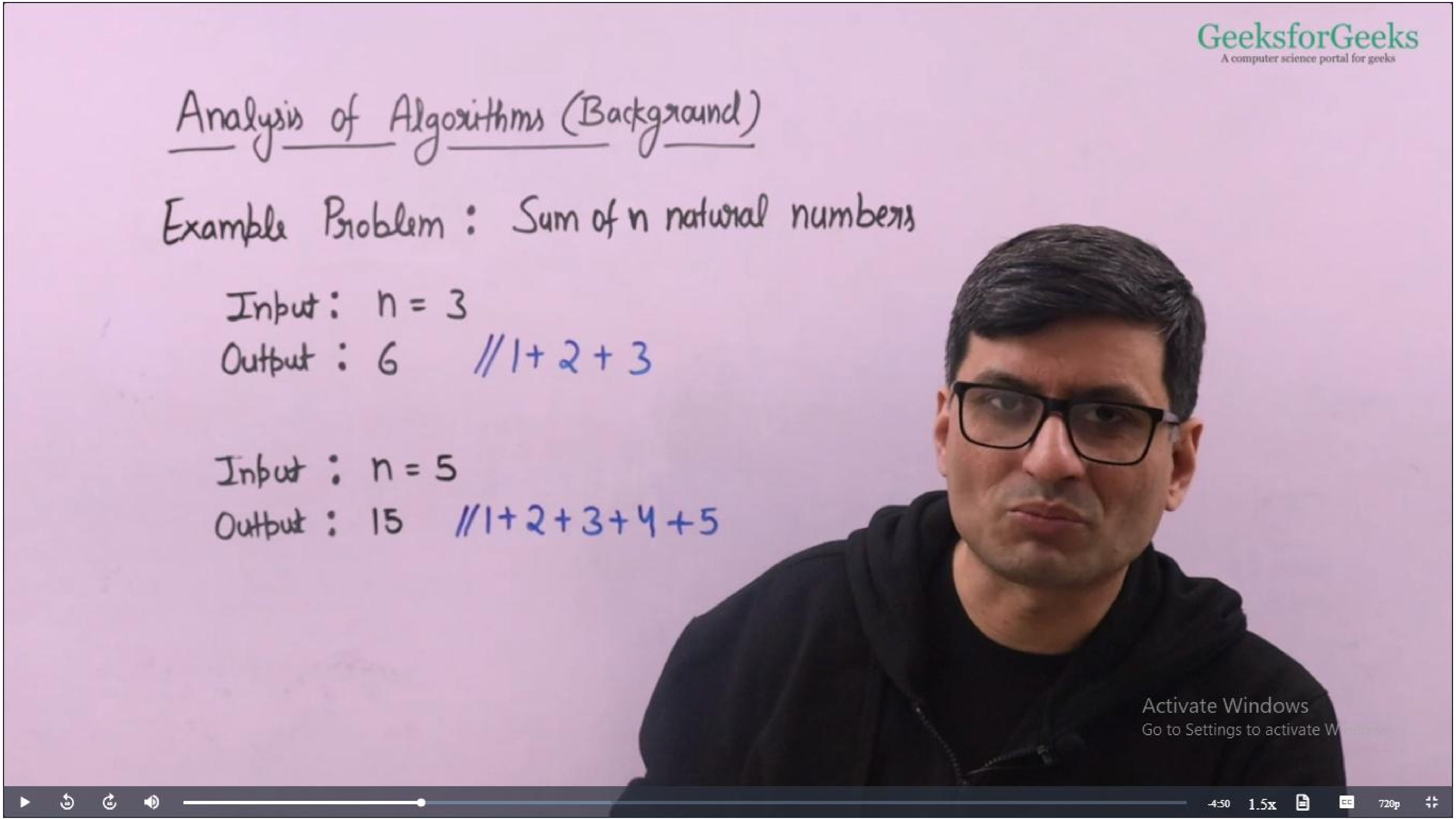
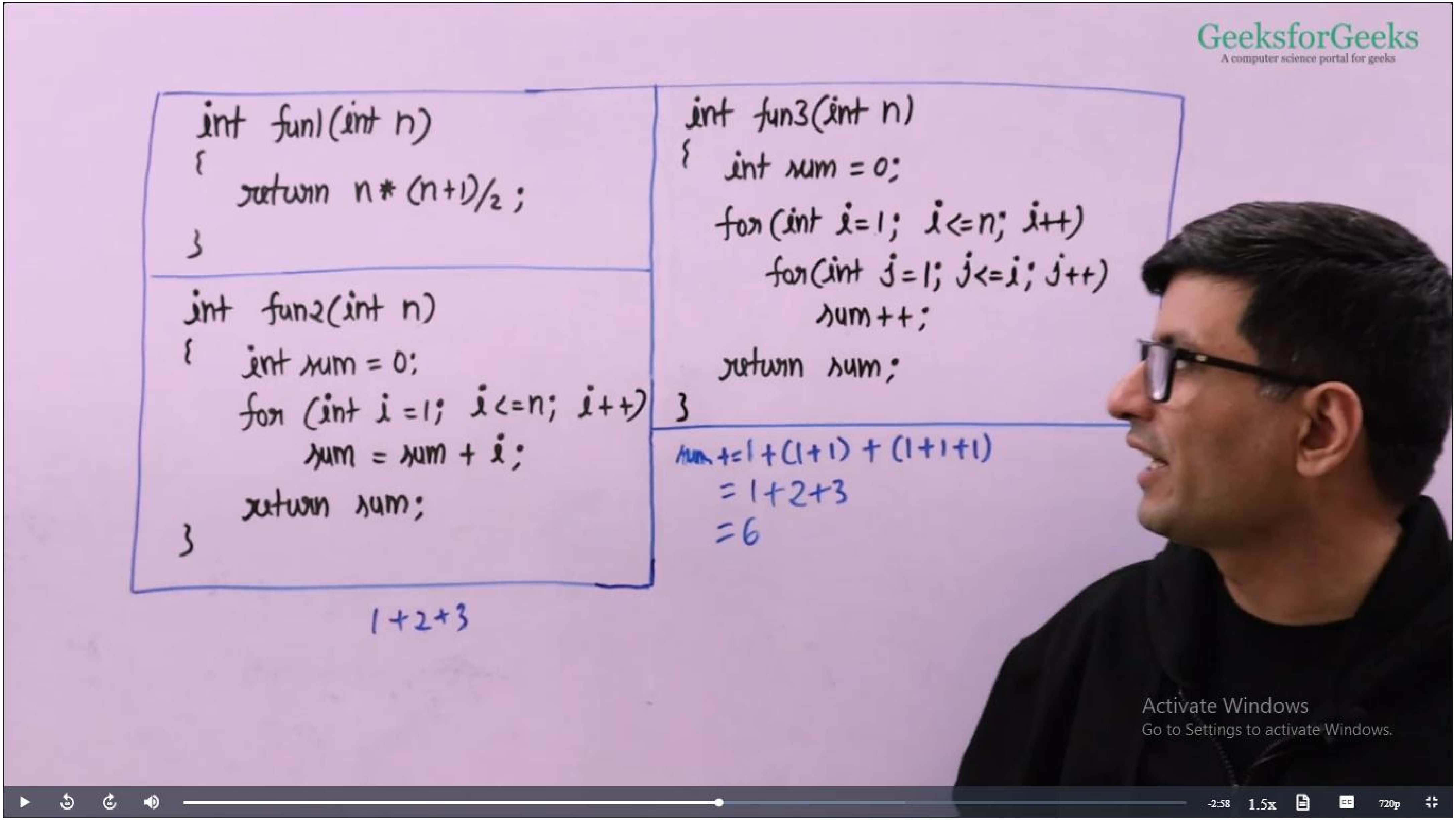
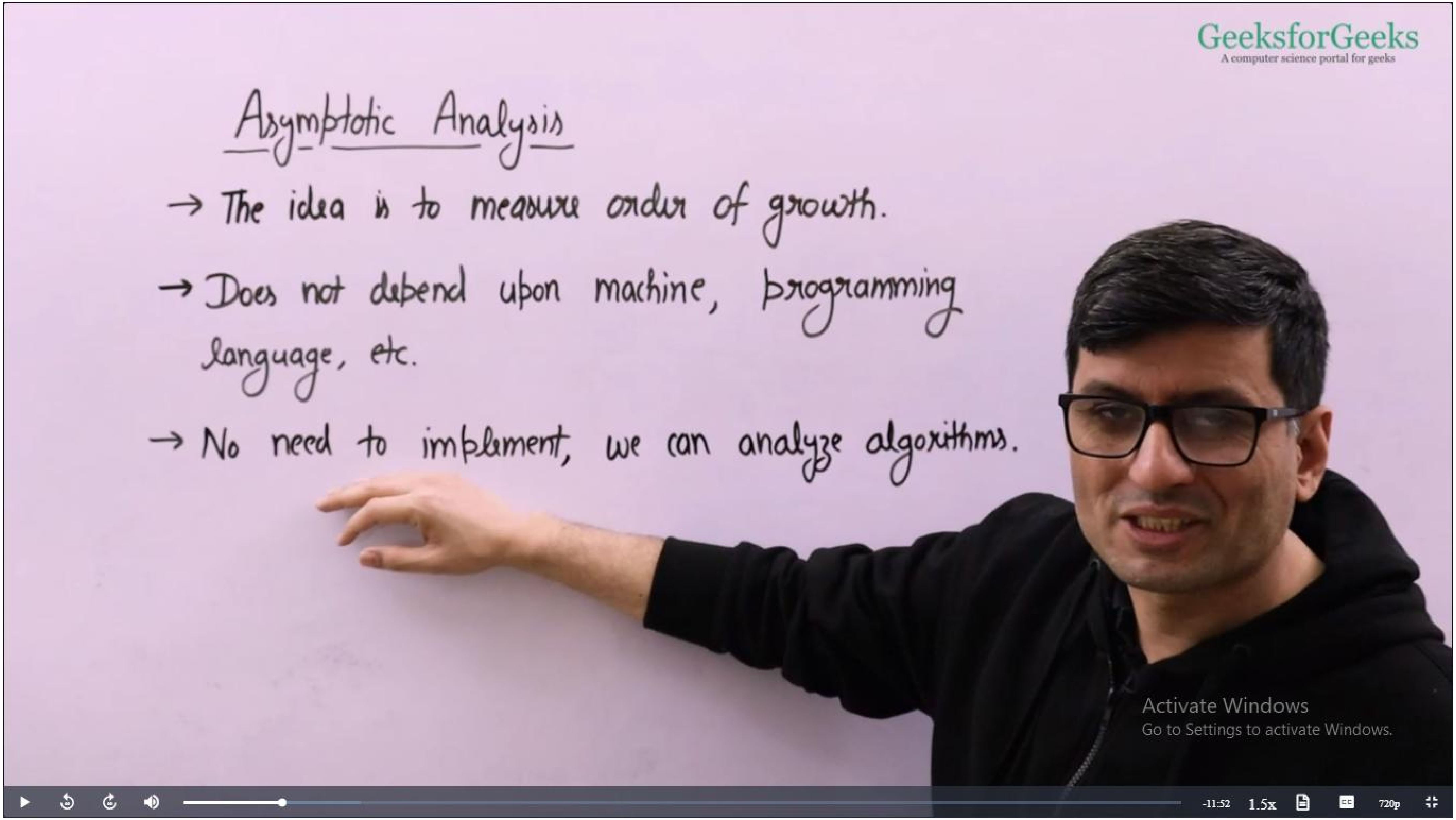
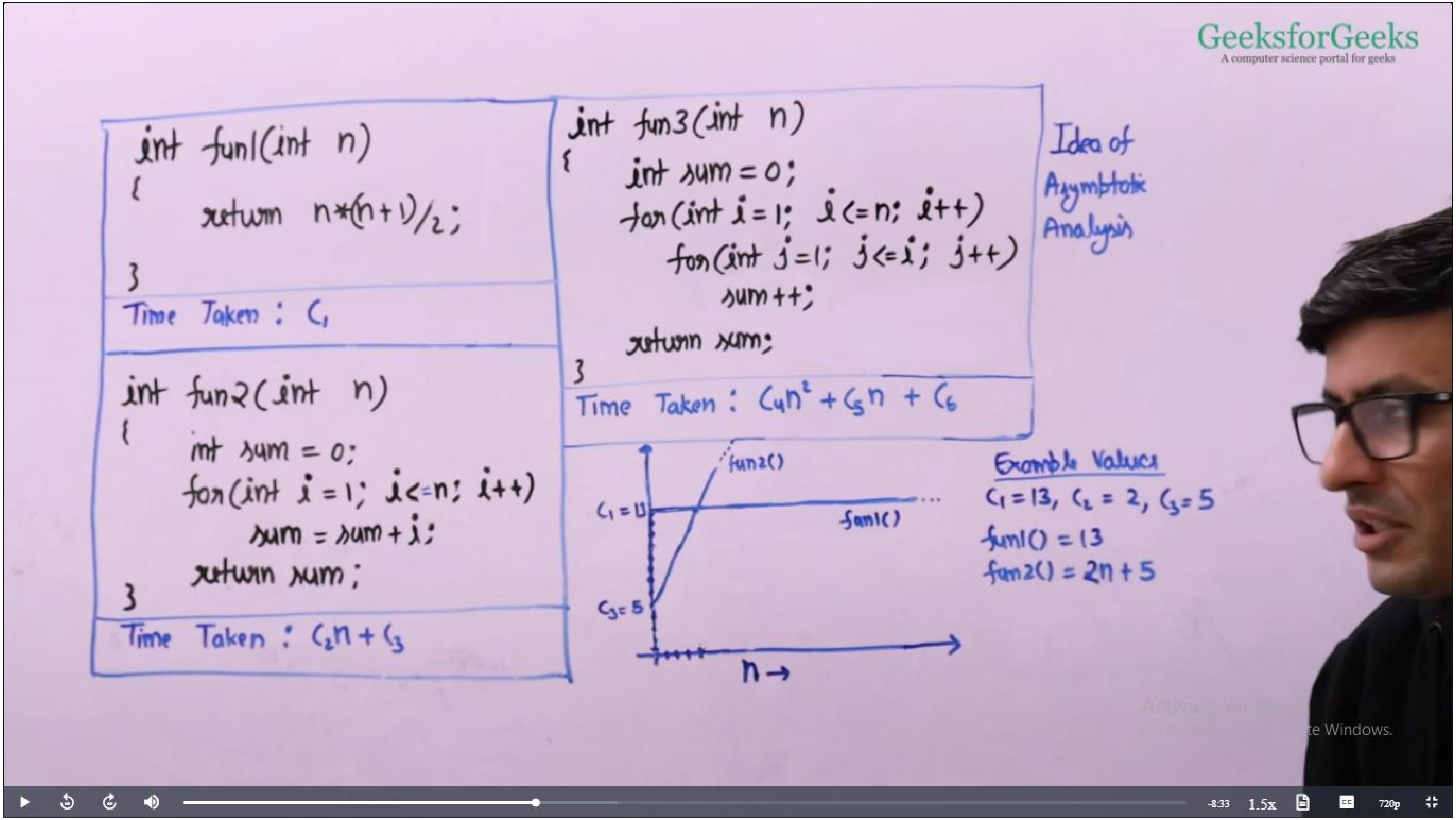

Why is DSA Important?
⇒ Make you a better Software Developer.
⇒ Helps you in getting a Job
⇒ Winning the Sport of Competitive Coding
Activate Windows
Go to Settings to activate Windows.
-0:17
1.5x
720p

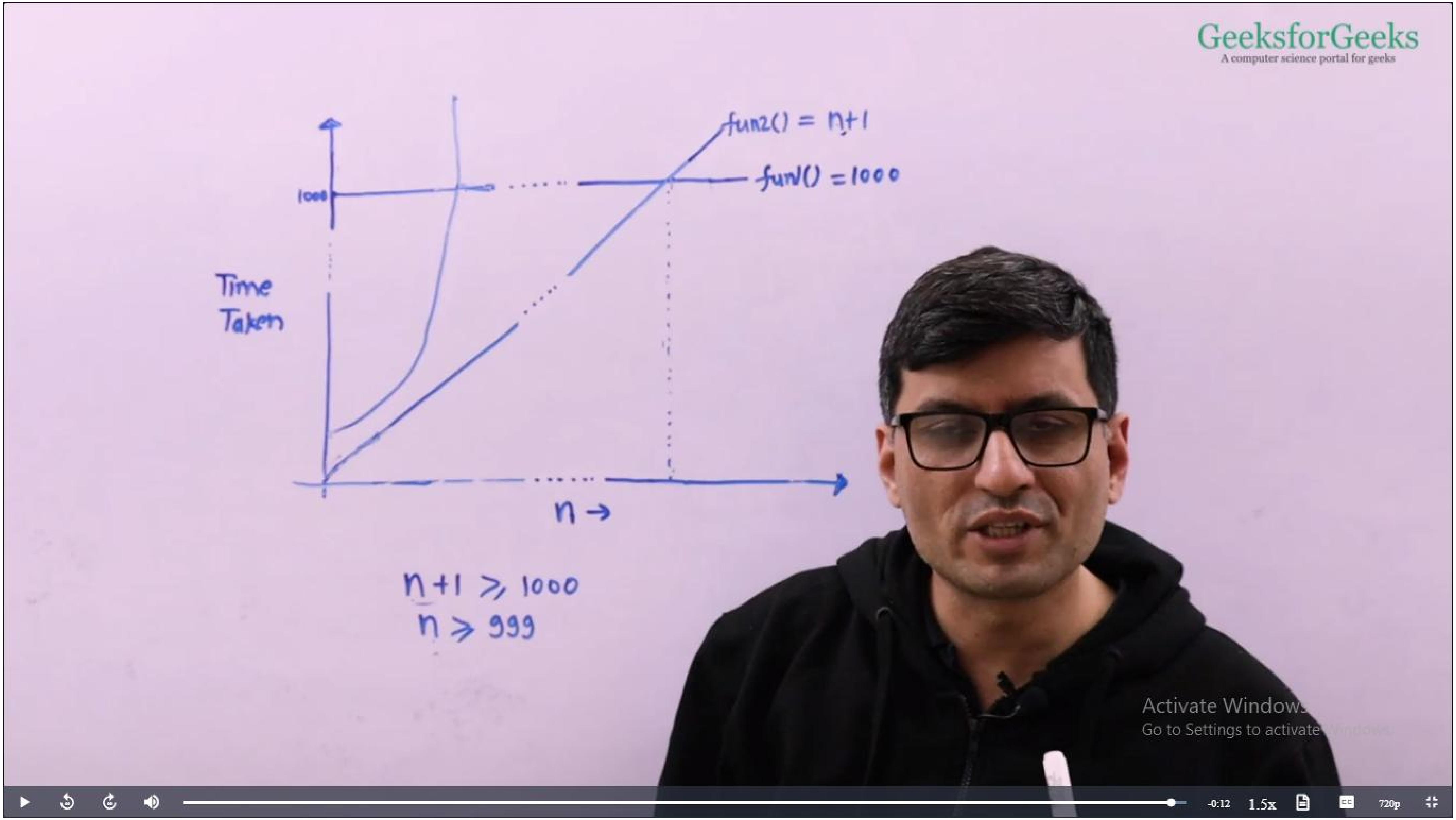

Roadmap to Learn DSA
⇒ Learn a programming language
C++
Java
Python
JavaScript
⇒ Learn DSA basics and impliment
⇒ Learn language libraries that have DSA implimented for you
⇒ Do Practice and Learning together
C++
↳ STL
Java
↳ Collectio
Activate Windows
Go to Settings to activate Windows.
-2:39
1.5x
720p

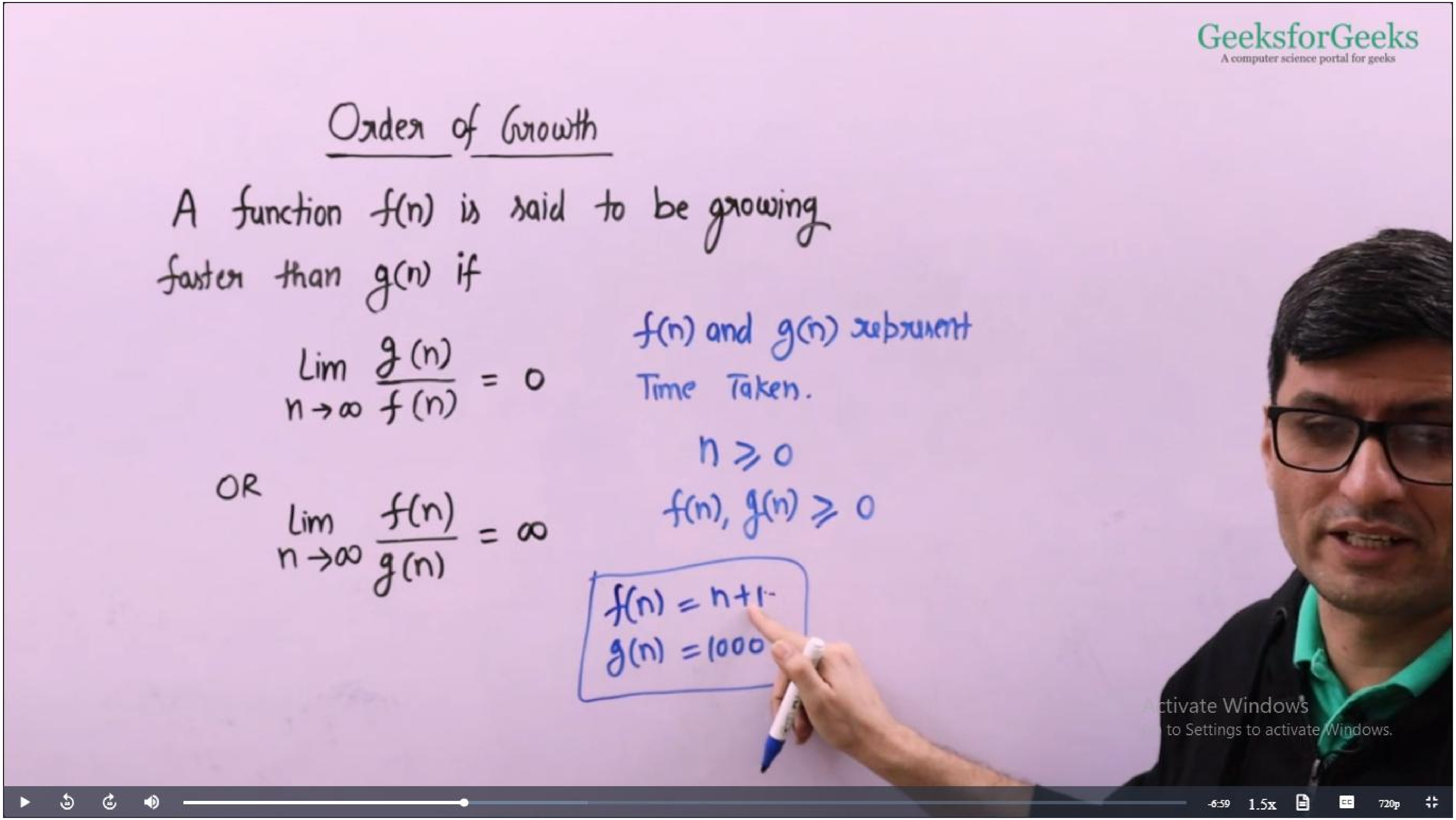

GeeksforGeeks
A computer science portal for geeks
Analysis of Algorithms (Background)
Example Problem : Sum of n natural numbers
Input : n = 3
Output : 6 //1+2+3
Input : n = 5
Output : 15 //1+2+3+4+5
Activate Windows
Go to Settings to activate W
-4:50
1.5x
720p

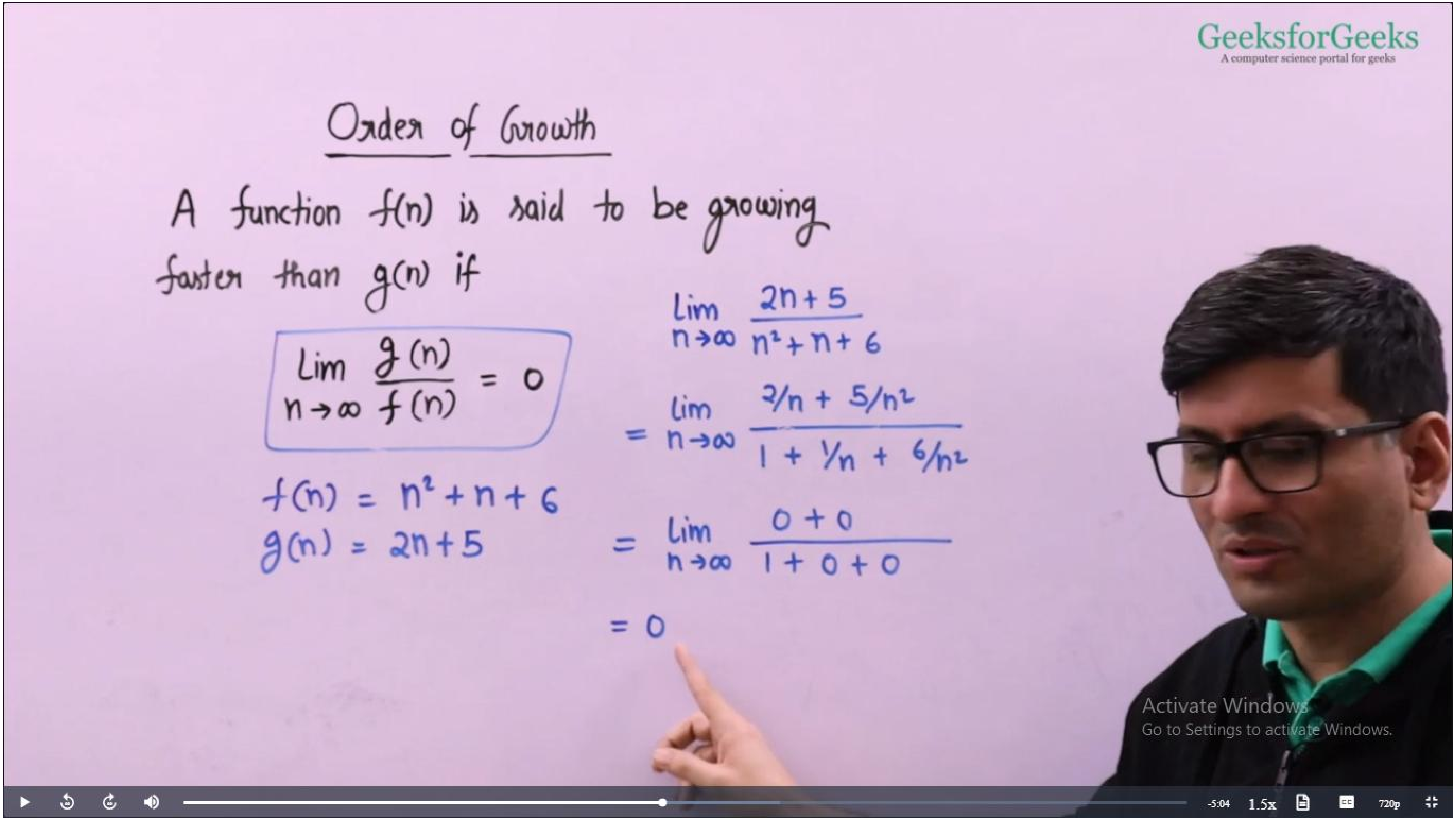

GeeksforGeeks
A computer science portal for geeks
int fun1(int n)
{
return n*(n+1)/2;
}
int fun2(int n)
{
int sum = 0;
for (int i = 1; i<=n; i++)
sum = sum + i;
return sum;
}
1+2+3
int fun3(int n)
{
int sum = 0;
for(int i=1; i<=n; i++)
for(int j=1; j<=i; j++)
sum++;
return sum;
}
sum += 1+(1+1)+(1+1+1)
= 1+2+3
= 6
Activate Windows
Go to Settings to activate Windows.
-2:58
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Asymptotic Analysis
→ The idea is to measure order of growth.
→ Does not depend upon machine, programming language, etc.
→ No need to implement, we can analyze algorithms.
Activate Windows
Go to Settings to activate Windows.
-11:52
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
int fun1(int n)
{
return n*(n+1)/2;
}
Time Taken : C1
int fun2(int n)
{
int sum = 0;
for(int i = 1; i<=n; i++)
sum = sum + i;
return sum;
}
Time Taken : C2n + C3
int fun3(int n)
{
int sum = 0;
for(int i = 1; i<=n; i++)
for(int j = 1; j<=i; j++)
sum++;
return sum;
}
Time Taken : C4n^2 + C5n + C6
Idea of Asymptotic Analysis
fun2()
C1 = 13
fun1()
C3 = 5
n →
Example Values
C1 = 13, C2 = 2, C3 = 5
fun1() = 13
fun2() = 2n + 5
-8:33
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
fun2() = n+1
fun1() = 1000
1000
Time Taken
n →
n+1 ⩾ 1000
n ⩾ 999
Activate Windows
Go to Settings to activate Windows
-0:12
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Order of Growth
A function f(n) is said to be growing faster than g(n) if
$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$
OR
$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$
f(n) and g(n) represent Time Taken.
$n \geq 0$
$f(n), g(n) \geq 0$
$f(n) = n+1$
$g(n) = 1000$
-6:59
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Order of Growth
A function f(n) is said to be growing faster than g(n) if
$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$
$f(n) = n^2 + n + 6$
$g(n) = 2n + 5$
$\lim_{n\to\infty} \frac{2n+5}{n^2+n+6}$
$= \lim_{n\to\infty} \frac{2/n + 5/n^2}{1 + 1/n + 6/n^2}$
$= \lim_{n\to\infty} \frac{0+0}{1+0+0}$
$= 0$
Activate Windows
Go to Settings to activate Windows.
-5:04
1.5x
720p

Direct Way to Find to Find and Compare Growths

1. Ignore Lower Order Terms
2. Ignore Leading Term Constant

Examples: $f(n) = \cancel{2}n^2 + \cancel{n} + \cancel{6}$, Order of Growth : n^2 (Quadratic)

$g(n) = \cancel{100}n + \cancel{3}$, Order of Growth : n (Linear)

How do we compare terms ?

$$c < \log\log n < \log n < n^{1/3} < n^{1/2} < n$$

$$< n^2 < n^3 < n^4 < 2^n < n^n$$

① $f(n) = c_1 \log n + c_2$

$g(n) = c_3 n + c_4 \log \log n + c_5$

$f(n) : \log n$

$g(n) : n$

② $f(n) = c_1 n^2 + c_2 n + c_3$

$g(n) = c_4 n \log n + c_5 n + c_6$

$f(n) : n^2$

$g(n) : n \log n$

How do we compare terms?

$$c < \log \log n < \log n < n^{1/3} < n^{1/2} < n$$

$$< n^2 < n^3 < n^4 < 2^n < n^n$$

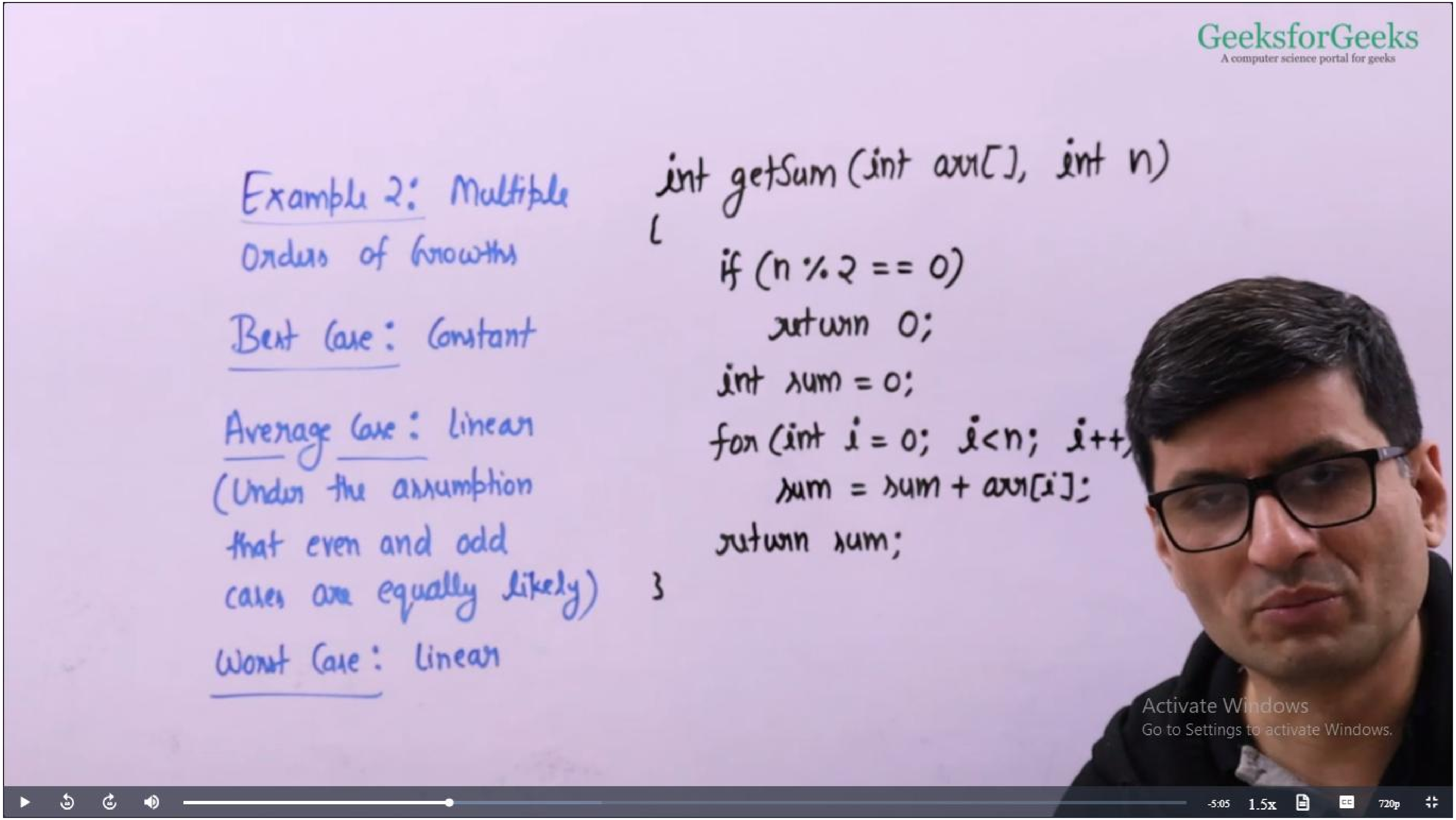
Best, Average and Worst Cases

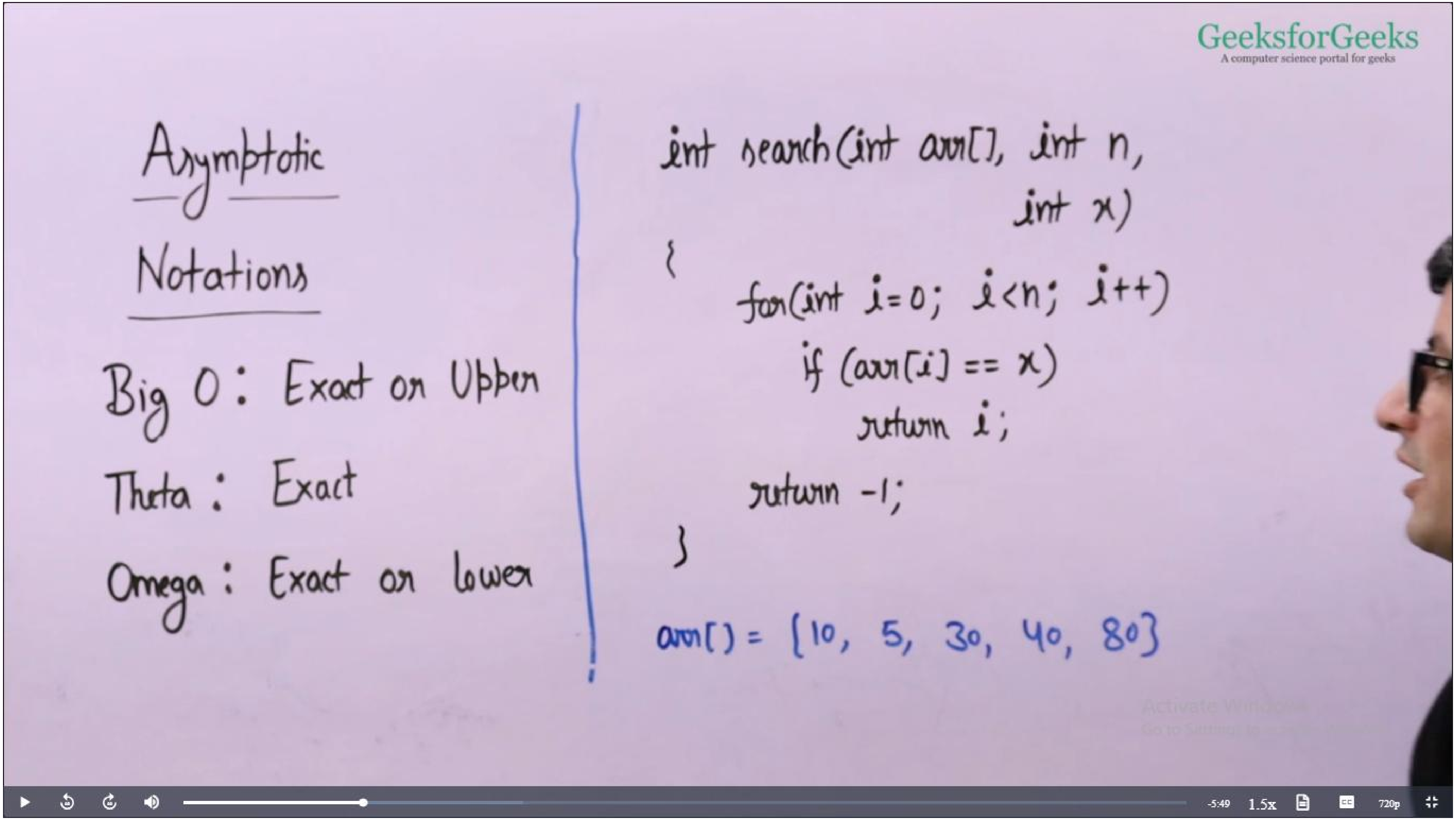
Example 1: Simple function with same order of growth for every input.

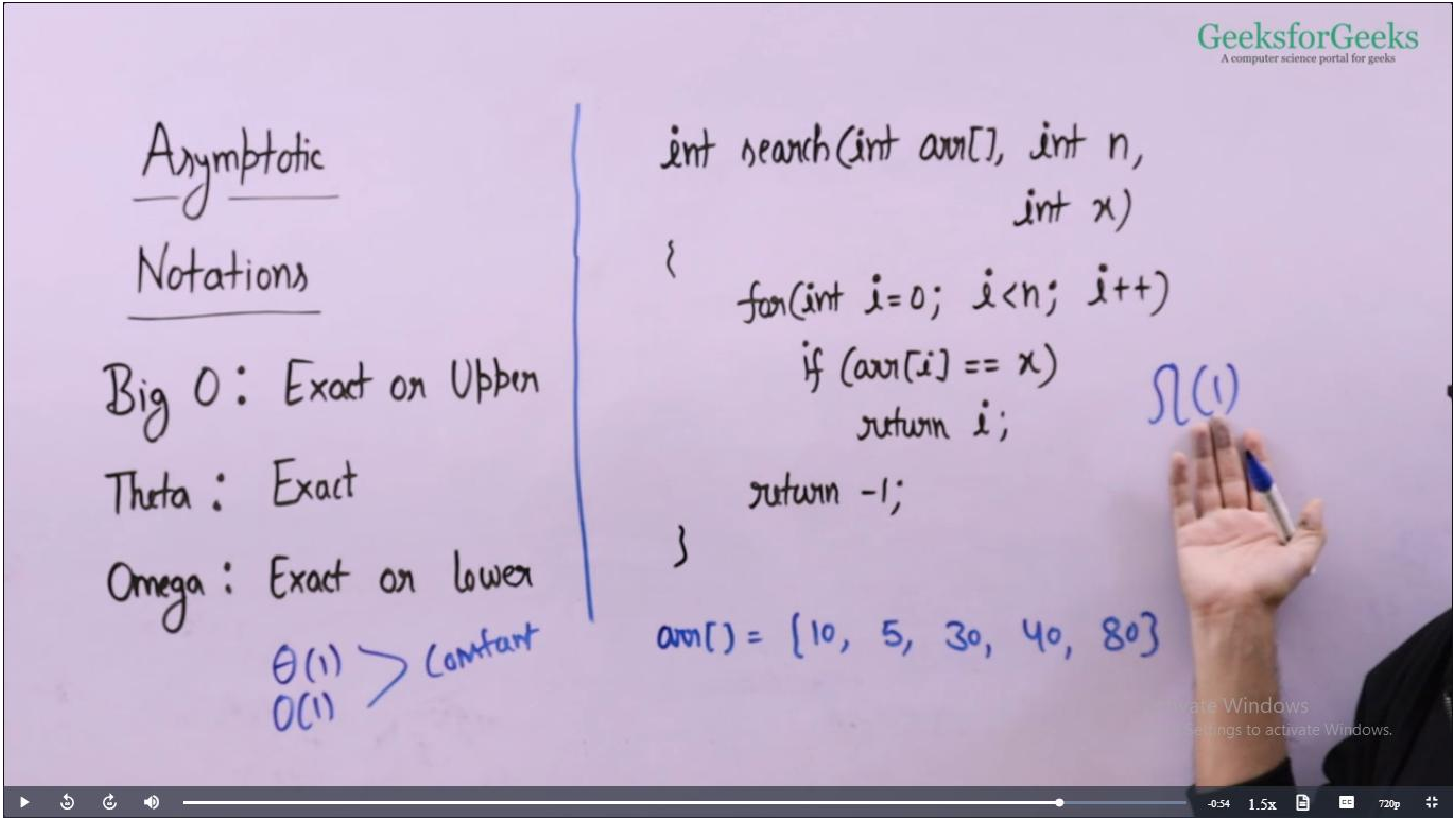
```
int getSum(int arr[], int n)
{
    int sum = 0;
    for (int i=0; i<n; i++)
        sum = sum + arr[i];
    return sum;
}
```

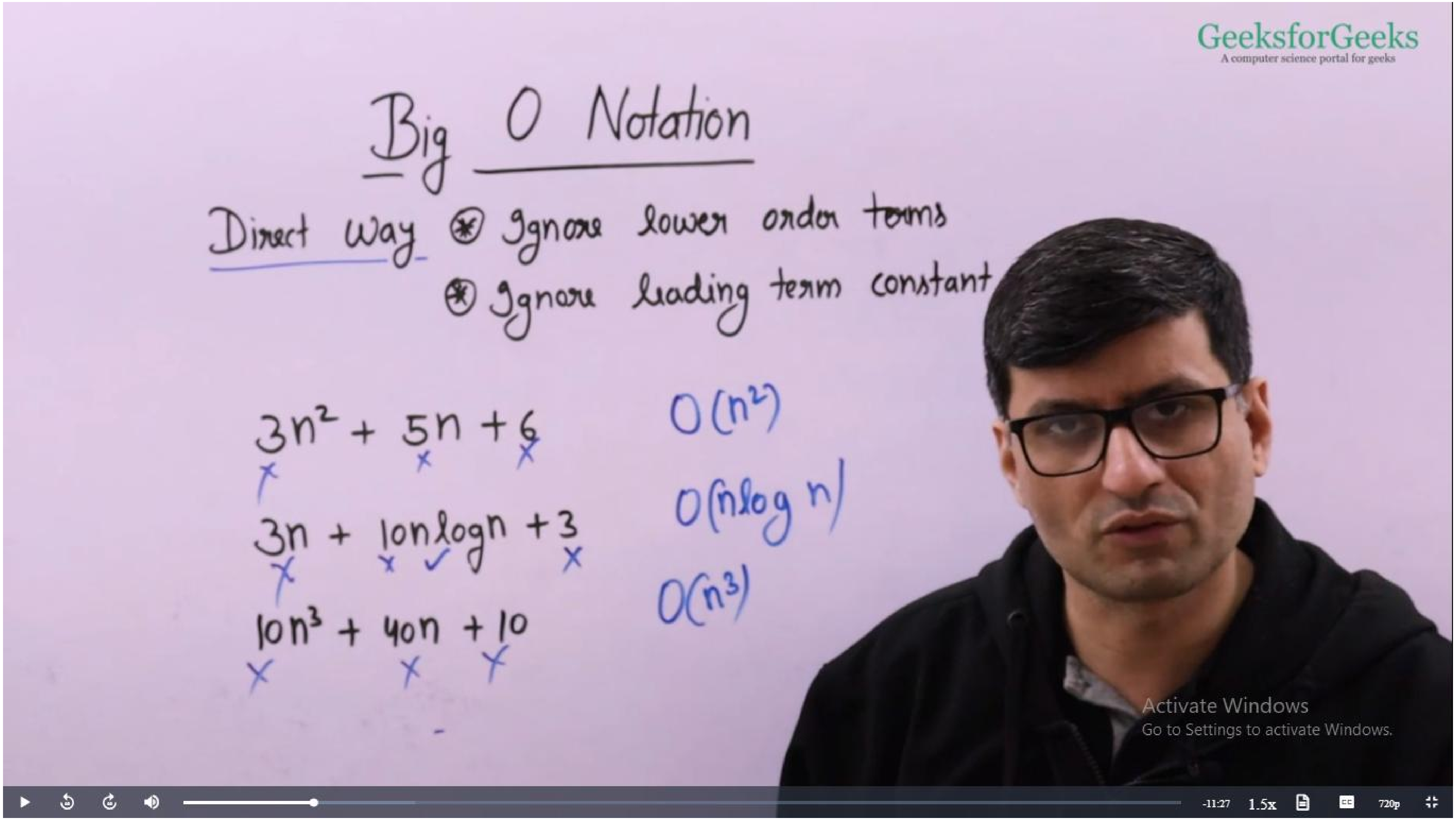
Time Taken : $C_1 n + C_2$

Order of Growth : n
(or linear)

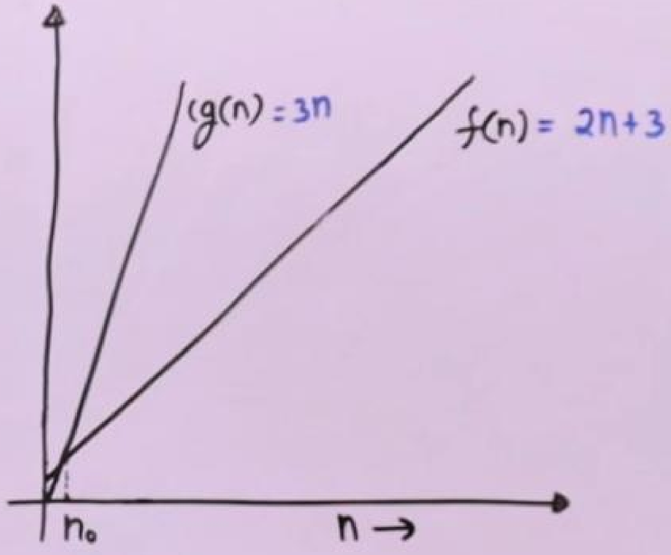

GeeksforGeeks
A computer science portal for geeks
Example 2: Multiple
Orders of Growths
Best Case: Constant
Average Case: linear
(Under the assumption
that even and odd
cases are equally likely)
Worst Case: linear
int getSum(int arr[], int n)
{
if (n % 2 == 0)
return 0;
int sum = 0;
for (int i = 0; i<n; i++)
sum = sum + arr[i];
return sum;
}
Activate Windows
Go to Settings to activate Windows.
-5:05
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Asymptotic
Notations
Big O : Exact or Upper
Theta : Exact
Omega : Exact or lower
int search(int arr[], int n,
int x)
{
for(int i=0; i<n; i++)
if (arr[i] == x)
return i;
return -1;
}
arr[] = {10, 5, 30, 40, 80}
-5:49
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Asymptotic
Notations
Big O : Exact or Upper
Theta : Exact
Omega : Exact or lower
θ(1)
O(1)
Constant
int search(int arr[], int n,
int x)
{
for(int i=0; i<n; i++)
if (arr[i] == x)
return i;
return -1;
}
Ω(1)
arr[] = {10, 5, 30, 40, 80}
-0:54
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Big O Notation
Direct way
Ignore lower order terms
Ignore leading term constant
$3n^2 + 5n + 6$
$O(n^2)$
$3n + 10n\log n + 3$
$O(n\log n)$
$10n^3 + 40n + 10$
$O(n^3)$
Activate Windows
Go to Settings to activate Windows.
-11:27
1.5x
720p

We say $f(n) = O(g(n))$ iff there exist constants c and n_0 such that $f(n) \le c\,g(n)$ for all $n \geq n_0$



Example

$$f(n) = 2n + 3$$

can be written as $O(n)$ $[g(n) = n]$

let us take $c = 3$

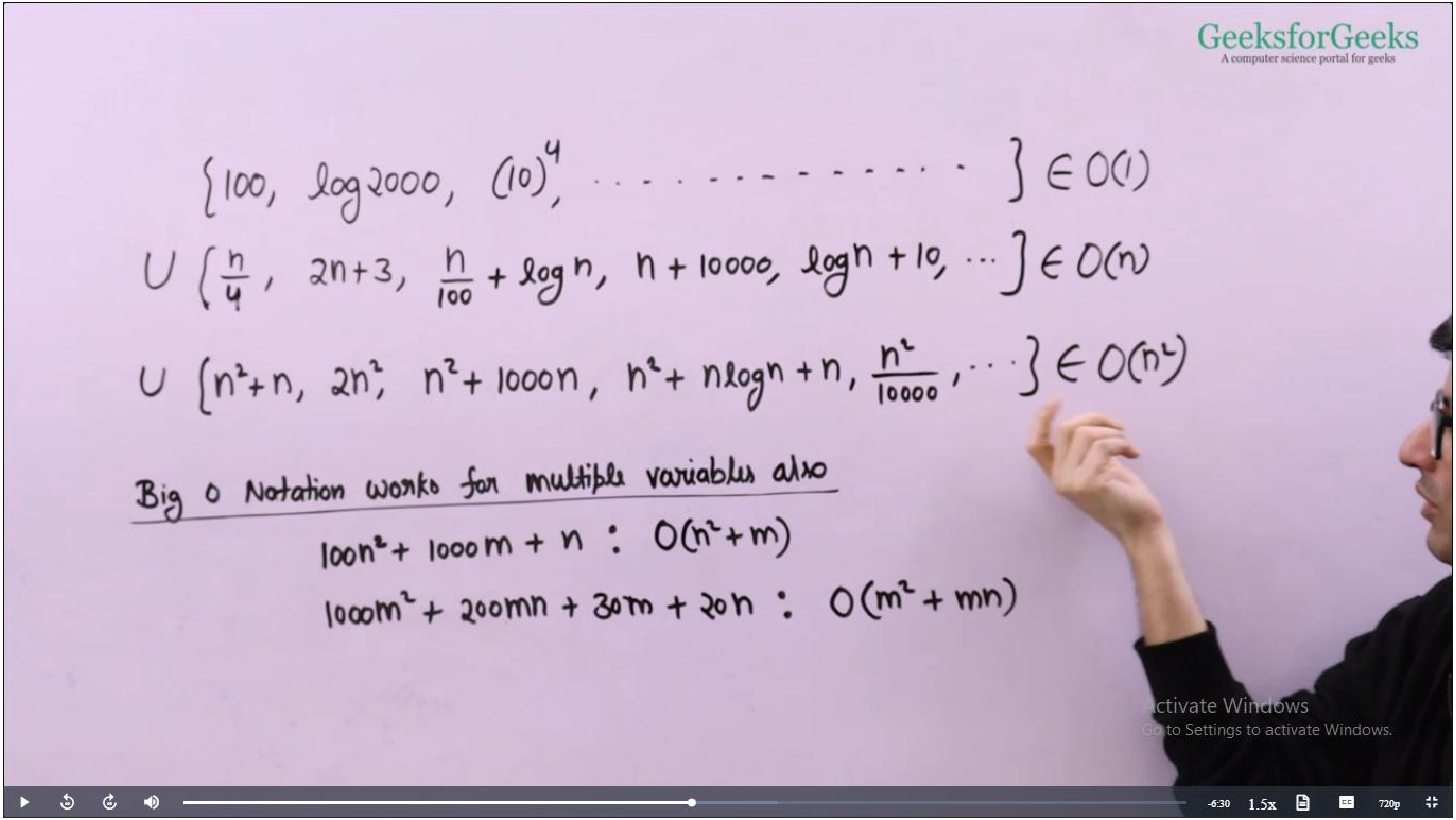
$$2n + 3 \le 3n$$

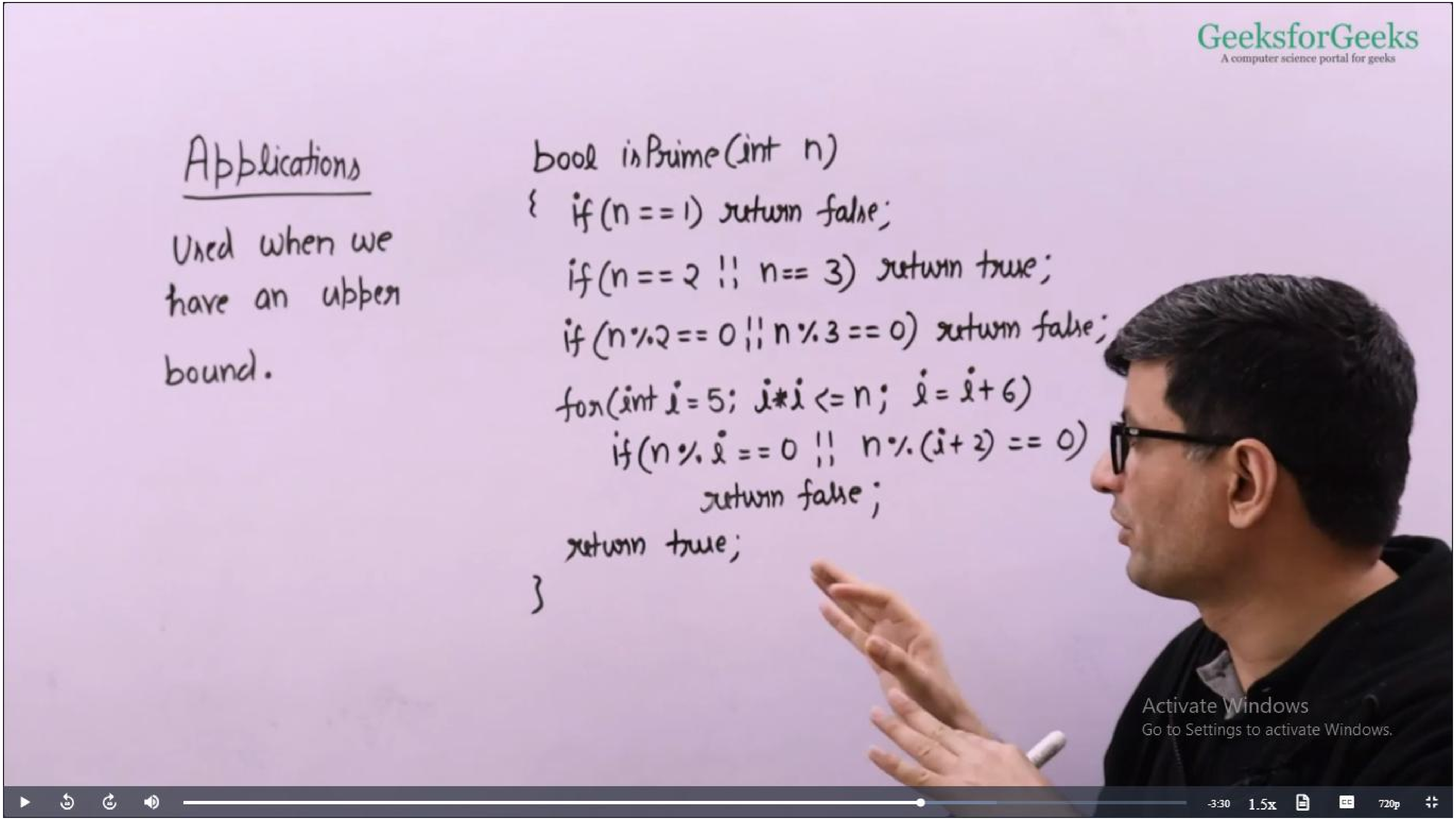
$$3 \le n$$

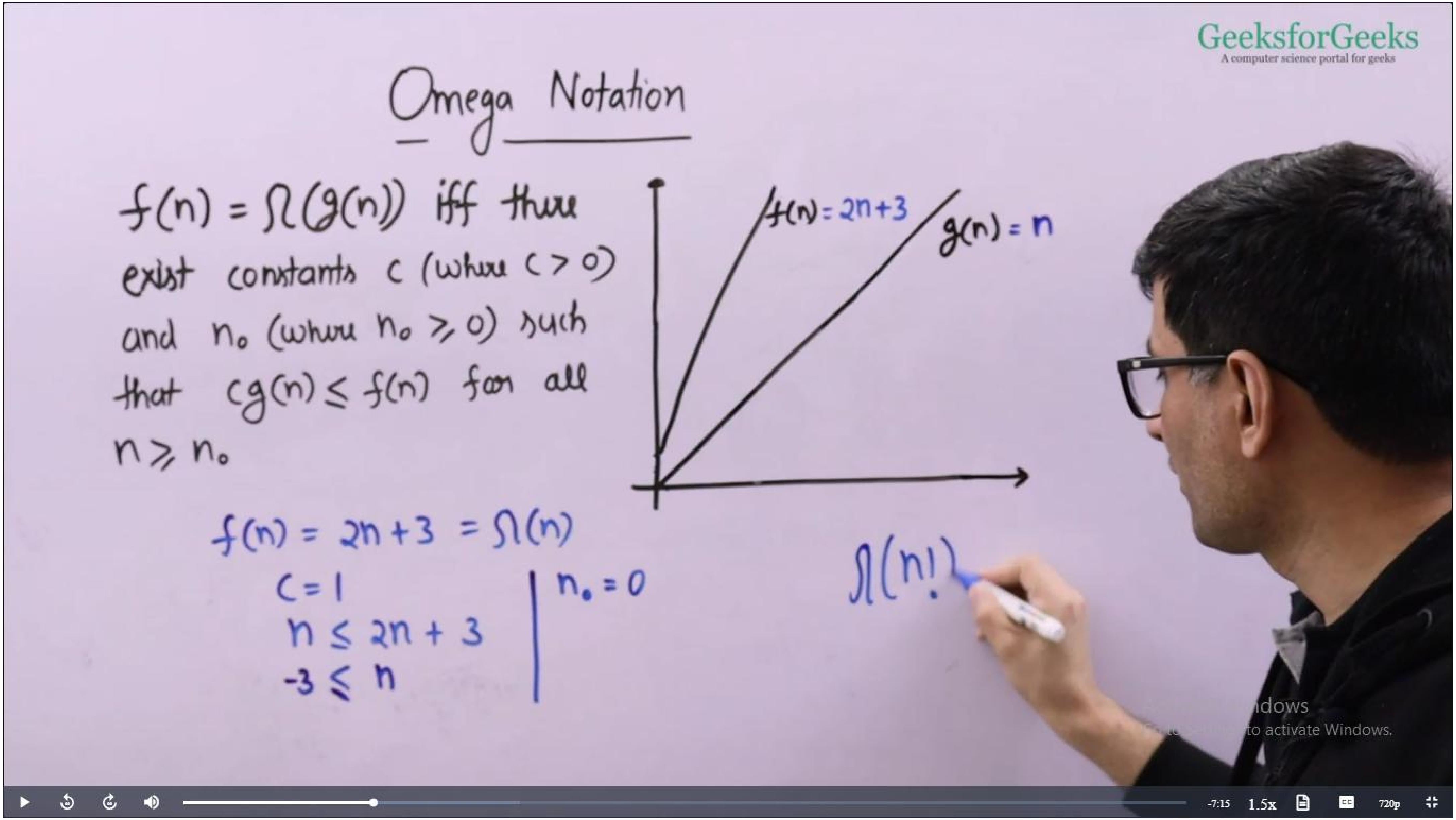
We get $n_0 = 3$

$$2n+3 \le cn$$

for all $n \geq n_0$


GeeksforGeeks
A computer science portal for geeks
$\{100, \log 2000, (10)^4, \ldots\ldots\ldots\} \in O(1)$
$\cup \{\frac{n}{4}, 2n+3, \frac{n}{100} + \log n, n + 10000, \log n + 10, \ldots\} \in O(n)$
$\cup \{n^2+n, 2n^2, n^2 + 1000n, n^2 + n\log n + n, \frac{n^2}{10000}, \ldots\} \in O(n^2)$
Big O Notation works for multiple variables also
$100n^2 + 1000m + n : O(n^2+m)$
$1000m^2 + 200mn + 30m + 20n : O(m^2 + mn)$
Activate Windows
Go to Settings to activate Windows.
-6:30
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Applications
Used when we have an upper bound.
bool isPrime(int n)
{ if(n==1) return false;
if(n==2 || n==3) return true;
if(n%2==0 || n%3==0) return false;
for(int i=5; i*i<=n; i=i+6)
if(n%i==0 || n%(i+2)==0)
return false;
return true;
}
Activate Windows
Go to Settings to activate Windows.
-3:30
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Omega Notation
f(n) = Ω(g(n)) iff there exist constants c (where c > 0) and n₀ (where n₀ ≥ 0) such that cg(n) ≤ f(n) for all n ≥ n₀
f(n) = 2n+3
g(n) = n
f(n) = 2n + 3 = Ω(n)
c = 1
n ≤ 2n + 3
-3 ≤ n
n₀ = 0
Ω(n!)
to activate Windows.
-7:15
1.5x
720p

* $\{n^2,\ 2n^2+5,\ 1000n^2,\ 2n^3+n,\ \dots\} \in \Omega(n^2)$

$\cup\ \{2n+3,\ 100n+\log n,\ \dots\dots\} \in \Omega(n)$

$\cup\ \{5000,\ (10)^5,\ \log 2000,\ \dots\} \in \Omega(1)$

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

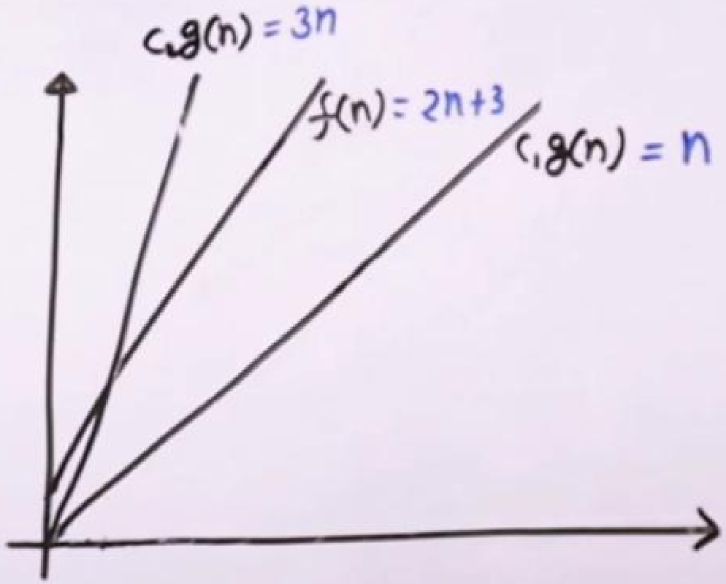
$g(n) = \Omega(f(n))$

Theta Notation

$f(n) = \Theta(g(n))$ iff there exist constants c_1, c_2 (where $c_1 > 0$ and $c_2 > 0$) and n_0 (where $n_0 \geqslant 0$) such that

$$c_1 g(n) \leqslant f(n) \leqslant c_2 g(n)$$

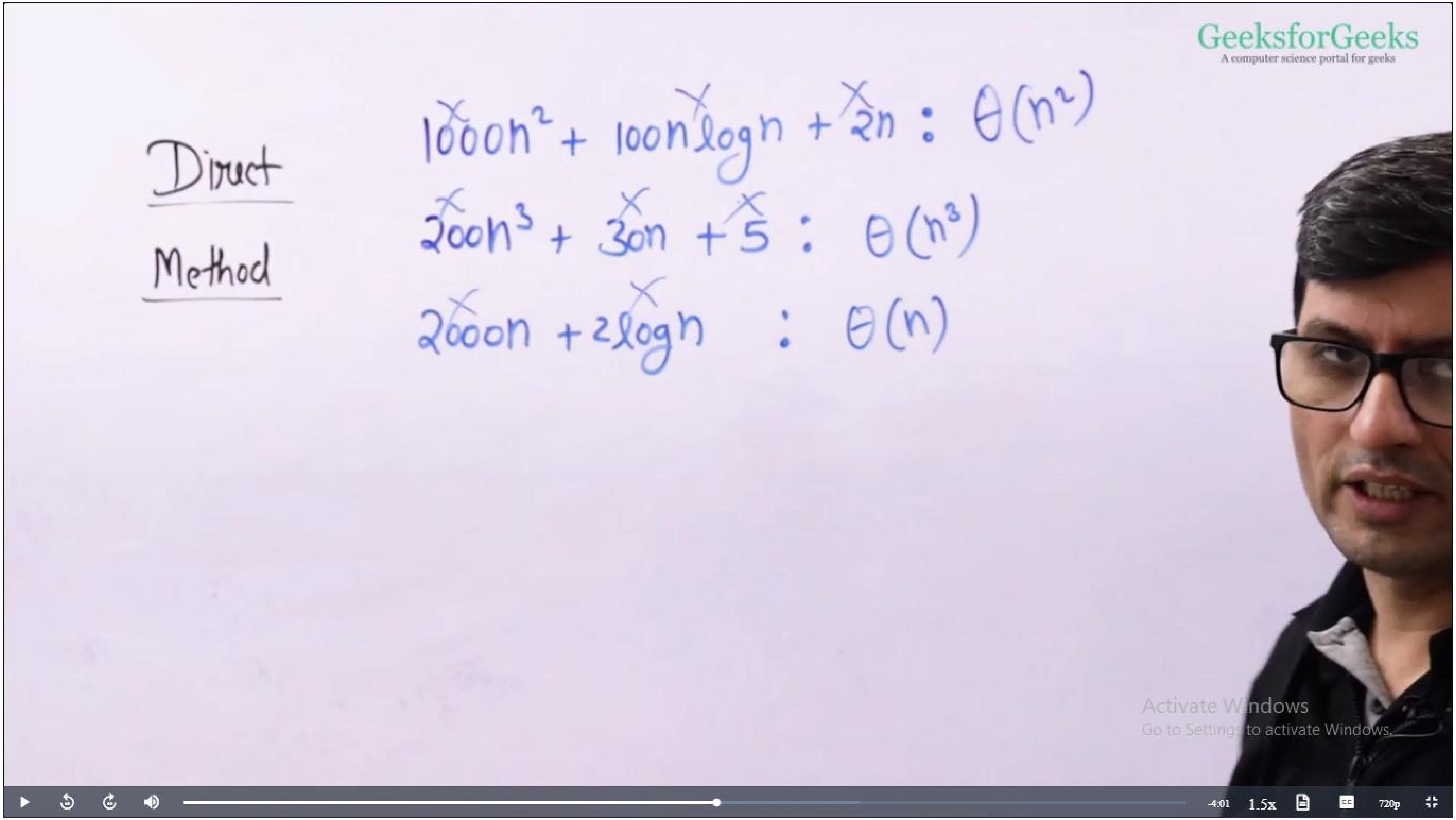
for all $n \geqslant n_0$

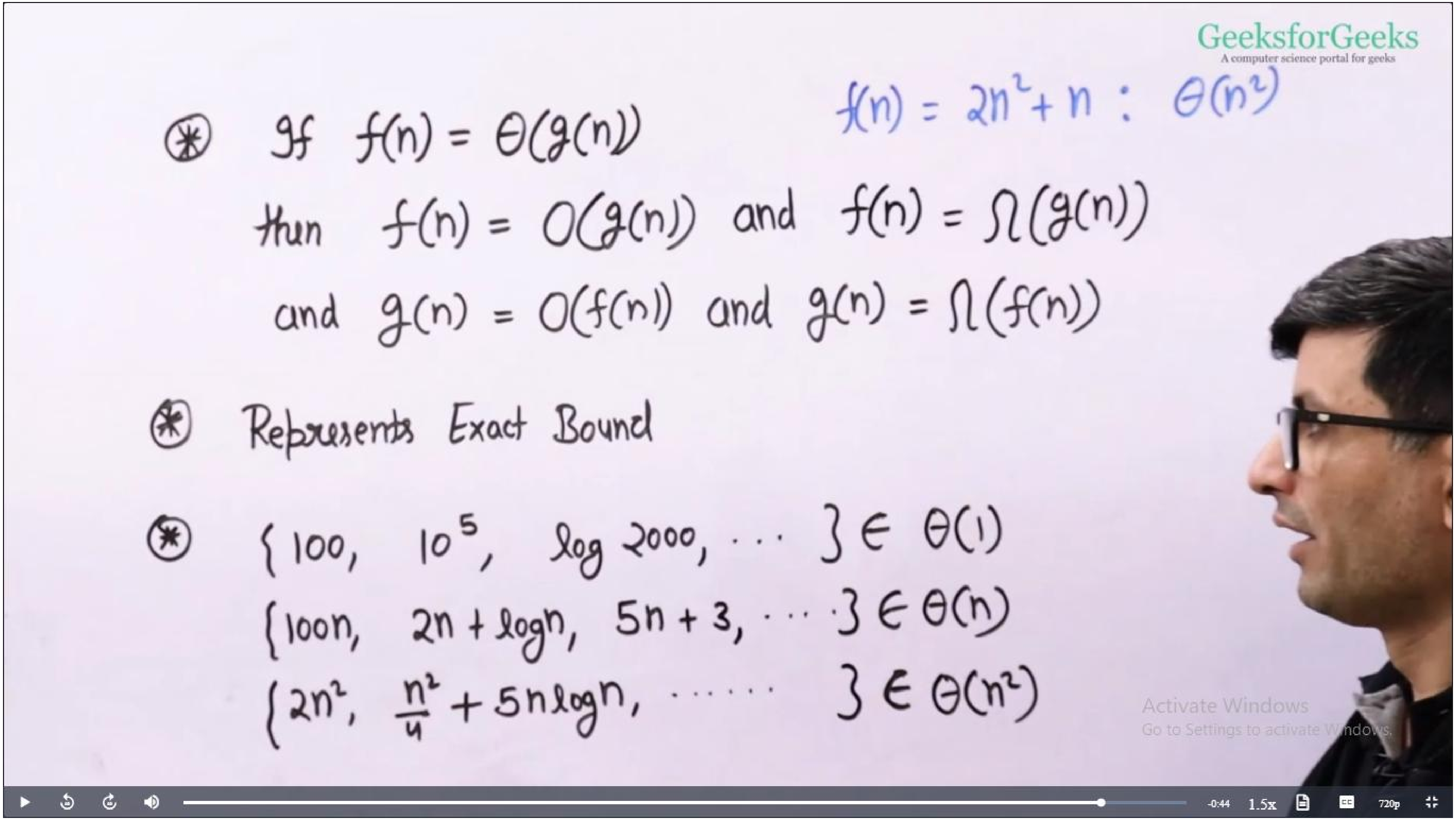


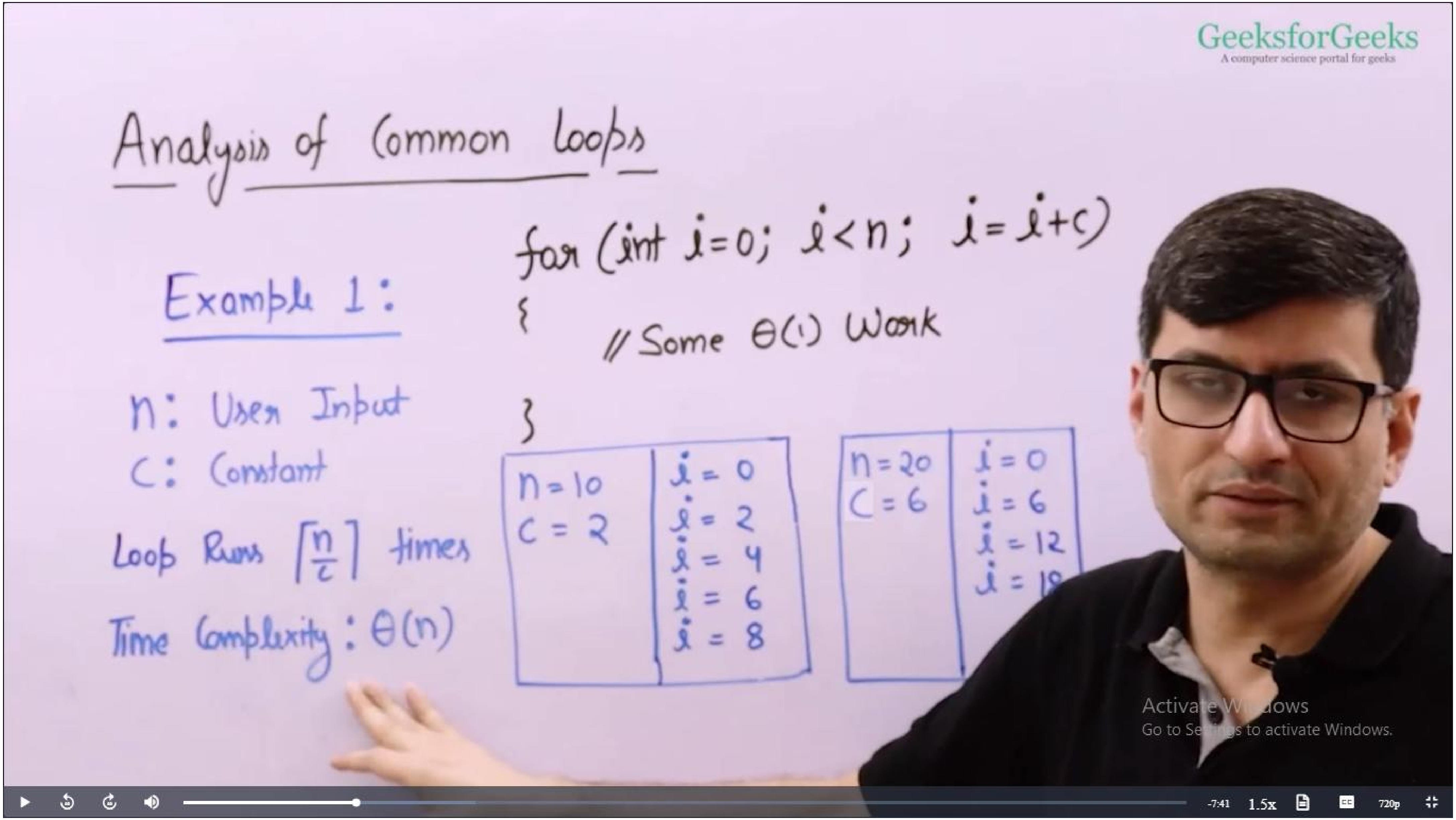
Example: $f(n) = 2n+3$: $\Theta(n)$

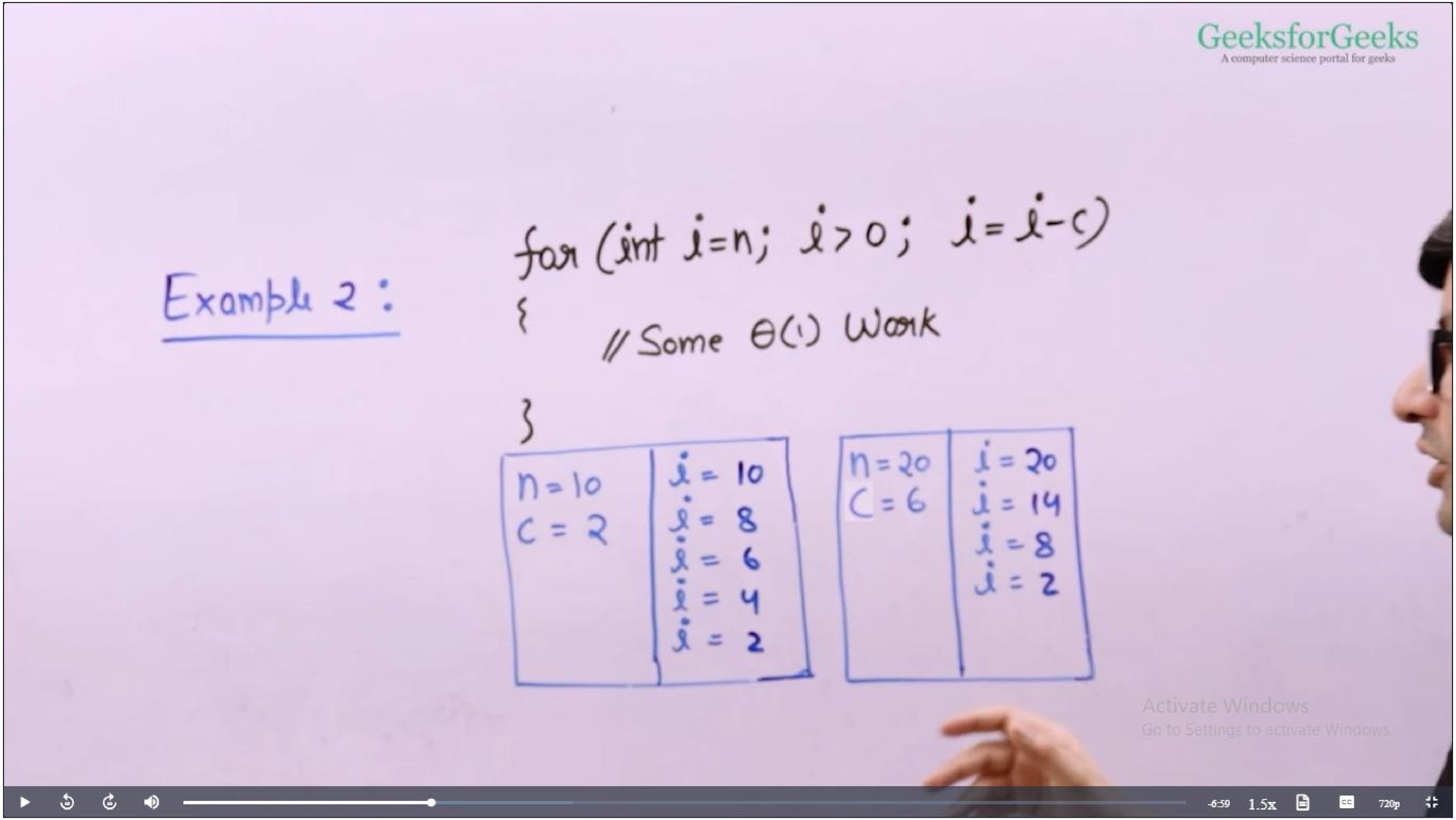
$c_1 = 1, \quad c_2 = 3$

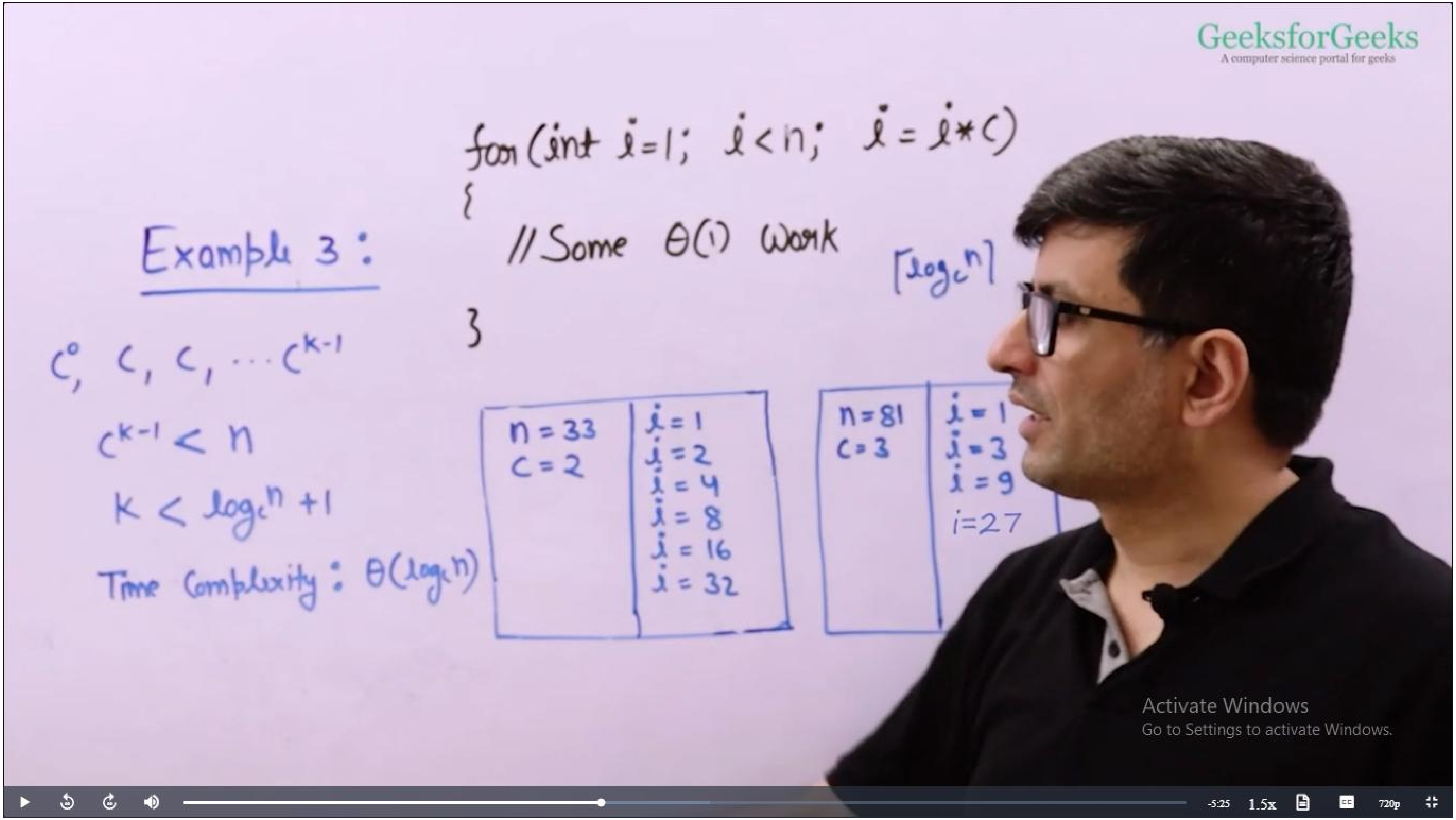
$$\underbrace{1 \times n \leqslant 2n}_{n \geqslant 0} \underbrace{+3 \leqslant 3n}_{n \geqslant 3} \qquad n_0 = 3$$

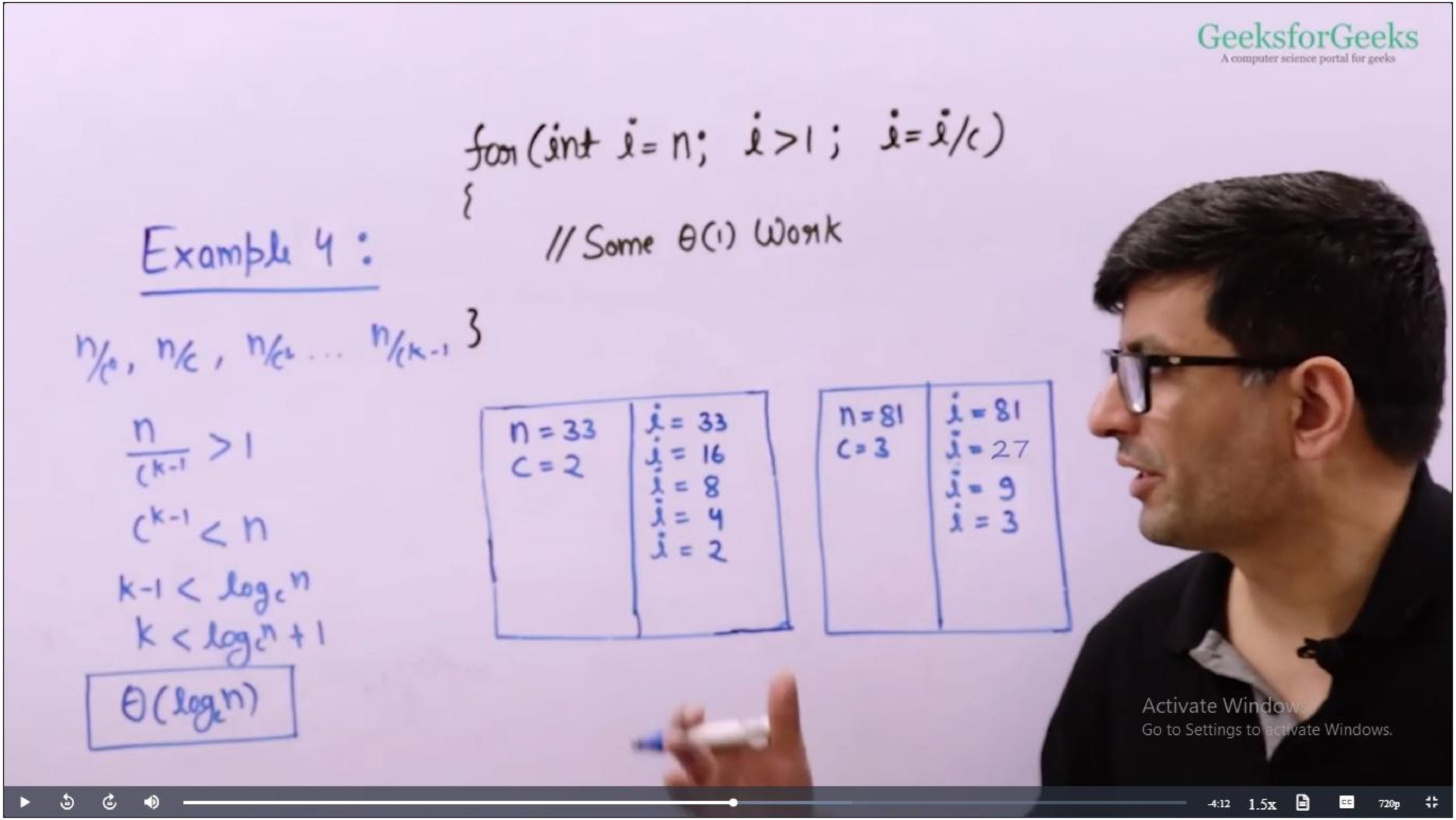

GeeksforGeeks
A computer science portal for geeks
Direct Method
$\cancel{1000}n^2 + \cancel{100n}\log n + \cancel{2n} : \Theta(n^2)$
$\cancel{200}n^3 + \cancel{30n} + \cancel{5} : \Theta(n^3)$
$\cancel{2000}n + \cancel{2\log n} : \Theta(n)$
Activate Windows
Go to Settings to activate Windows.
-4:01
1.5x
720p

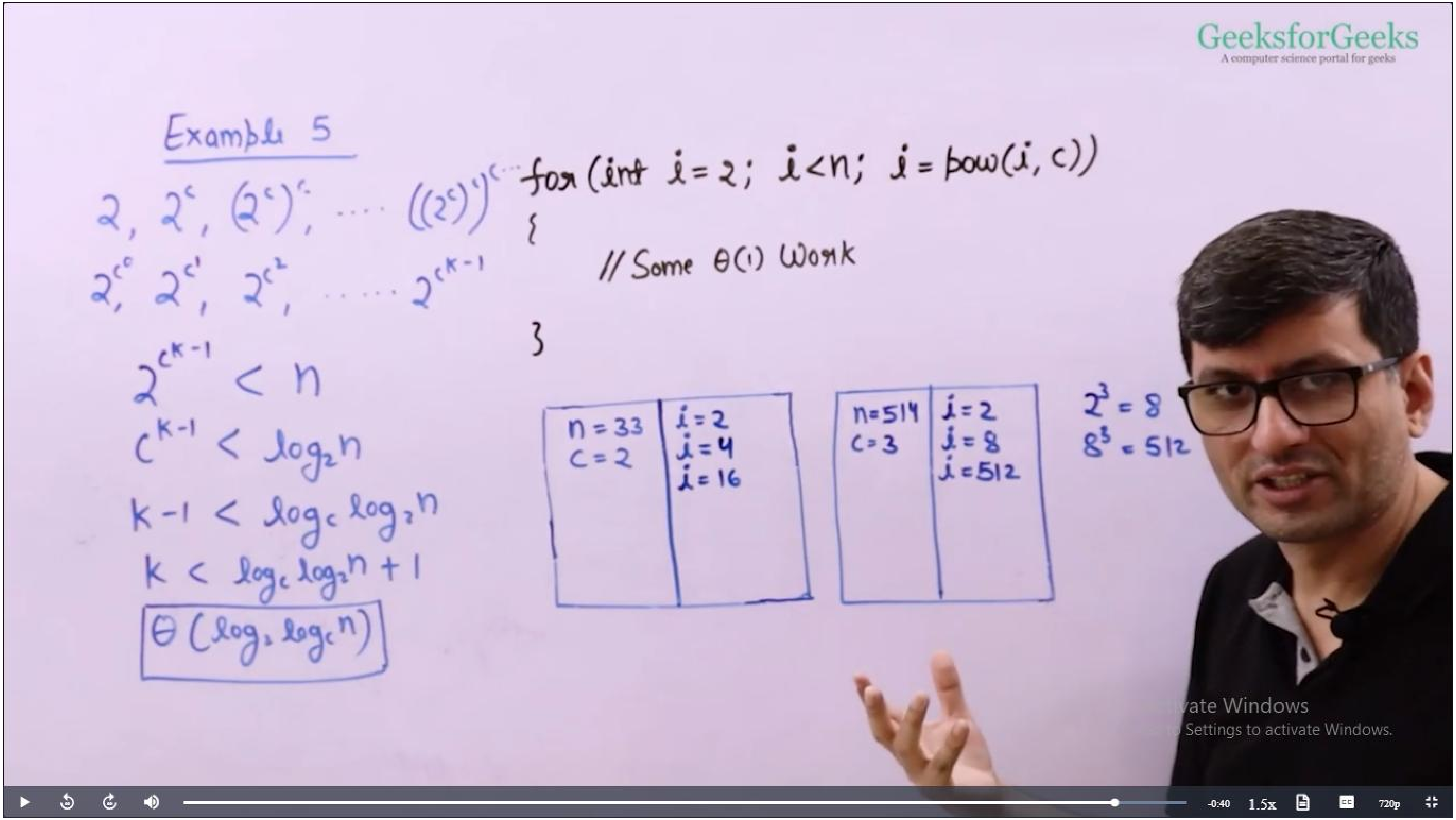

GeeksforGeeks
A computer science portal for geeks
$f(n) = 2n^2 + n : \Theta(n^2)$
* If $f(n) = \Theta(g(n))$
then $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
and $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$
* Represents Exact Bound
* $\{100, 10^5, \log 2000, \cdots\} \in \Theta(1)$
$\{100n, 2n + \log n, 5n + 3, \cdots\} \in \Theta(n)$
$\{2n^2, \frac{n^2}{4} + 5n\log n, \cdots\} \in \Theta(n^2)$
Activate Windows
Go to Settings to activate Windows.
-0:44
1.5x
720p

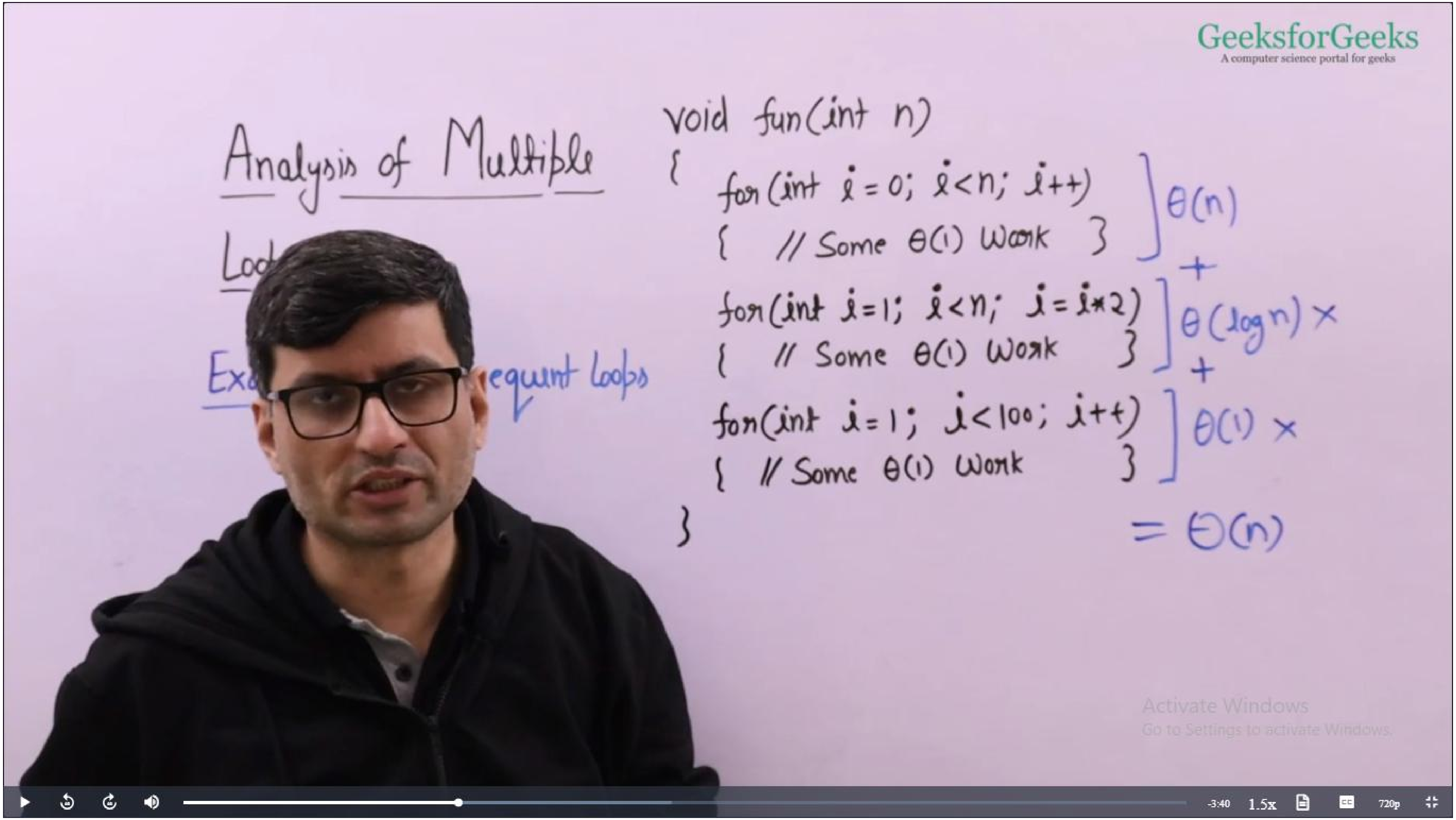

GeeksforGeeks
A computer science portal for geeks
Analysis of Common Loops
for (int i=0; i<n; i=i+c)
{
// Some Θ(1) Work
}
Example 1:
n: User Input
c: Constant
Loop Runs ⌈n/c⌉ times
Time Complexity: Θ(n)
n=10
c = 2
i = 0
i = 2
i = 4
i = 6
i = 8
n=20
C=6
i=0
i=6
i=12
-7:41
1.5x
720p

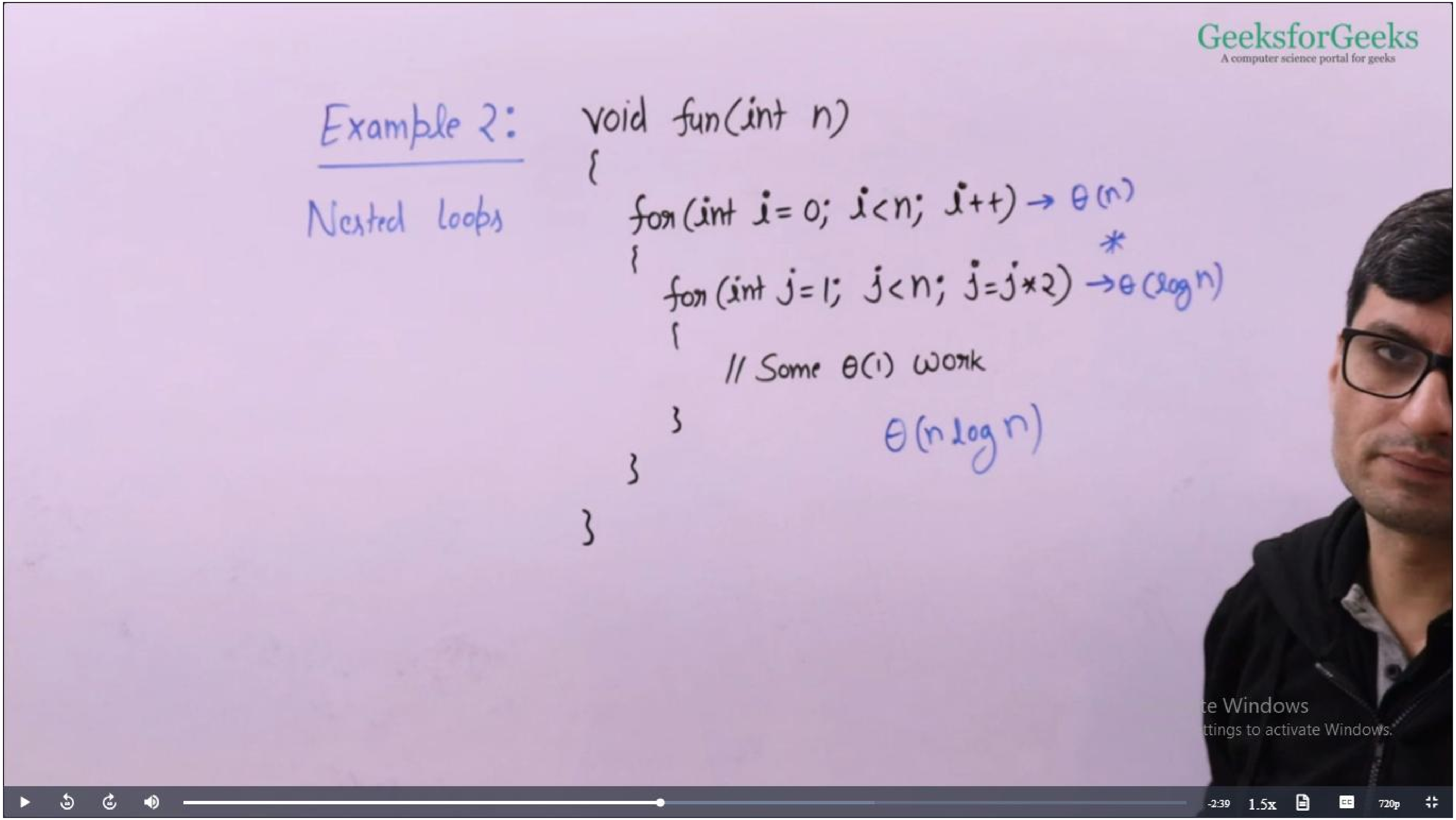

GeeksforGeeks
A computer science portal for geeks
Example 2 :
for (int i=n; i>0; i=i-c)
{
// Some Θ(1) Work
}
n=10
c = 2
i = 10
i = 8
i = 6
i = 4
i = 2
n=20
C=6
i = 20
i = 14
i = 8
i = 2
-6:59
1.5x
720p

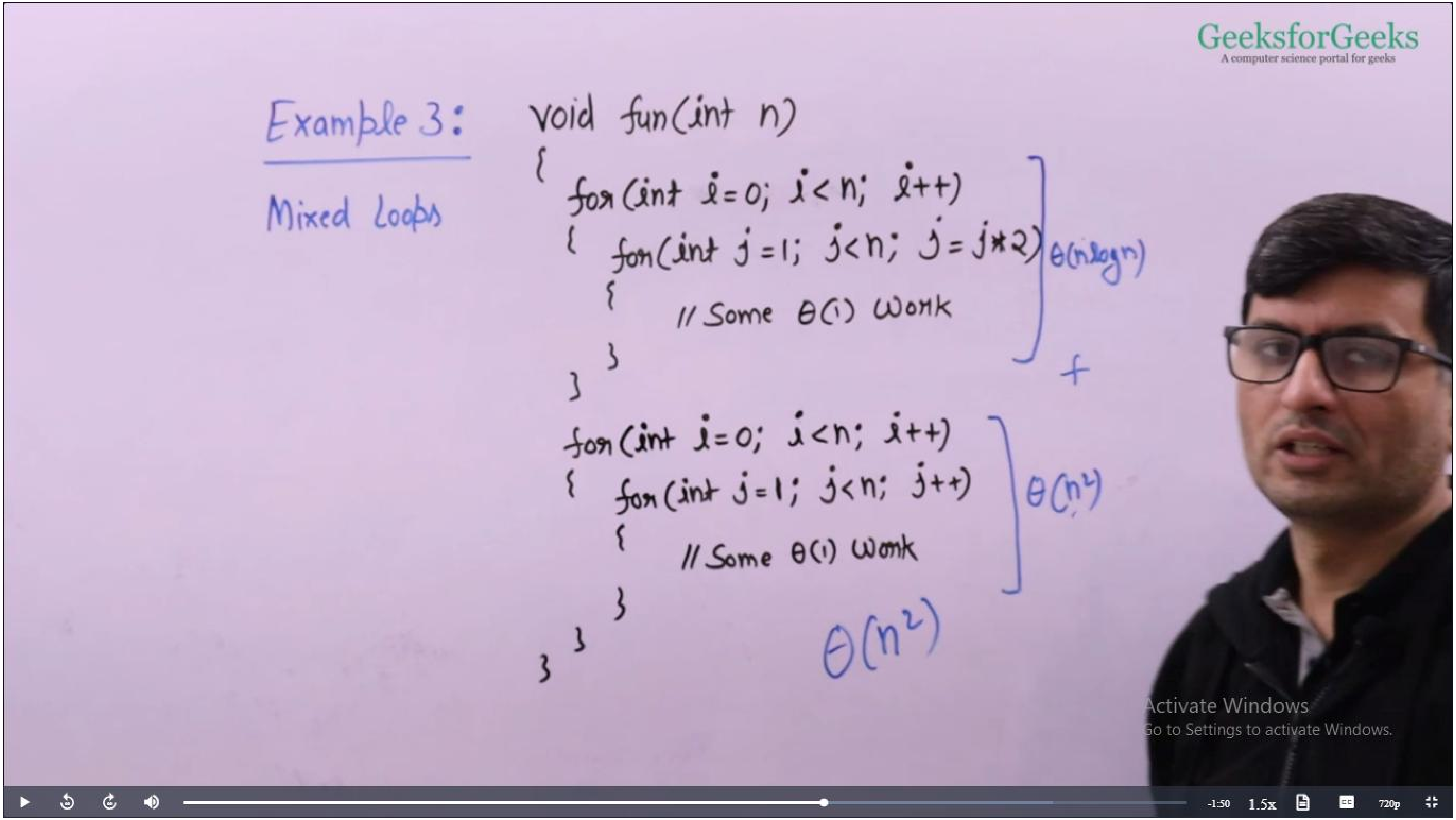

GeeksforGeeks
A computer science portal for geeks
Example 3:
for(int i=1; i<n; i=i*c)
{
//Some θ(1) Work
}
⌈log_c n⌉
c^0, c, c, ... c^(k-1)
c^(k-1) < n
k < log_c n + 1
Time Complexity: θ(log_c n)
n=33
c=2
i=1
i=2
i=4
i=8
i=16
i=32
n=81
c=3
i=1
i=3
i=9
i=27
Activate Windows
Go to Settings to activate Windows.
-5:25
1.5x
720p

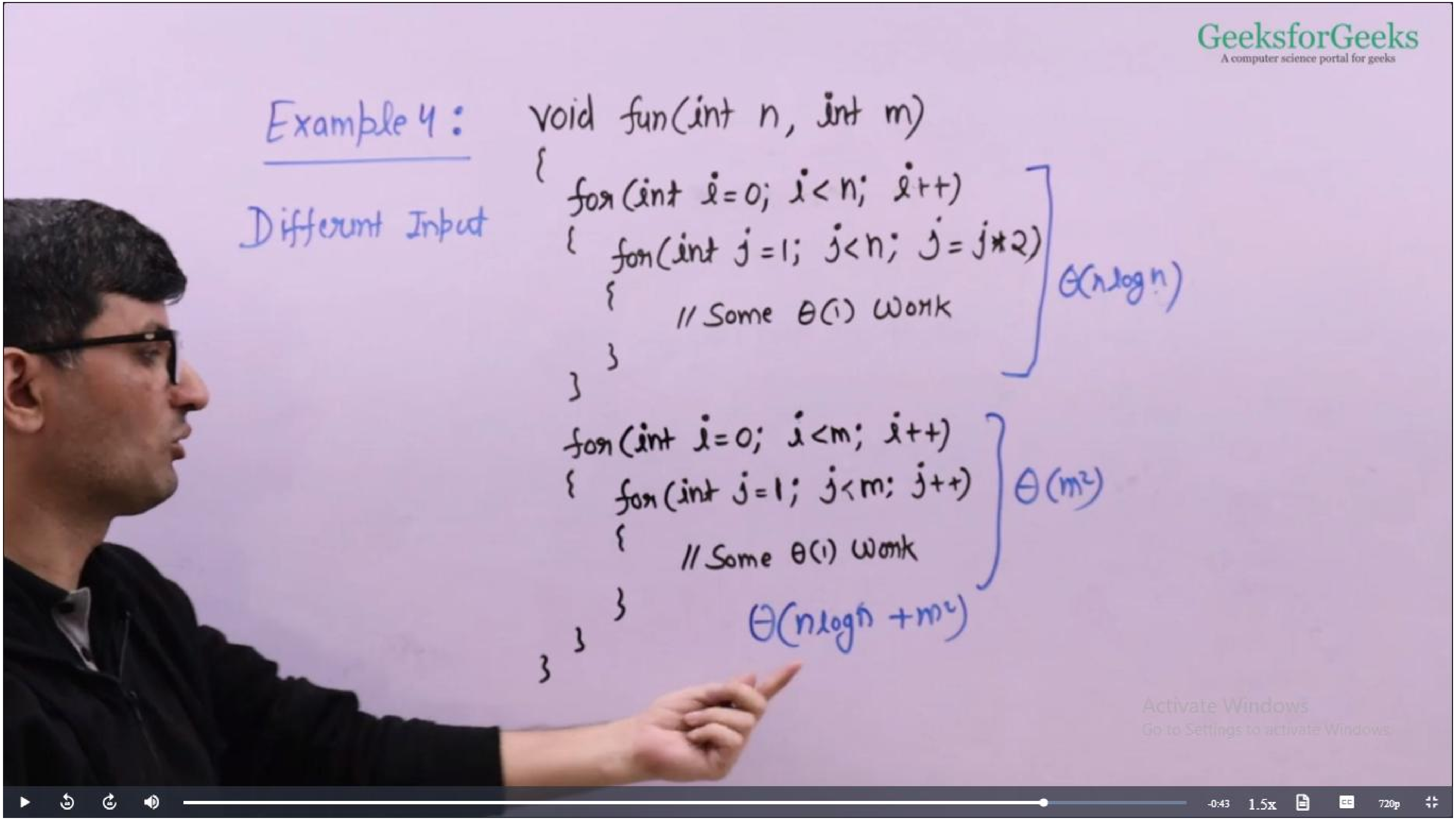

GeeksforGeeks
A computer science portal for geeks
for(int i=n; i>1; i=i/c)
{
//Some Θ(1) Work
}
Example 4 :
n/c⁰, n/c, n/c² ... n/c^(k-1)
n/c^(k-1) > 1
c^(k-1) < n
k-1 < log_c n
k < log_c n + 1
Θ(log_c n)
n = 33
c = 2
i = 33
i = 16
i = 8
i = 4
i = 2
n = 81
c = 3
i = 81
i = 27
i = 9
i = 3
Activate Windows
Go to Settings to activate Windows.
-4:12
1.5x
720p

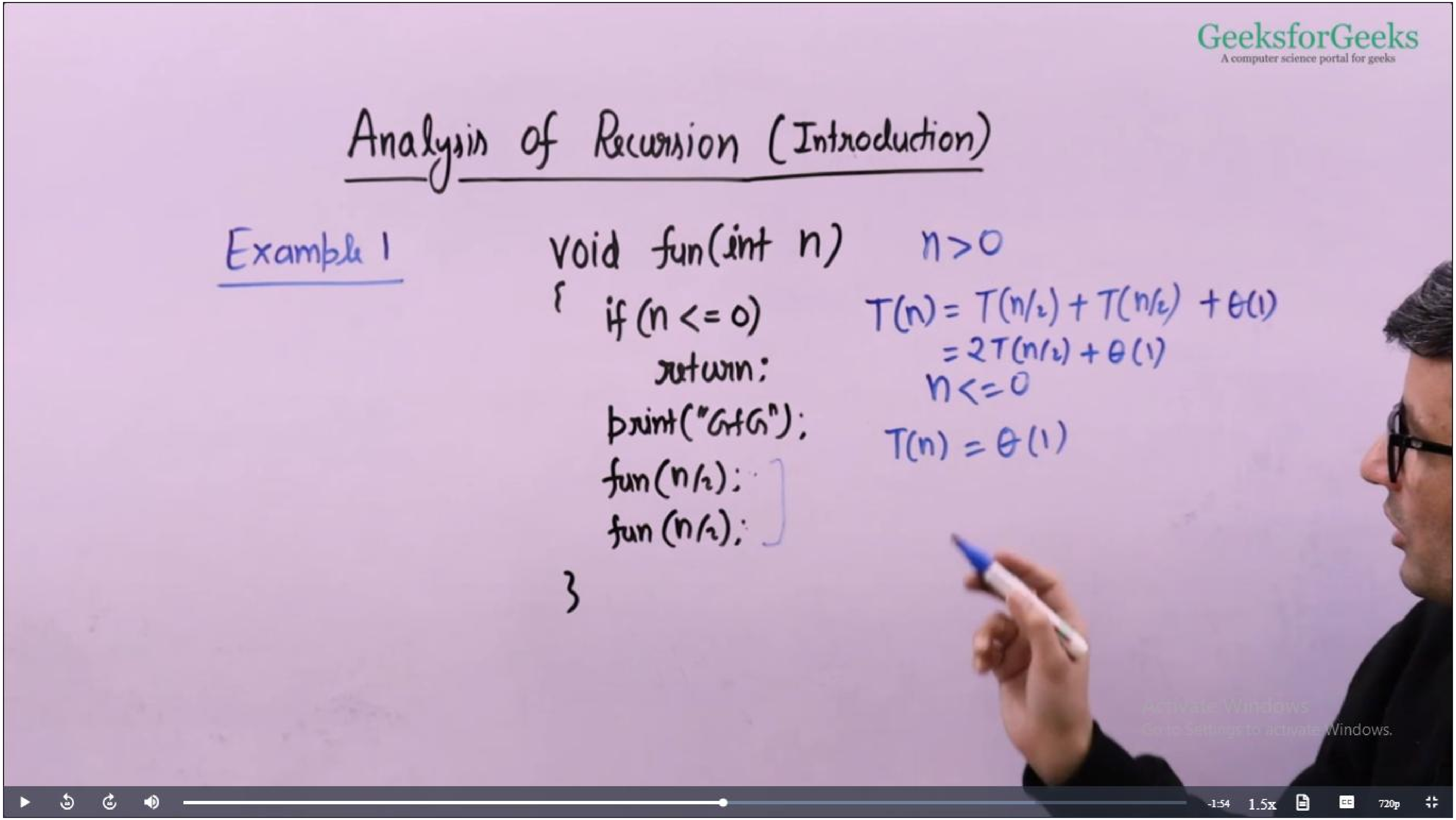

GeeksforGeeks
A computer science portal for geeks
Example 5
2, 2^c, (2^c)^c, ... ((2^c)^c)^c
2^(c^0), 2^(c^1), 2^(c^2), ... 2^(c^(k-1))
2^(c^(k-1)) < n
c^(k-1) < log2 n
k-1 < logc log2 n
k < logc log2 n + 1
Θ(log2 logc n)
for (int i = 2; i<n; i = pow(i, c))
{
// Some Θ(1) Work
}
n = 33
c = 2
i = 2
i = 4
i = 16
n=514
c=3
i = 2
i = 8
i = 512
2^3 = 8
8^3 = 512
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Analysis of Multiple
void fun(int n)
{
for(int i = 0; i<n; i++)
{ // Some θ(1) Work }
θ(n)
+
for(int i=1; i<n; i=i*2)
{ // Some θ(1) Work }
θ(log n) ×
+
for(int i=1; i<100; i++)
{ // Some θ(1) Work }
θ(1) ×
}
= θ(n)
Activate Windows
Go to Settings to activate Windows.
-3:40
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Example 2:
Nested loops
void fun(int n)
{
for (int i = 0; i < n; i++) → θ(n)
*
{
for (int j = 1; j < n; j = j * 2) → θ(log n)
{
// Some θ(1) work
}
θ(n log n)
}
}
-2:39
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Example 3:
Mixed Loops
void fun(int n)
{
for(int i=0; i<n; i++)
{ for(int j=1; j<n; j=j*2)
{
// Some θ(1) work
}
}
θ(nlogn)
+
for(int i=0; i<n; i++)
{ for(int j=1; j<n; j++)
{
// Some θ(1) work
}
}
}
θ(n²)
θ(n²)
Activate Windows
Go to Settings to activate Windows.
-1:50
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Example 4 :
Different Input
void fun(int n, int m)
{
for(int i=0; i<n; i++)
{ for(int j=1; j<n; j=j*2)
{
// Some θ(1) work
}
}
θ(n log n)
for(int i=0; i<m; i++)
{ for(int j=1; j<m; j++)
{
// Some θ(1) work
}
}
}
θ(m²)
θ(nlogn + m²)
Activate Windows
Go to Settings to activate Windows
-0:43
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Analysis of Recursion (Introduction)
Example 1
void fun(int n)
{
if (n <= 0)
return;
print("GfG");
fun(n/2);
fun(n/2);
}
n>0
T(n) = T(n/2) + T(n/2) + θ(1)
= 2T(n/2) + θ(1)
n<=0
T(n) = θ(1)
-1:54
1.5x
720p

Analysis of Recursion (Introduction)

Example 1

```
void fun(int n)
{
    if (n <= 0)
        return;
    print("GfG");
    fun(n/2);
    fun(n/2);
}
```

$n > 0$

$$T(n) = T(n/2) + T(n/2) + \theta(1)$$

$$\boxed{\begin{gathered} T(n) = 2T(n/2) + \theta(1) \\ T(0) = \theta(1) \end{gathered}}$$

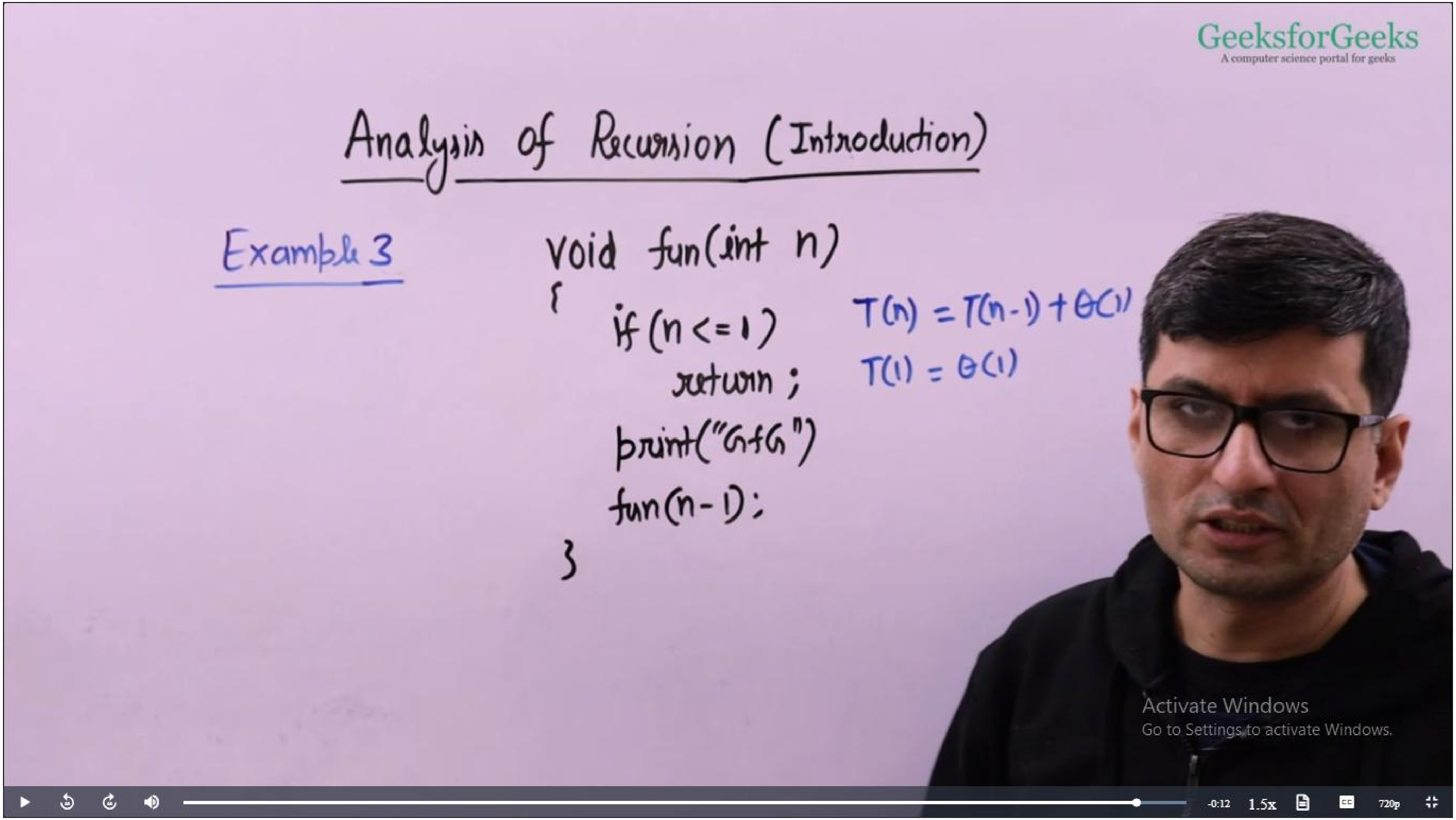
Analysis of Recursion (Introduction)

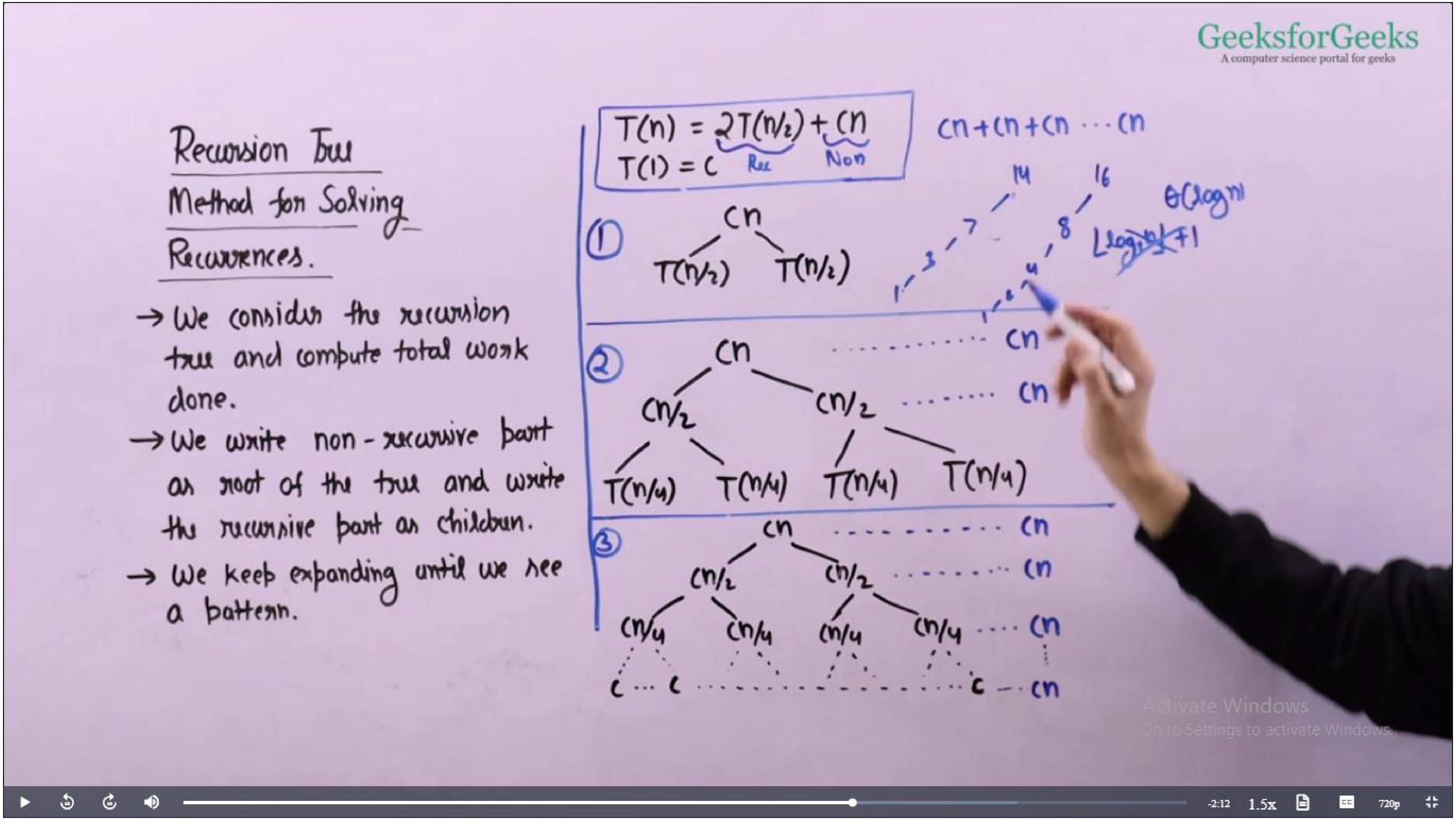
Example 2

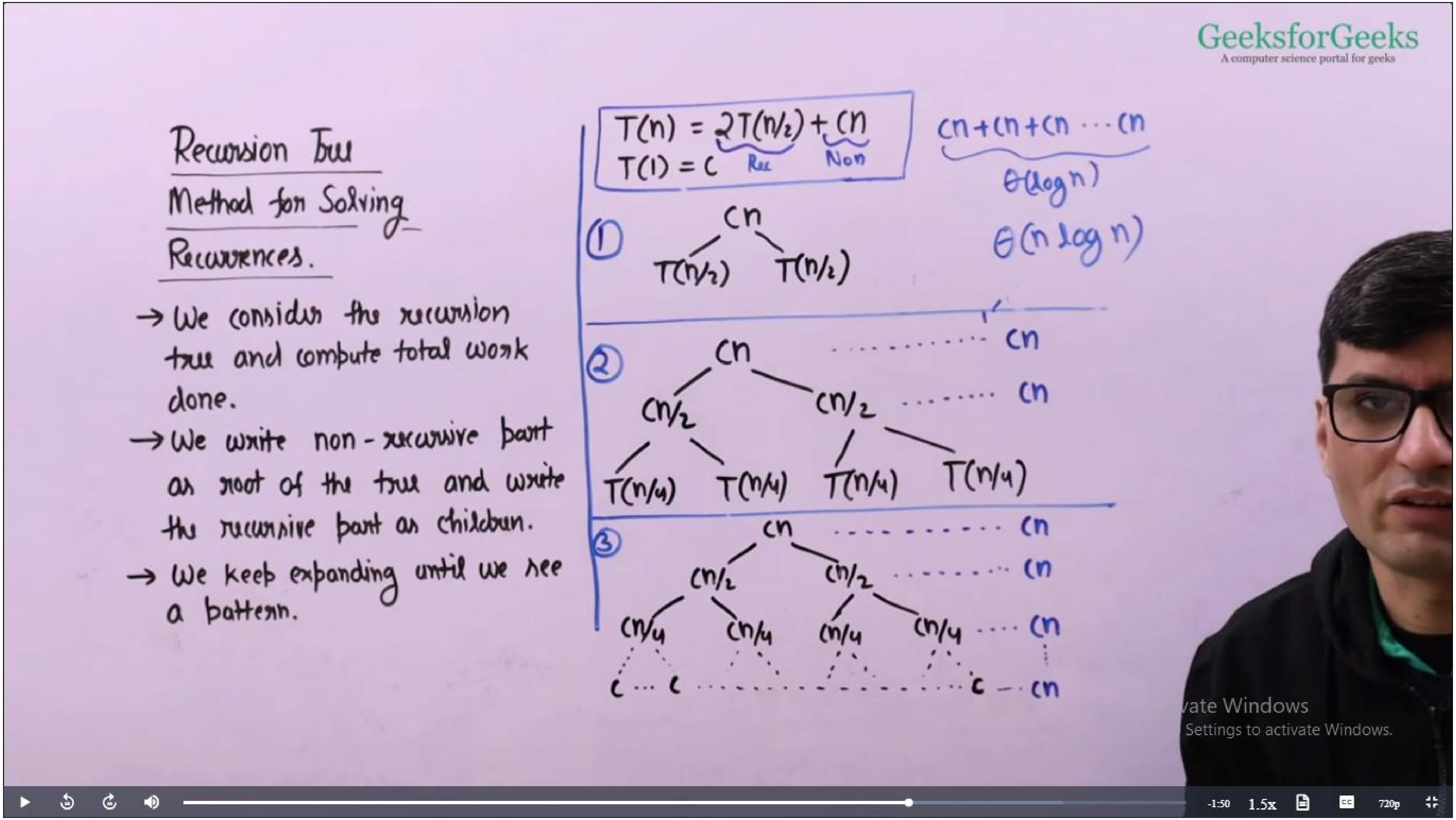
```
void fun(int n)
{
    if (n <= 0)
        return;
    for (int i = 0; i < n; i++)
        print("GfG");
    fun(n/2);
    fun(n/3);
}
```

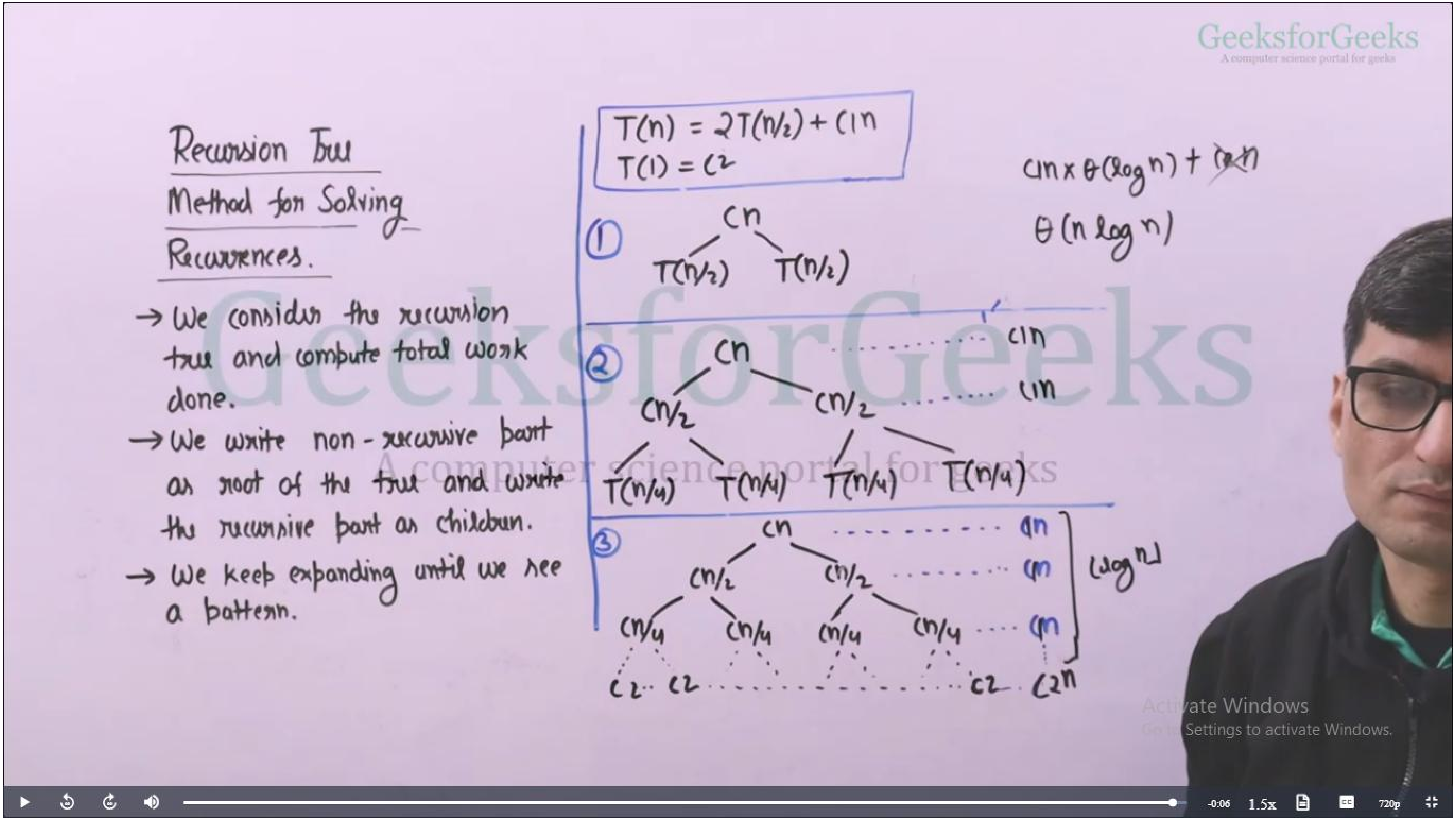
$n > 0$

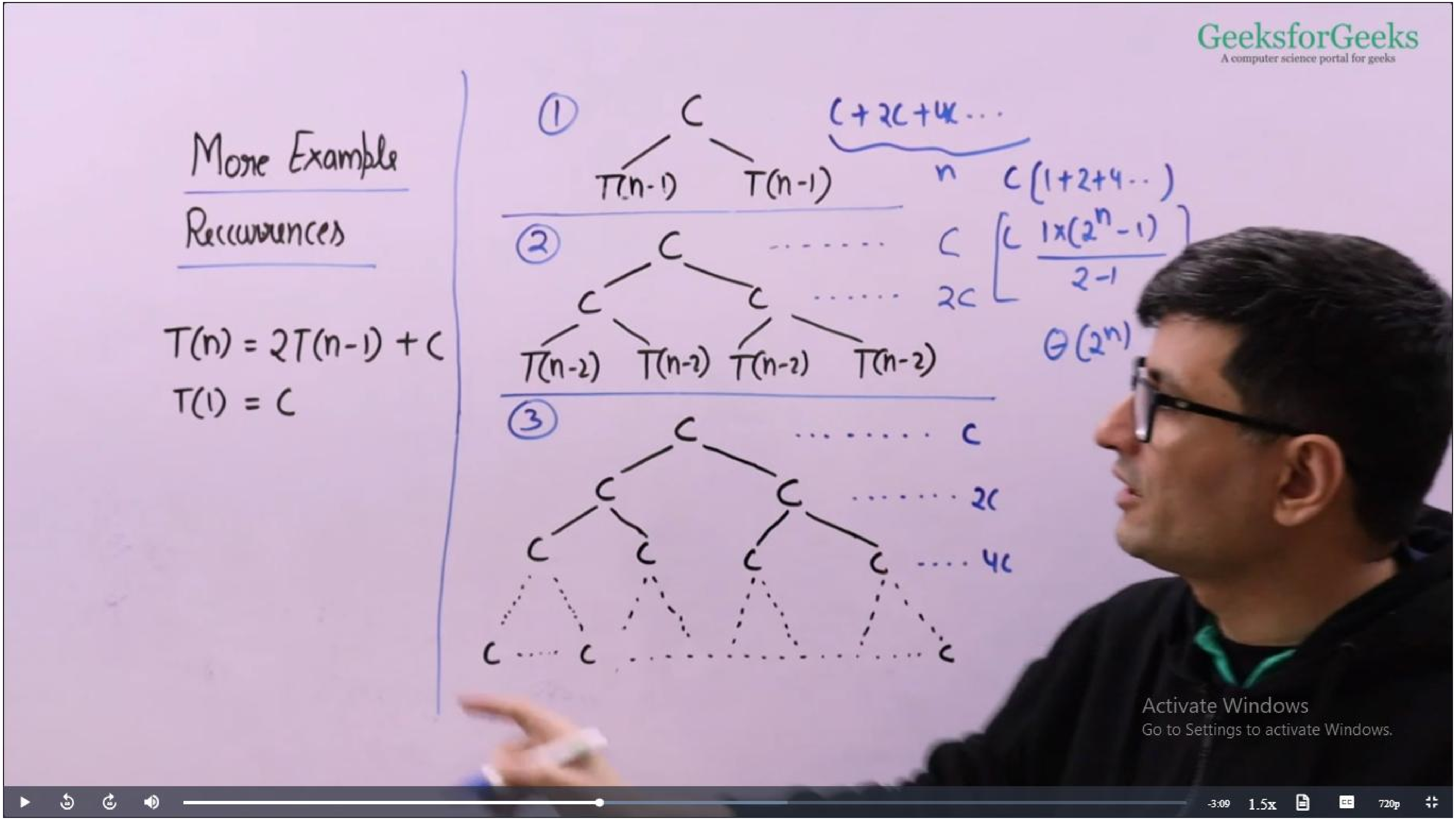
$$\boxed{\begin{aligned} T(n) &= T(n/2) + T(n/3) + \theta(n) \\ T(0) &= \theta(1) \end{aligned}}$$

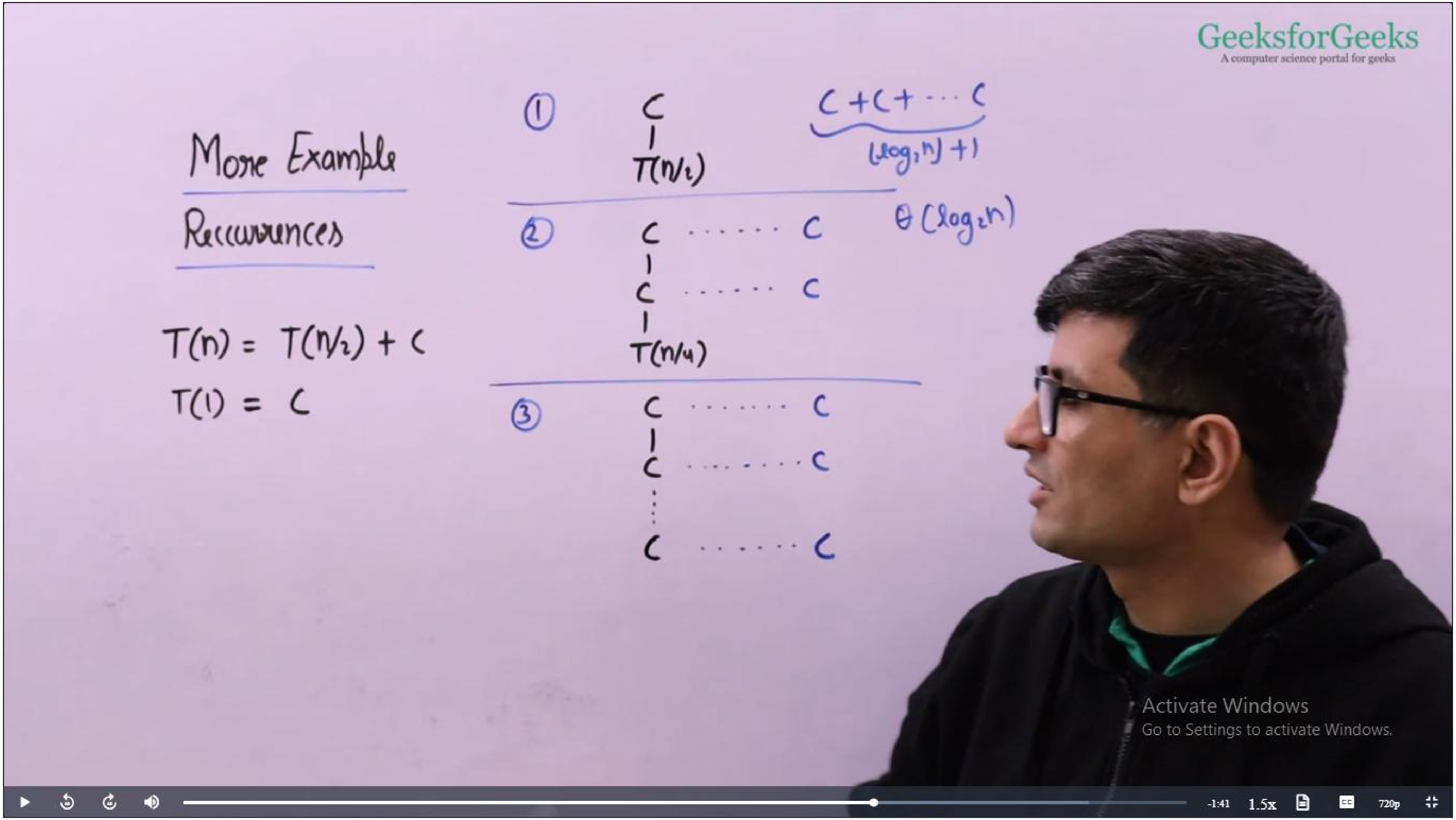

GeeksforGeeks
A computer science portal for geeks
Analysis of Recursion (Introduction)
Example 3
void fun(int n)
{
if (n <= 1)
return;
print("GfG")
fun(n-1);
}
T(n) = T(n-1) + Θ(1)
T(1) = Θ(1)
Activate Windows
Go to Settings to activate Windows.
-0:12
1.5x
720p

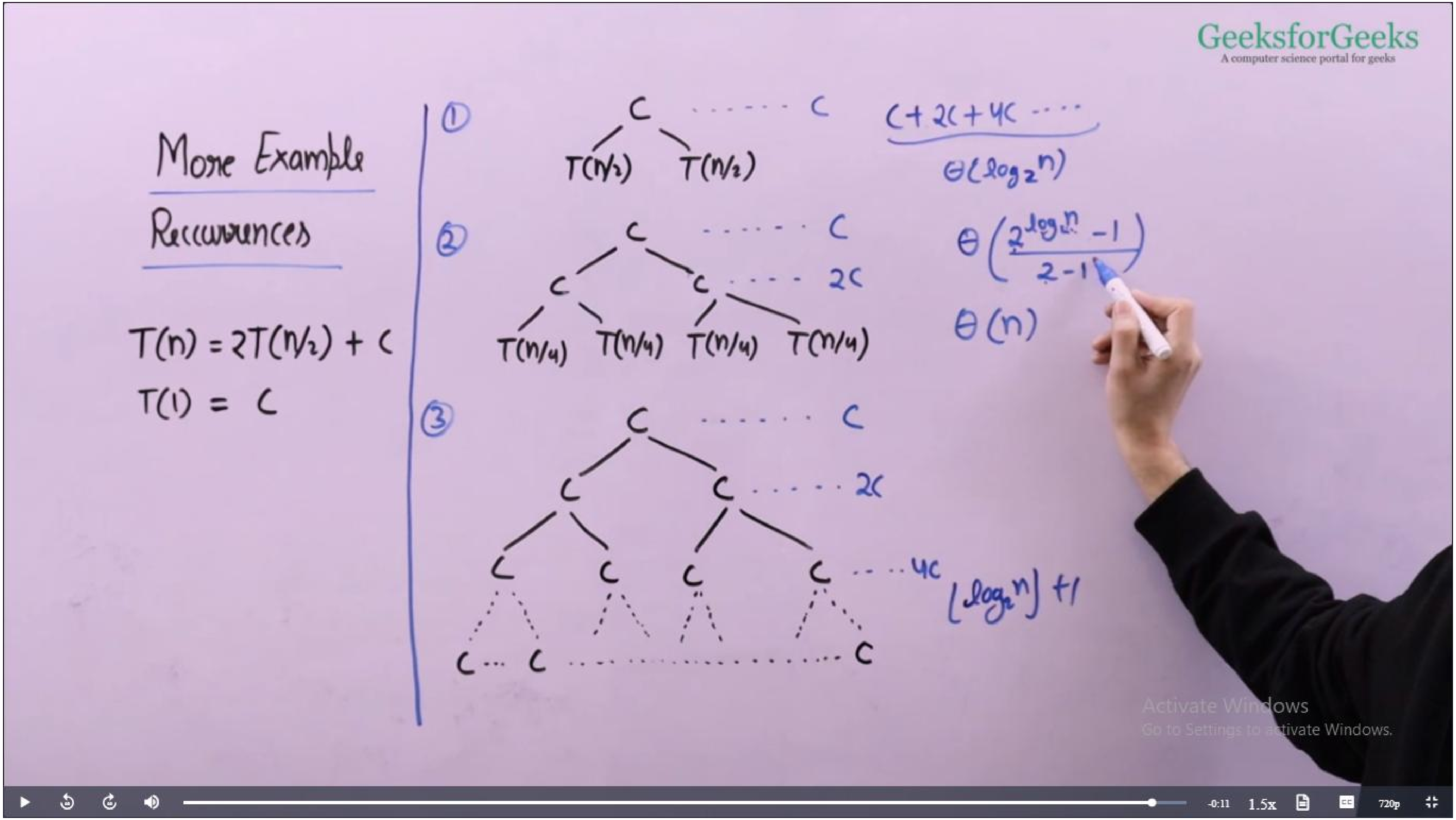

GeeksforGeeks
A computer science portal for geeks
Recursion Tree
Method for Solving
Recurrences.
→ We consider the recursion tree and compute total work done.
→ We write non-recursive part as root of the tree and write the recursive part as children.
→ We keep expanding until we see a pattern.
T(n) = 2T(n/2) + cn
T(1) = C
Rec
Non
cn + cn + cn ··· cn
θ(logn)
⌊log₂n⌋+1
①
cn
T(n/2) T(n/2)
②
cn
cn/2 cn/2
T(n/4) T(n/4) T(n/4) T(n/4)
③
cn
cn/2 cn/2
cn/4 cn/4 cn/4 cn/4
c ··· c ··· c
cn
cn
cn
cn
cn
cn
cn
Activate Windows
Go to Settings to activate Windows.
-2:12
1.5x
720p

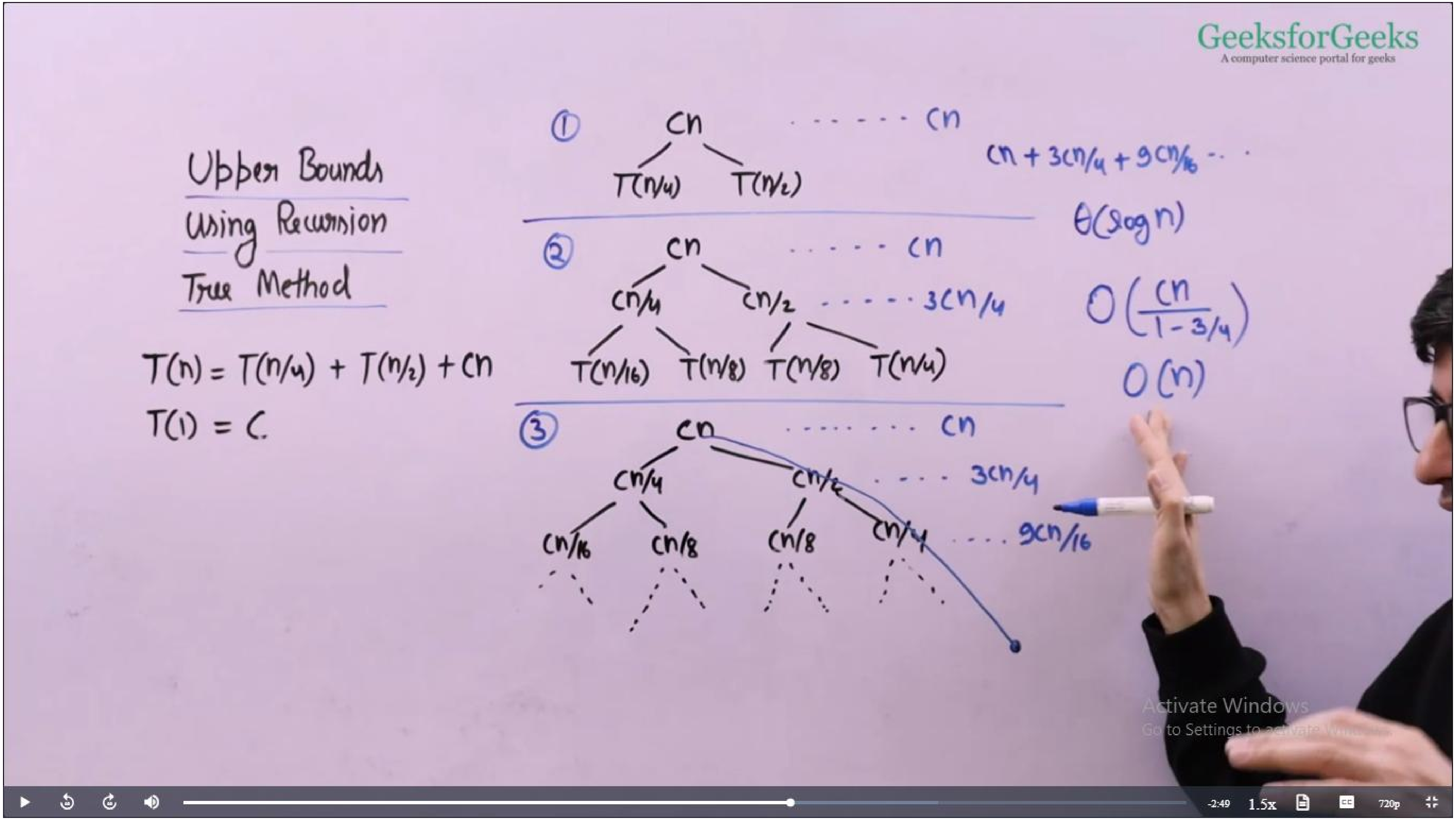

GeeksforGeeks
A computer science portal for geeks
Recursion Tree
Method for Solving
Recurrences.
→ We consider the recursion tree and compute total work done.
→ We write non-recursive part as root of the tree and write the recursive part as children.
→ We keep expanding until we see a pattern.
T(n) = 2T(n/2) + cn
Rec
Non
T(1) = c
cn + cn + cn ⋯ cn
θ(log n)
θ(n log n)
①
cn
T(n/2)
T(n/2)
②
cn
cn/2
cn/2
T(n/4)
T(n/4)
T(n/4)
T(n/4)
cn
cn
③
cn
cn/2
cn/2
cn/4
cn/4
cn/4
cn/4
c … c
c
cn
cn
cn
cn
1.5x
720p

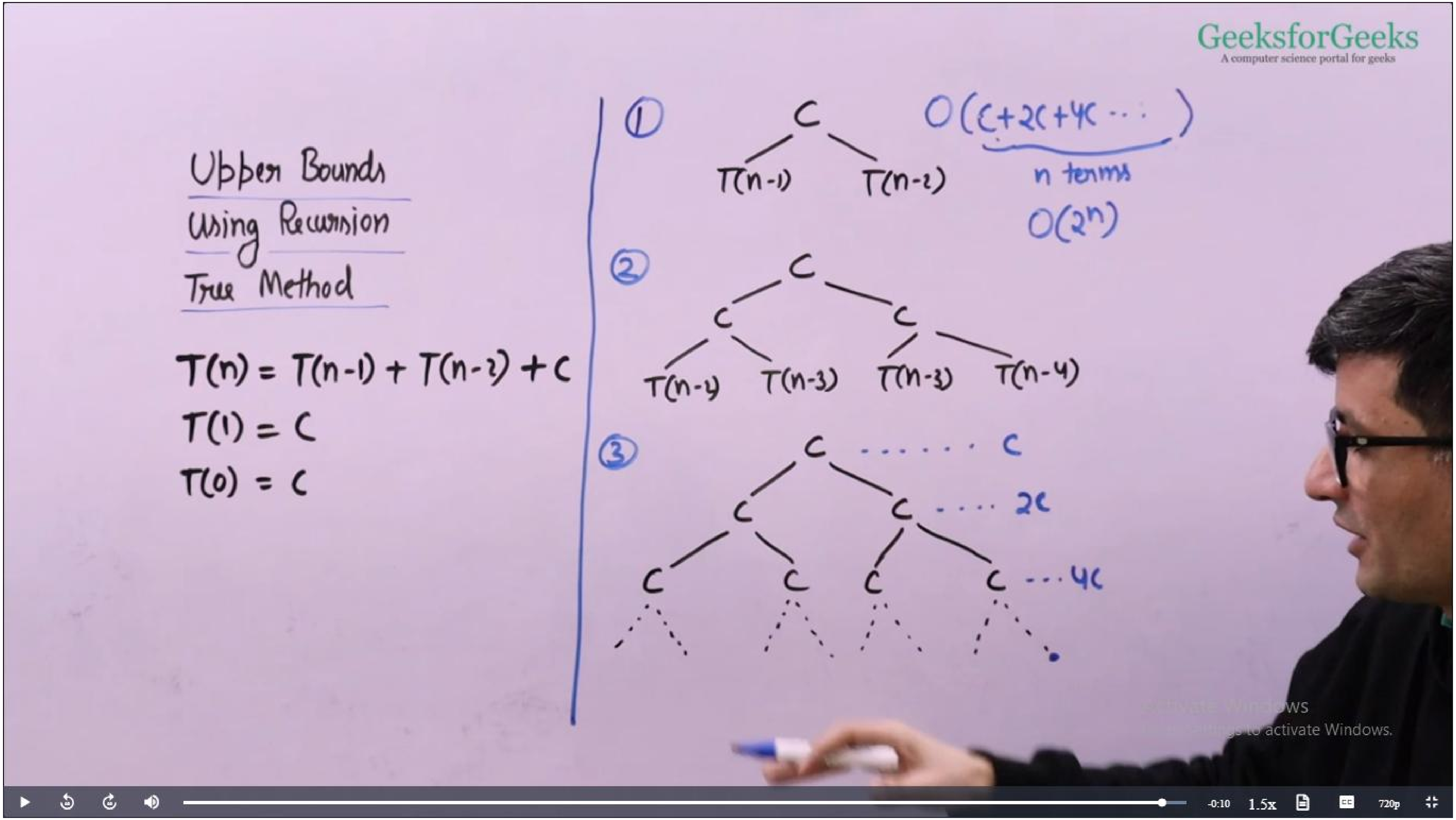

GeeksforGeeks
A computer science portal for geeks
Recursion Tree
Method for Solving
Recurrences.
→ We consider the recursion tree and compute total work done.
→ We write non-recursive part as root of the tree and write the recursive part as children.
→ We keep expanding until we see a pattern.
T(n) = 2T(n/2) + c1n
T(1) = c2
① cn
T(n/2) T(n/2)
② cn c1n
cn/2 cn/2 .... c1n
T(n/4) T(n/4) T(n/4) T(n/4)
③ cn c1n
cn/2 cn/2 .... c1n
cn/4 cn/4 cn/4 cn/4 .... c1n
c2 .. c2 .... c2 . c2n
log n
c1n × θ(log n) + c2n
θ(n log n)
Activate Windows
Go to Settings to activate Windows.
-0:06 1.5x 720p


GeeksforGeeks
A computer science portal for geeks
More Example
Reccurences
T(n) = 2T(n-1) + C
T(1) = C
① C
T(n-1) T(n-1)
C+2C+4C...
n
C(1+2+4..)
② C
C C
T(n-2) T(n-2) T(n-2) T(n-2)
C
2C
C [C 1x(2^n - 1) / 2-1]
Θ(2^n)
③ C
C C
C C C C
C C C
C
2C
4C
Activate Windows
Go to Settings to activate Windows.
-3:09
1.5x
720p

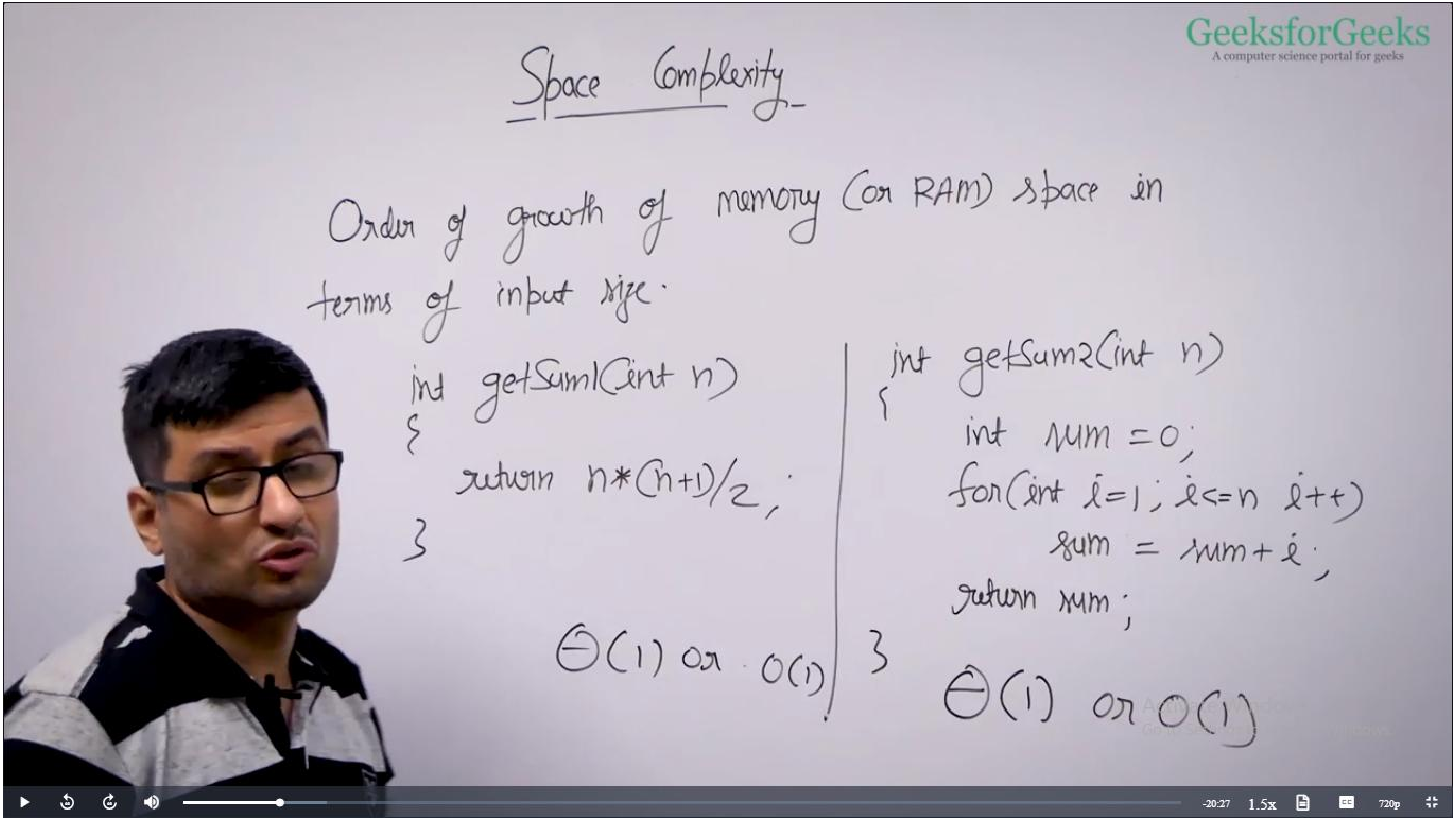

GeeksforGeeks
A computer science portal for geeks
More Example
Reccurrences
T(n) = T(n/2) + c
T(1) = c
① c
T(n/2)
c + c + ⋯ c
⌊log₂n⌋ + 1
② c ······· c
c ······· c
T(n/4)
Θ(log₂n)
③ c ······· c
c ······· c
c ······· c
Activate Windows
Go to Settings to activate Windows.
-1:41
1.5x
720p

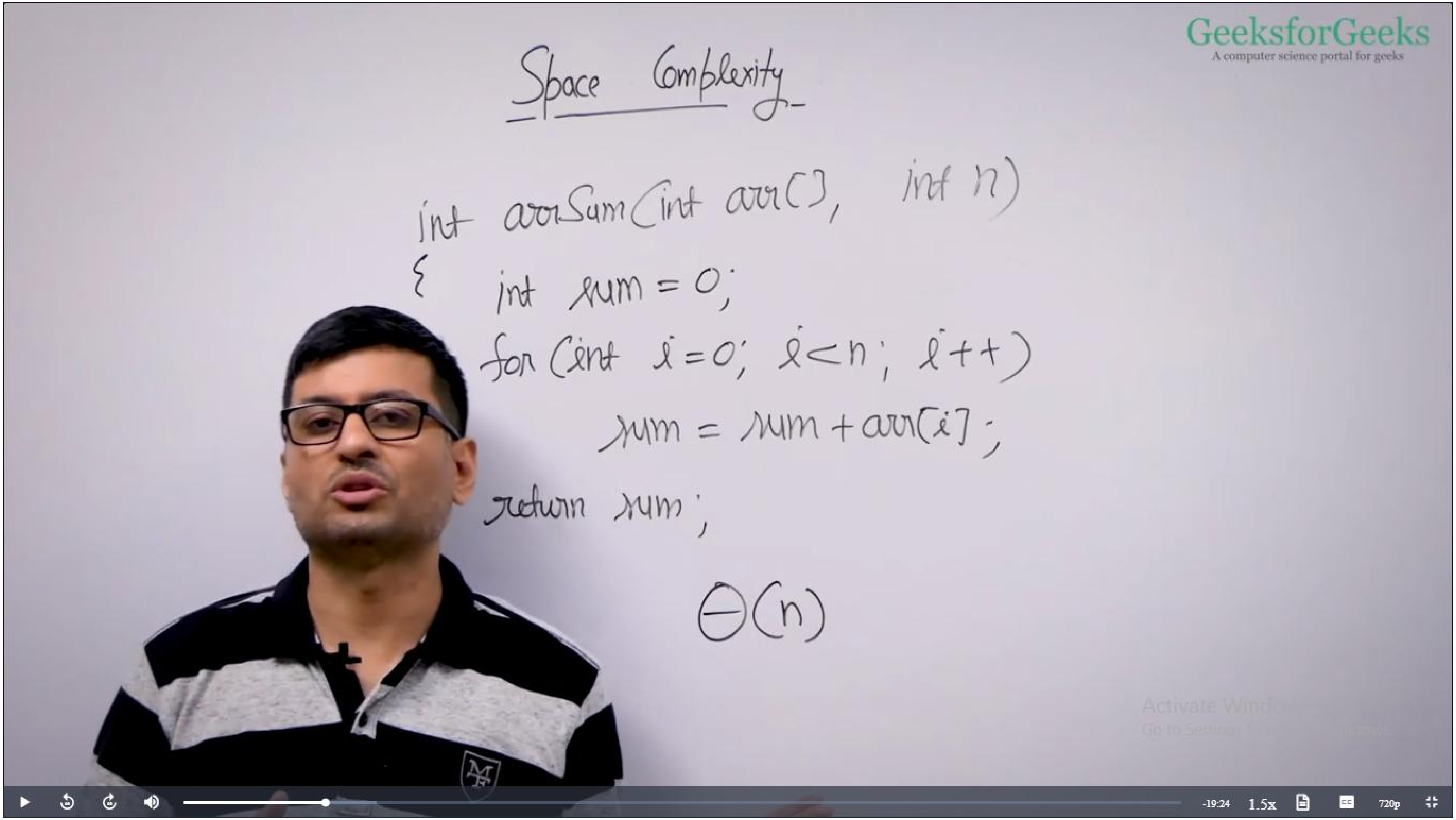

GeeksforGeeks
A computer science portal for geeks
More Example
Recurrences
T(n) = 2T(n/2) + c
T(1) = c
①
c c
T(n/2) T(n/2)
②
c c
c c 2c
T(n/4) T(n/4) T(n/4) T(n/4)
③
c c
c c 2c
c c c c ... 4c
c ... c c
c + 2c + 4c ....
Θ(log₂n)
Θ((2^(log₂n) − 1)/(2 − 1))
Θ(n)
⌊log₂n⌋ + 1
Activate Windows
Go to Settings to activate Windows.
-0:11
1.5x
720p

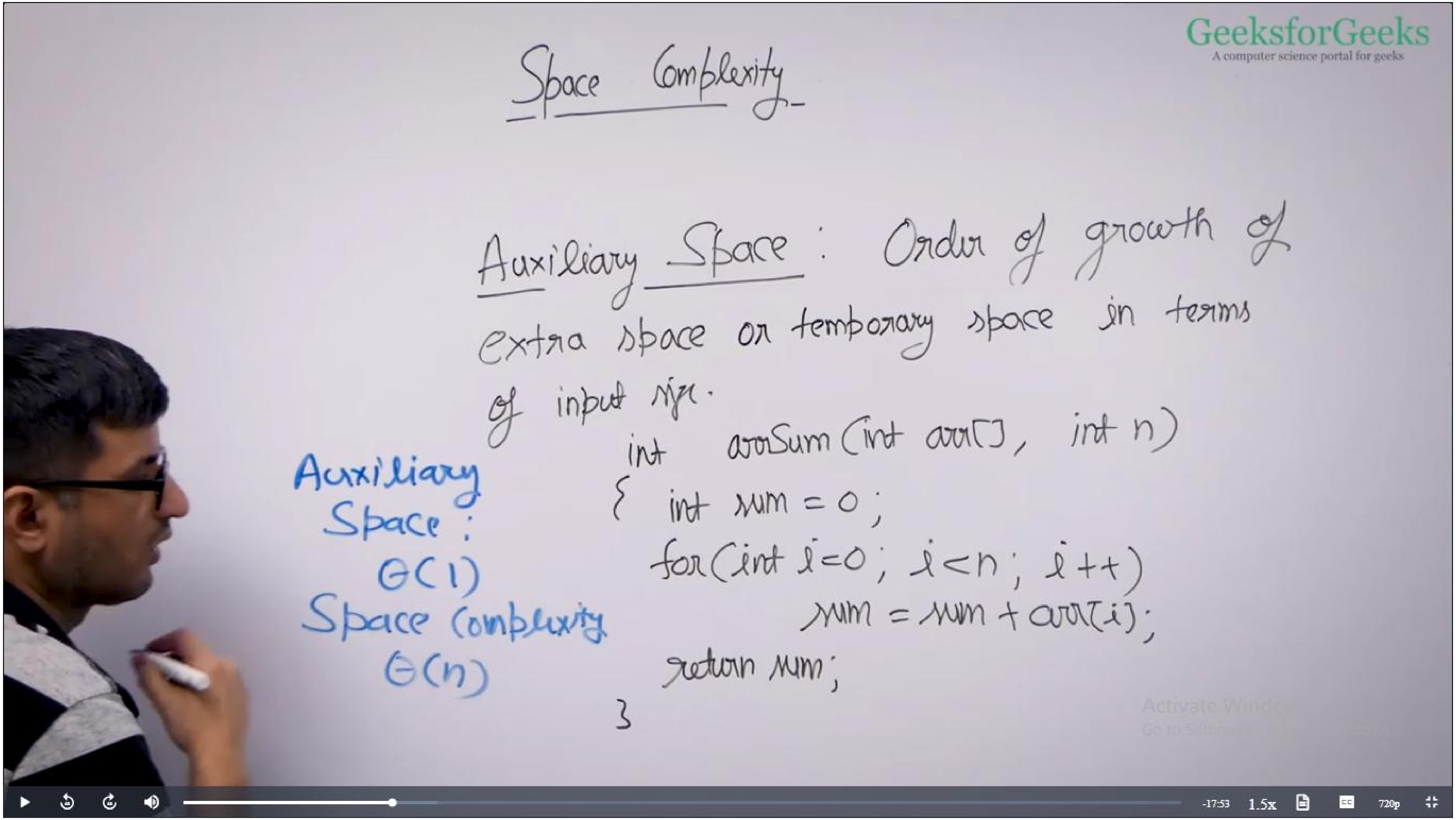

GeeksforGeeks
A computer science portal for geeks
Upper Bounds
Using Recursion
Tree Method
T(n) = T(n/4) + T(n/2) + cn
T(1) = C
1 cn ...... cn
T(n/4) T(n/2)
cn + 3cn/4 + 9cn/16 ...
Θ(log n)
2 cn ..... cn
cn/4 cn/2 ..... 3cn/4
T(n/16) T(n/8) T(n/8) T(n/4)
O(cn / (1 - 3/4))
O(n)
3 cn .... cn
cn/4 cn/2 .... 3cn/4
cn/16 cn/8 cn/8 cn/4 .... 9cn/16
-2:49
1.5x
720p

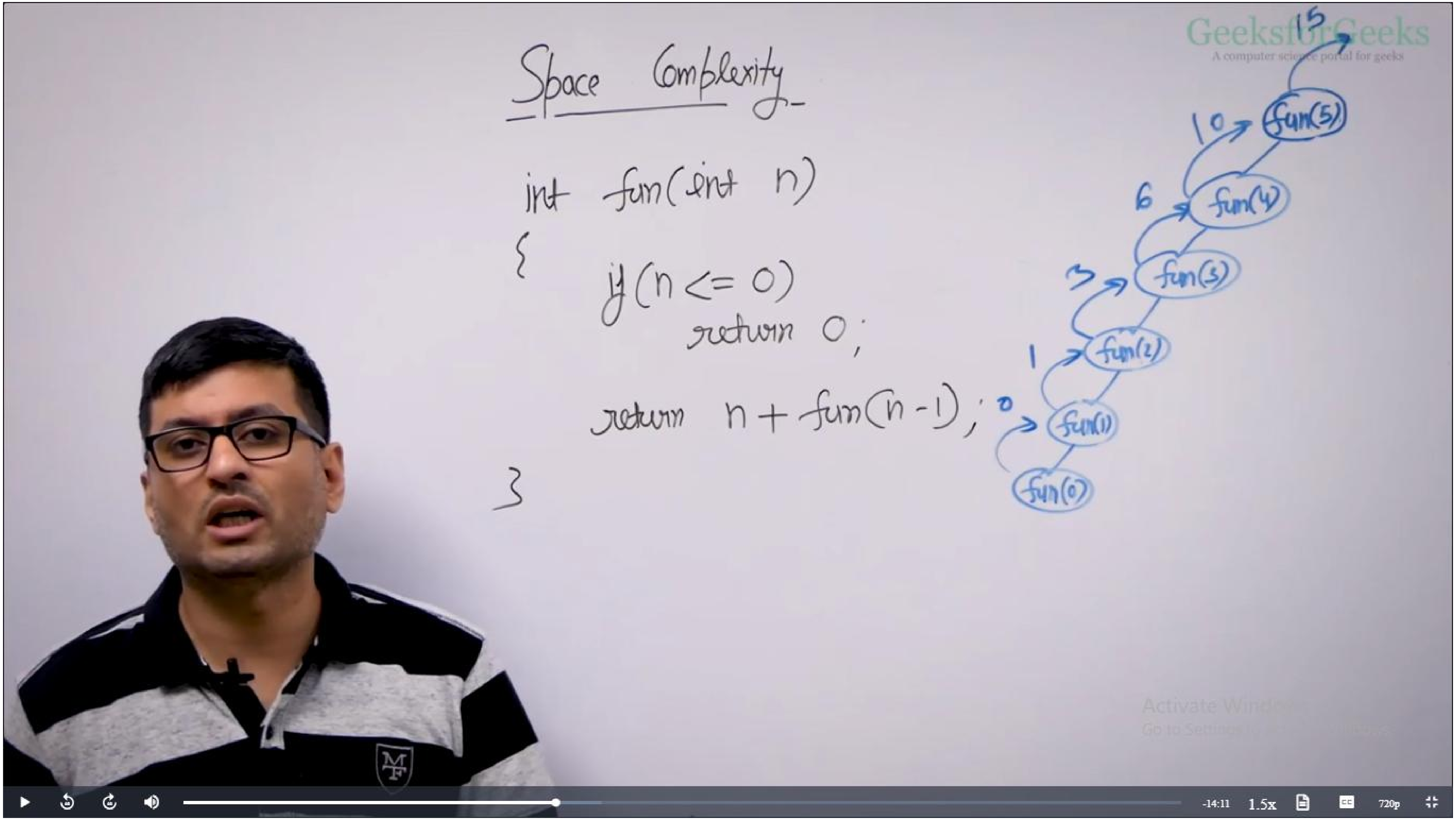


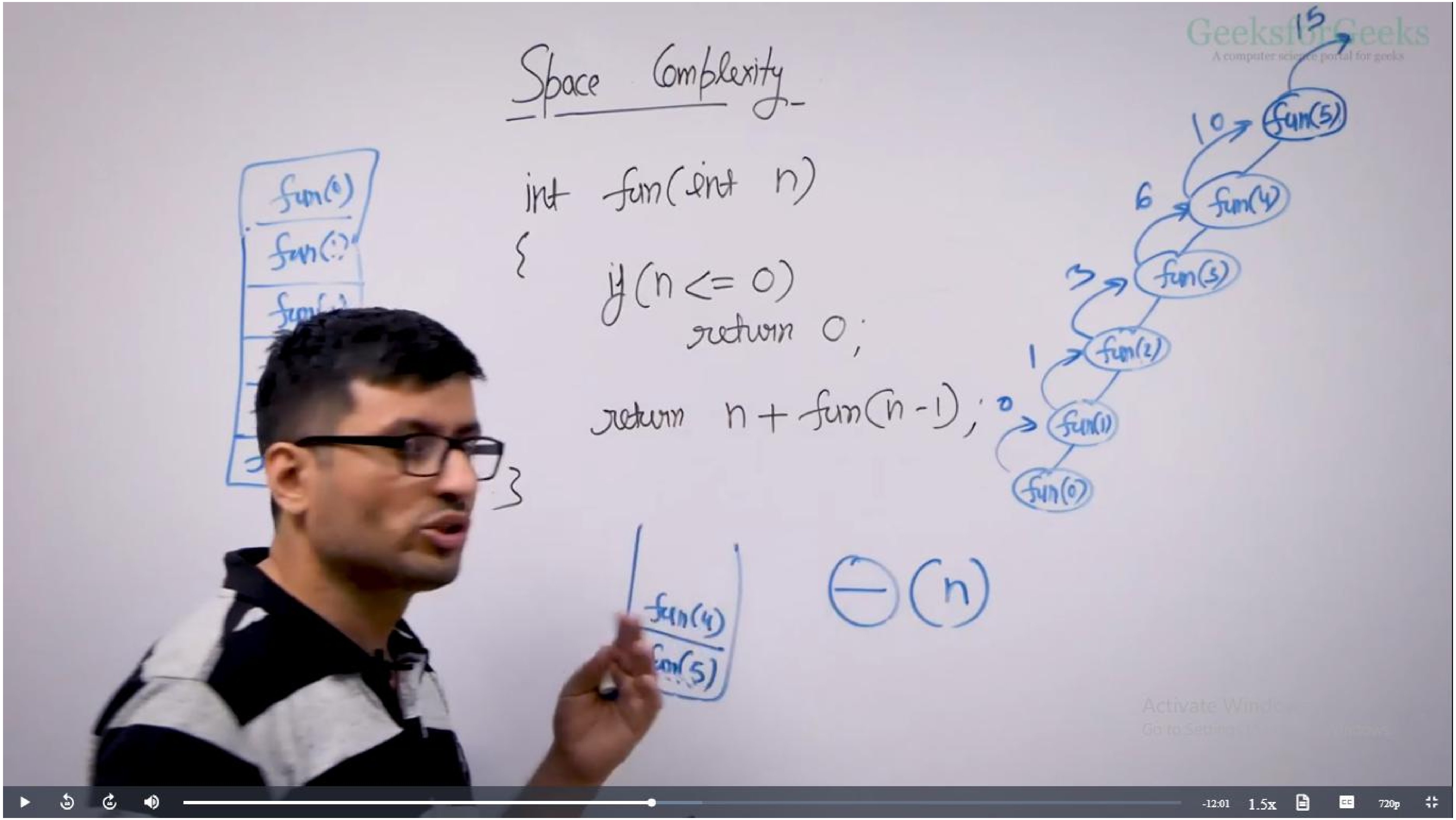
GeeksforGeeks
A computer science portal for geeks
Upper Bounds
Using Recursion
Tree Method
T(n) = T(n-1) + T(n-2) + c
T(1) = c
T(0) = c
①
c
T(n-1) T(n-2)
O(c+2c+4c ⋯)
n terms
O(2^n)
②
c
c c
T(n-2) T(n-3) T(n-3) T(n-4)
③
c ⋯⋯ c
c c ⋯ 2c
c c c c ⋯ 4c
-0:10
1.5x
720p

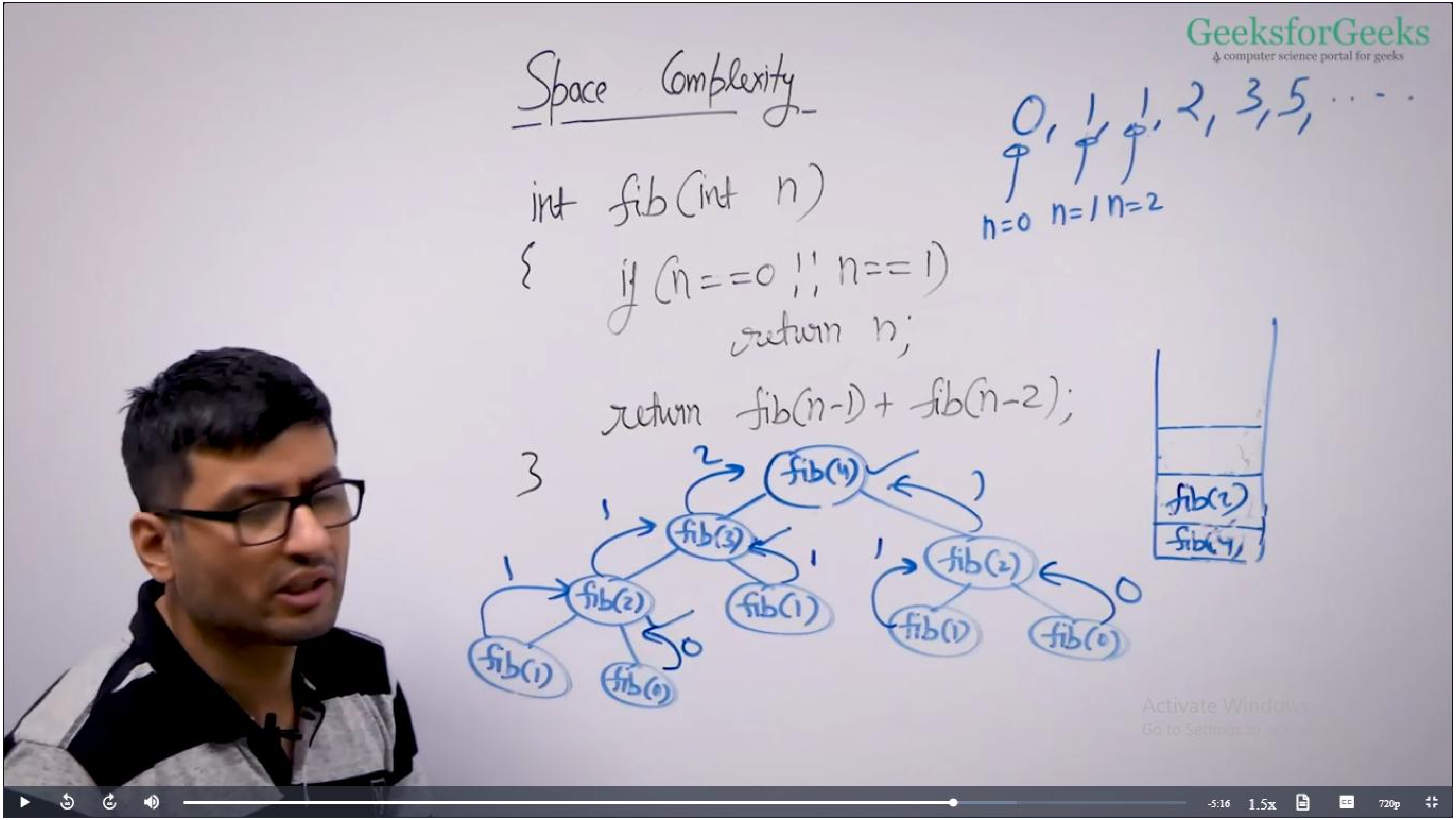

GeeksforGeeks
A computer science portal for geeks
Space Complexity
Order of growth of memory (or RAM) space in terms of input size.
int getSum1(int n)
{
return n*(n+1)/2;
}
Θ(1) or O(1)
int getSum2(int n)
{
int sum = 0;
for(int i=1; i<=n i++)
sum = sum+i;
return sum;
}
Θ(1) or O(1)
-20:27
1.5x
720p

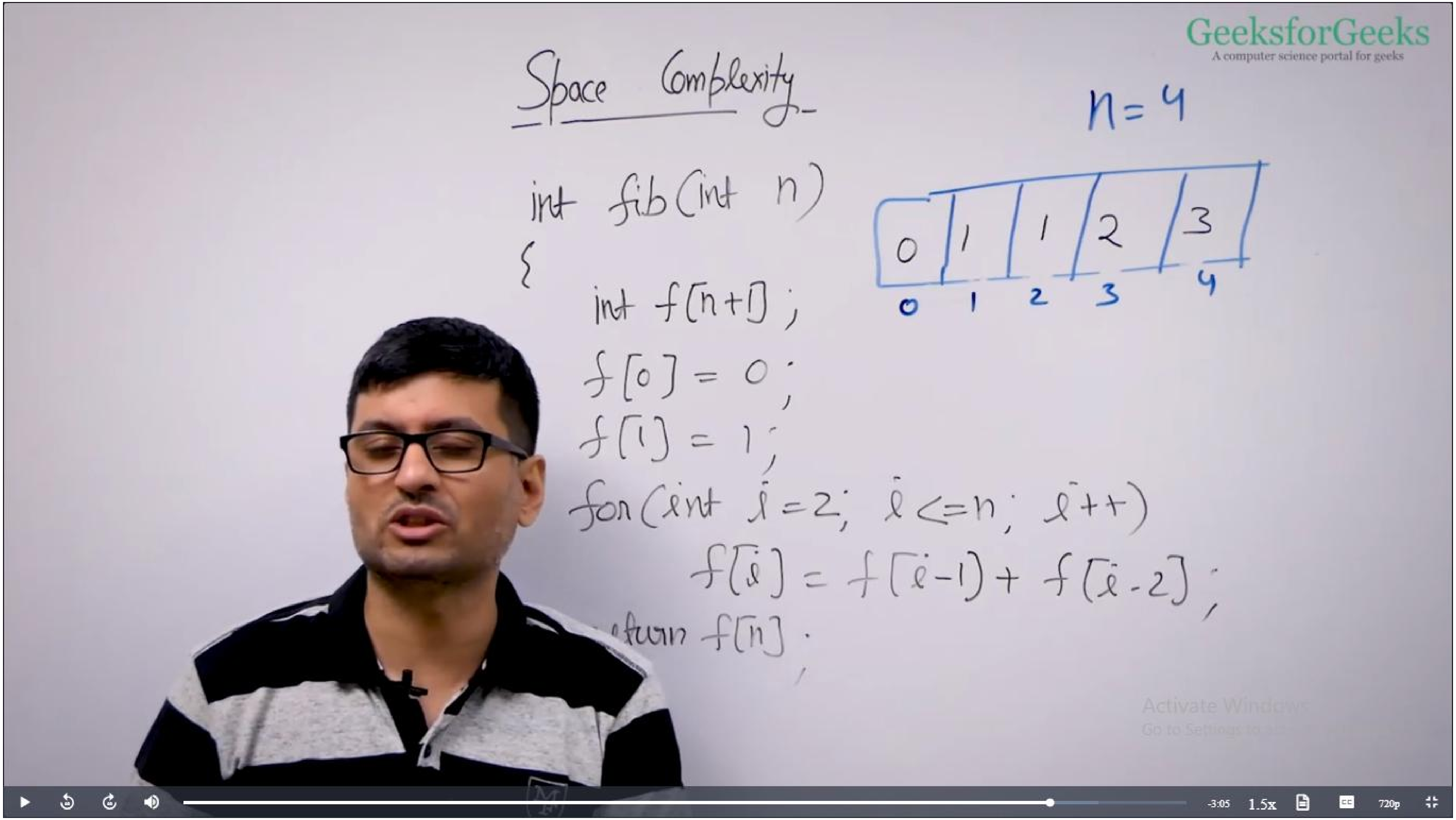

GeeksforGeeks
A computer science portal for geeks
Space Complexity
int arrSum(int arr[], int n)
{
int sum = 0;
for (int i=0; i<n; i++)
sum = sum + arr[i];
return sum;
Θ(n)
-19:24
1.5x
720p

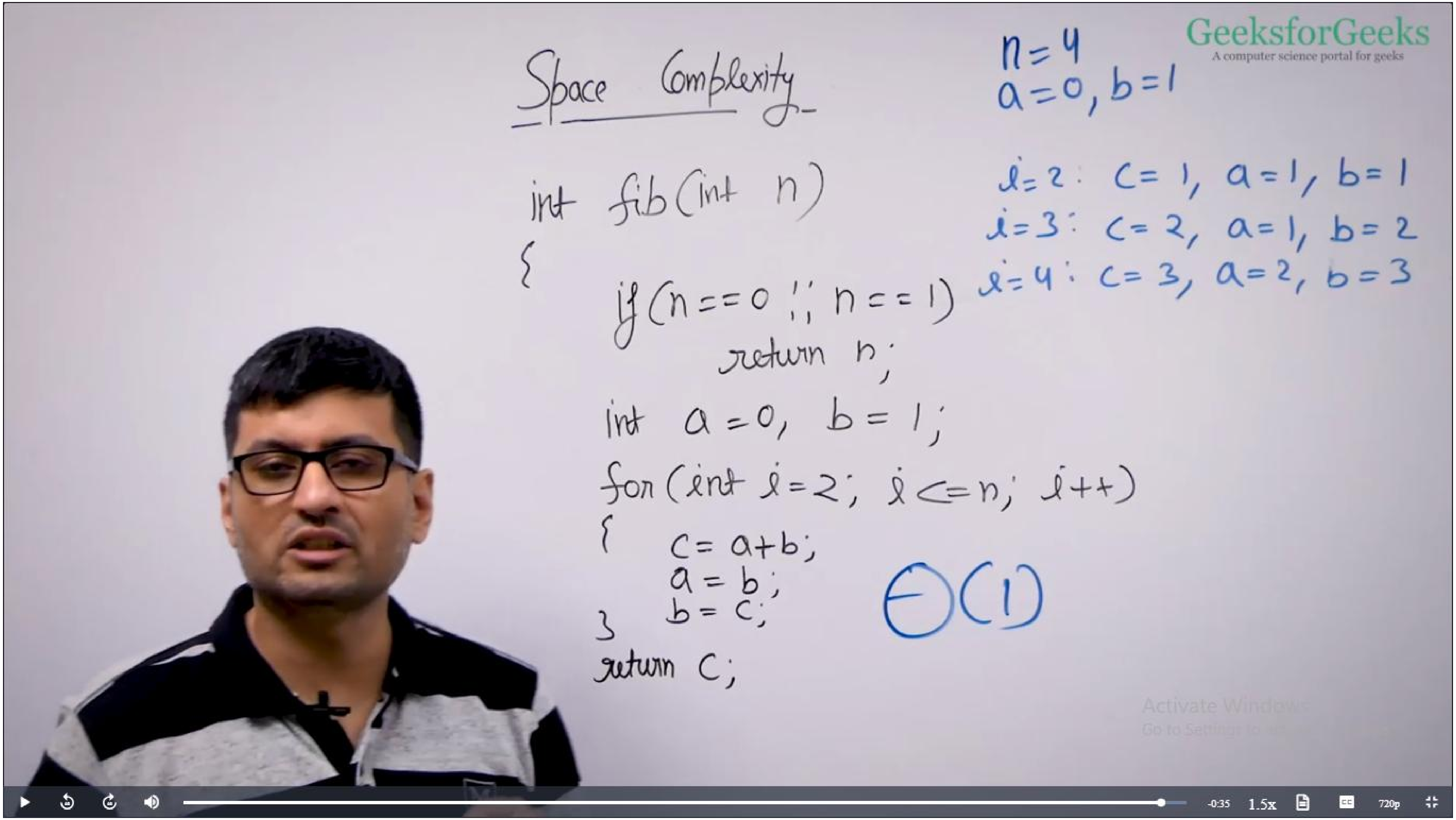

GeeksforGeeks
A computer science portal for geeks
Space Complexity
Auxiliary Space: Order of growth of extra space or temporary space in terms of input size.
int arrSum(int arr[], int n)
{ int sum = 0;
for(int i=0; i<n; i++)
sum = sum + arr[i];
return sum;
}
Auxiliary Space: Θ(1)
Space Complexity Θ(n)
-17:53
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Space Complexity
int fun(int n)
{
if(n <= 0)
return 0;
return n + fun(n-1);
}
fun(5)
fun(4)
fun(3)
fun(2)
fun(1)
fun(0)
15
10
6
3
1
0
Activate Windows
-14:11
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Space Complexity
int fun(int n)
{
if(n <= 0)
return 0;
return n + fun(n-1);
}
fun(5)
fun(4)
fun(3)
fun(2)
fun(1)
fun(0)
15
10
6
3
1
0
Θ(n)
-12:01
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Space Complexity
int fib(int n)
{
if (n==0 || n==1)
return n;
return fib(n-1) + fib(n-2);
3
0, 1, 1, 2, 3, 5, ...
n=0 n=1 n=2
fib(4)
fib(3)
fib(2)
fib(1)
fib(0)
fib(2)
fib(1)
fib(0)
fib(1)
fib(2)
fib(4)
-5:16
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Space Complexity
n = 4
int fib(int n)
{
int f[n+1];
f[0] = 0;
f[1] = 1;
for(int i=2; i<=n; i++)
f[i] = f[i-1] + f[i-2];
turn f[n];
0 1 1 2 3
0 1 2 3 4
Activate Windows
-3:05
1.5x
720p


GeeksforGeeks
A computer science portal for geeks
Space Complexity
int fib(int n)
{
if(n==0 || n==1)
return n;
int a=0, b=1;
for(int i=2; i<=n; i++)
{
c=a+b;
a=b;
b=c;
}
return c;
Θ(1)
n=4
a=0, b=1
i=2: c=1, a=1, b=1
i=3: c=2, a=1, b=2
i=4: c=3, a=2, b=3
-0:35
1.5x
720p