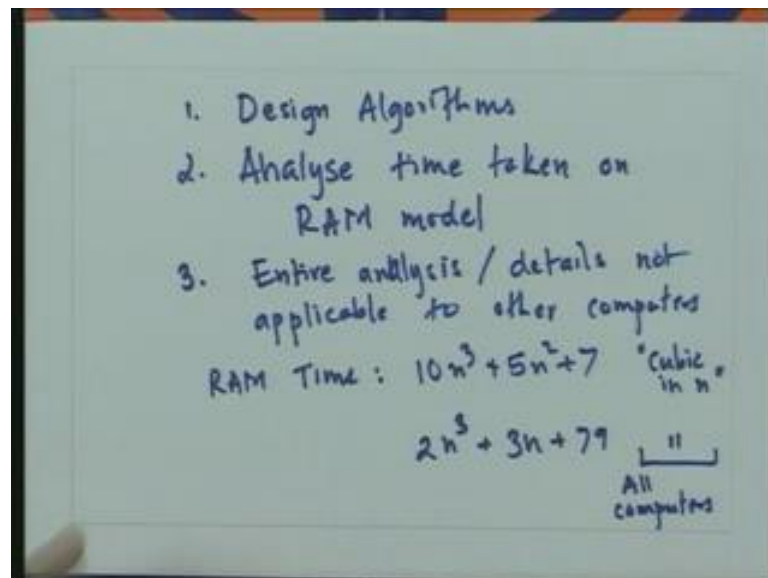


Design and Analysis of Algorithms
Prof. Abhiram Ranade
Department of Computer Science Engineering
Indian Institute of Technology, Bombay

Lecture - 4
Asymptotic Notation

Welcome to the course on Design and Analysis of Algorithms. Our topic today is Asymptotic Notation. Let me begin by setting down this topic, in the context of our overall course goals. We said last time that one of the main course goals was to well first design algorithms.

(Refer Slide Time: 01:10)



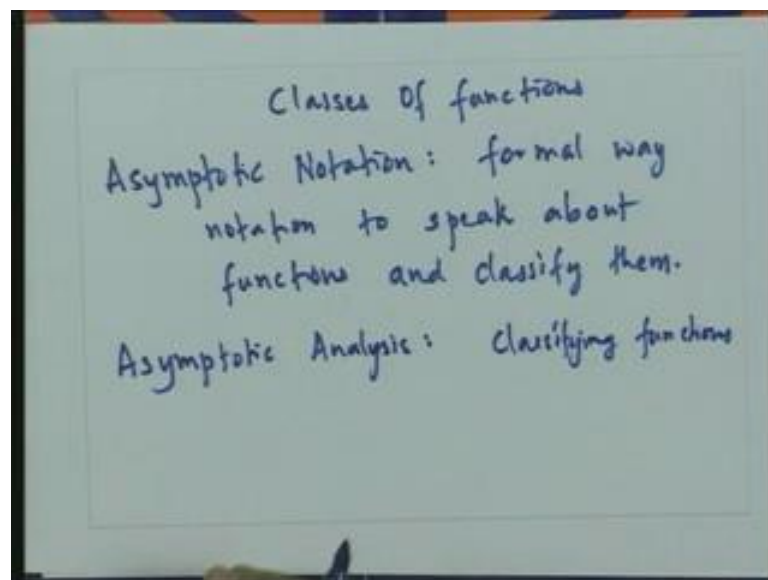
And then, we want to analyze their time. Analyze the time taken on the RAM model, which we defined. We also said that the results of the analysis are not directly applicable. So, there is some care needed in understanding, how to interpret the time taken on the RAM model? And how to use that predict, what happens on other computers. So, for example, so in all this we need some level of, we need to be a little bit imprecise.

We need to throw away some details of our analysis, in order to predict what happens on other computers. So, the entire analysis or what I really mean the entire detailed analysis not applicable to other computers. We said that suppose the time taken in the RAM was something like say $10n^3 + 5n^2 + 7$. Then, all that we can say for other computers is that the time is going to be cubic in n .

However, the same conclusion would be arrived at say if the time was $2n^3 + 3n + 79$. Even here, our interpretation of our conclusions for the computers at large would be that the time taken is only cubic in n . So, this is what our conclusions will be, for any computer or all computers. So, you see that we start off with the precise number over here. But, over here we are going down to a very rough statement.

And in some sense, we are saying in all this that this function, this expression which we are going to think of is a function in n . N is the problem size. So, this expression $10n^3 + 5n^2 + 7$. And this expression or this function, $2n^3 + 3n + 79$ are really in the same class. So, you want to define the notion of classes of functions.

(Refer Slide Time: 04:07)

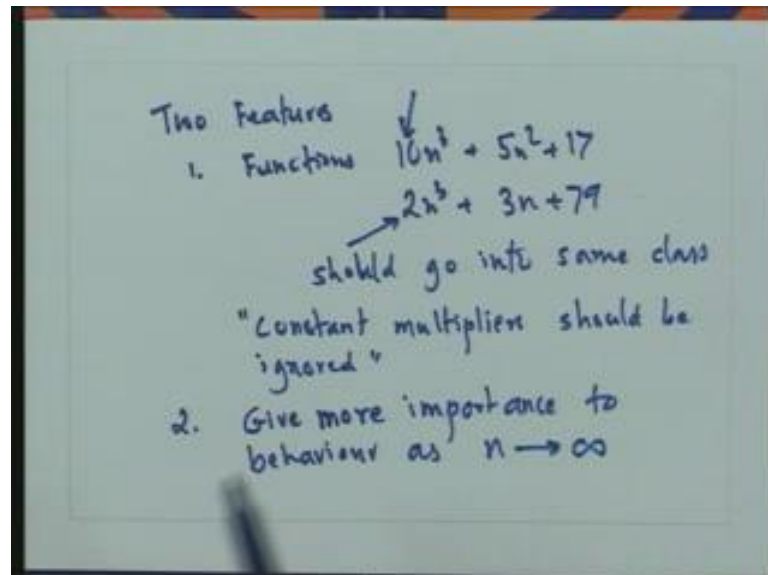


So, the idea is that we want to put functions in the same class and really think about the entire class. So, our conclusion will be that instead of saying that the time taken is $10n^3 + 5n^2 + 7$. We really want to say something like cubic, but we want to be a lot more systematic and formal about it. So, that is really the goal of today's lecture. So, we would like to develop the notation, which allows us to talk nicely about classes of functions.

So, asymptotic notation, so is a formal way or formal notation to speak about functions and classify them. Asymptotic analysis refers to the question of classifying functions or classifying the behavior of anything, but in this not too precisely, but by putting them

into classes. So, let me start out by writing down what do, we need from the classes, that we are going to define. So, we want really two kinds of features.

(Refer Slide Time: 06:05)



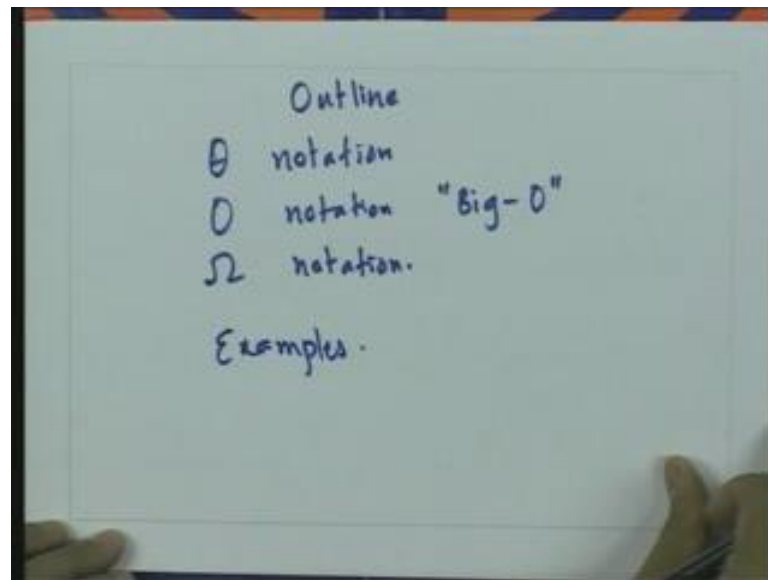
So one, we would like to put functions such as say $10n^3 + 5n^2 + 17$ and $2n^3 + 3n + 79$, should belong to the same class. Because, we said that in some sense we are going to be classifying these as cubic. And we want them to get together. Another way of saying this is that constant multipliers should be ignored. So, the constant multiplier over here is 10. The constant multiplier over here is 2.

So, we are going to ignore that. And that is our desire. Because, eventually what we can really say is that, the time taken on the RAM is this. And the time taken on any computer has to have the form, like something times n^3 . So, we want a class notation, which allows us to nicely ignore constant multipliers. Our class notation should also really worry about, what happens as n tends to infinity. So, we should give more importance to behavior as n tends to infinity.

So, it is also seen in this example itself as n tends to infinity. Really the $5n^2 + 17$ and this $3n + 79$, these two parts of these functions will go out. And therefore, we will really be worrying about $10n^3$ versus $2n^3$. And then, our first property or first feature which we said, we want in our class definition will take over.

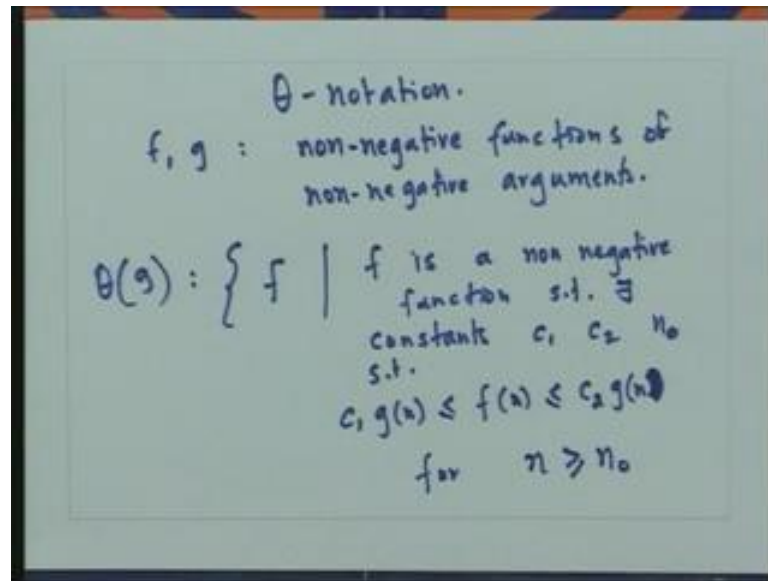
And it will say that, really $10n^3$ and $2n^3$ are really the same thing. So, that is the spirit. So, we want a notation a class notation, which will allow us to conclude that say functions of this kind are really similar or are in the same class. So, let me give an outline of today's lecture.

(Refer Slide Time: 08:43)



So, we are going to define three main kinds of notation today. So, one is the theta notation. One is the O notation. This is the capital or the big O . And then, there is the omega notation. And these will define function classes. And they will do exactly, what we said we want. And we will have lots of examples throughout. But, at the end we will also have a series of examples. So, let us go in order. And let me start off with the theta notation.

(Refer Slide Time: 09:38)



So, in what follows we are going to have functions, say something like f and g . And these functions are always going to be say non-negative functions, which will take non negative values functions of non negative arguments. This is natural in the sense that, we are going to be talking about time or may be sometimes the memory or any kind of resource. And these values will not really, there would not be any occasion when functions will need to take negative values.

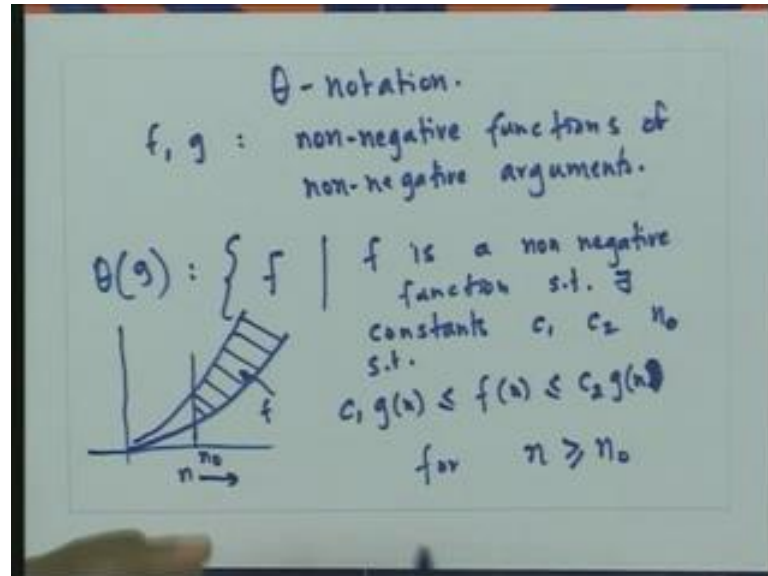
So, now theta of g where remember g is a function, is the following class. So, it is the class of all functions f where. So, let me write that down. So, f is a non negative function such that, there exists constant c_1, c_2 and n_0 . Such that $c_1 g(n)$ is less than or equal to $f(n)$ less than or equal to $c_2 g(n)$. And this is true, not necessarily for all values of n , but certainly for n greater than or equal to n_0 .

This is bit of a big definition. But, let me just indicate the spirit of it. So, let us go back to the properties that we wanted. So, we said that whenever whatever class structure we defined, should give importance to behavior as n tends to infinity. So, this is the part of the behavior. This is the part of that requirement. We are saying that for, only that we are only really bothered about, what happens as n is bigger than some n_0 .

So, we are not worried about smaller values of n . Then we said that we should really not be worrying so much about constant multipliers. So, this is also what is going on over

here. So, it says that we want f of n to be sandwiched between c_1 times g of n and c_2 times g of n . So, let me draw a picture here.

(Refer Slide Time: 12:35)

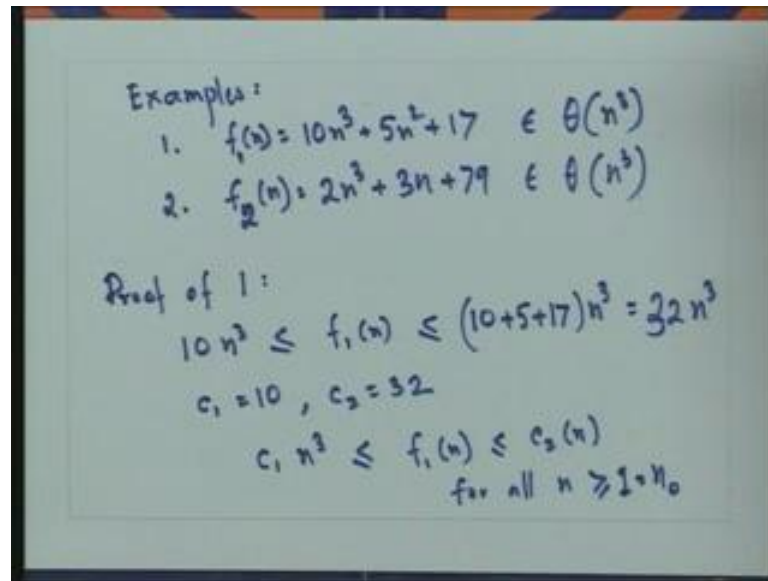


So, this is where I plot the function values and this is n . Then, c_1 times g of n will look something like this. Say in general, that is going to be something like. This is c_1 times g of n . This is going to be c_2 times g of n . And let us say this is n_0 . Then, our claim our requirement is that, if f occupies this region entirely and does not go beyond this region, go outside this region. This is the region for f , somewhere inside anywhere inside.

So, then it is sandwiched between c_1 times g of n and c_2 times g of n . Then we will put f in the class $\Theta(g)$. So, notice that we are not caring what happens, if for values of n below. Over here, f could go outside this. That is ok. We are not worried about that. But, beyond n_0 f must only lie between this sandwich region. So, essentially we are saying that, we do not worry about what that constant factor is.

So, it is bracketed below by some multiple of g . It is bracketed above by another multiple of g . So, essentially it behaves like g and that for large n . That is exactly, that is exactly in the consistent with the features that we wanted. So, let us take a few examples.

(Refer Slide Time: 14:12)



Examples:

1. $f_1(n) = 10n^3 + 5n^2 + 17 \in \theta(n^3)$
2. $f_2(n) = 2n^3 + 3n + 79 \in \theta(n^3)$

Proof of 1:

$$10n^3 \leq f_1(n) \leq (10+5+17)n^3 = 32n^3$$

$c_1 = 10, c_2 = 32$

$$c_1 n^3 \leq f_1(n) \leq c_2 n^3$$

for all $n \geq 1 = n_0$

Let us take our two functions, which we started. So, I will write down. Say f of n equals or let me call it f_1 of n equals, say $10n^3$ plus $5n^2$ plus 17 . Then, this function belongs to θ of n^3 . Let me write down. I will prove this, but let me write down the other claim as well. So, let me write f_2 of n another function, which is say $2n^3$ plus $3n$ plus 79 . And this also belongs to the same class n^3 .

So, I just want to reassure you. That the goals with which we started namely, developing a class notation which will enable us to put these two functions in the same class. Or those that goal is actually being met. So, let me now say in one what basis I am concluding something like this. So, let me go back to the old definition. So, in order to classify a function as being a member of this classed functions, this set of functions.

All I need to do is, to find suitable constants c_1 c_2 and n_0 . So, if I find these suitable a constant c_1 c_2 n_0 such that, these properties are met. Then, I am really done. So, let us take one. So, let me write this down as proof of one. So, clearly $10n^3$ is less than or equal to f_1 of n . Because, there is a that additional $5n^2$ plus 17 term. I can write down f_1 of n is certainly less than. I will just raise all these to n^3 , instead of keeping them n to the zero n^2 over here.

So, this is certainly less than 10 plus 5 plus 17 times n^3 . Or this is equal to $32n^3$ or $32n^3$ really. And this is true for all n . So, I have established that if I take c_1 equals 10 and c_2 equals 32 . Then, c_1 times n^3 is less than or equal to f_1 of n less

than or equal to c_2 of n for all n greater than or equal to say even 1. That does not really matter. So, these constants c_1 , c_2 and n_0 , and 1 is equal to n naught having found. And we know that the functions. And these constants satisfy the properties that, we wanted.

And therefore, we can legitimately claim, that f_1 belongs to this class. So, the conclusion from this is that f_1 belongs to the class n^3 . And in fact, the same kind of analysis where example is to prove, that f_2 also belongs to the same class. Let me take one more example, which actually illustrates that our notation is a bit more is going beyond what we really started off.

(Refer Slide Time: 17:54)

$$3. f_3(n) = 10n^3 + n \log n \in \Theta(n^3) \neq \text{"cubic"}$$

$$10n^3 \leq f_3(n) \leq 11n^3$$

$$c_1 = 10 \quad c_2 = 11 \quad n_0 = 1$$

So, I am going to argue now. So, that suppose I take f_3 of n is equal to n^3 plus, say $n \log n$. Then, even this is in the class $\Theta(n^3)$. This is not something that you would classify, as being cubic, because cubic has the connotation of cubic polynomial. So, it has to have the form something times n^3 plus something times n^2 plus something times n plus a constant.

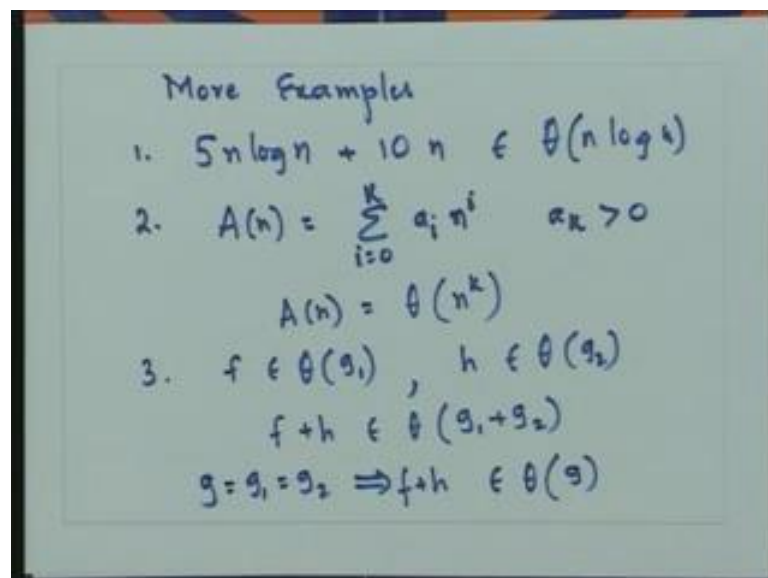
Whereas here there is something funny that is there is a term, which is not a polynomial term. However, note that it still is true that $10n^3$ is less than equal to f_3 of n , because $n \log n$ is certainly always at least zero or certainly greater than zero in fact. And in fact, I can since I know that, $n \log n$ less than n^3 . I can also write this as less than or equal to $11n^3$. So, I have found c_1 equals 10 c_2 equals 11.

And say n naught equals 1 for which, my whole definition holds. And therefore, I can write fully claim that this function f also belongs to the class n cube. So, this is a good idea. It is good that in fact, this function belongs to this class n cube. Because, as n increases as n tends to infinity then, this function really is the same as $10 n$ cube essentially, because this term is going to be negligible as for large n as compared to this term.

And so therefore, since it is essentially the same as n cube. It is a good thing that our classification system, is putting it in the same class. Let me write down a few more examples. Actually before that, let us come back to this definition itself. So, θ of g is a set of functions or class of functions. And we are going to think of g as being sort of a representative or $v g$ s being a sort of a prototypical function.

So, instead of talking about a very detailed function like, say $10 n$ cube plus $n \log n$ or $10 n$ cube plus $5 n$ square plus 17 . We will say, we are roughly going to say that it is n cube. If we are ignoring constant factors and as n goes to infinity. And we will instead be saying that, it is that the class θ of n cube. So, think of g also as a representative of all this, all these functions. So, let us take a few more examples.

(Refer Slide Time: 20:53)



More Examples

1. $5n \log n + 10n \in \theta(n \log n)$
2. $A(n) = \sum_{i=0}^k a_i n^i \quad a_k > 0$
 $A(n) = \theta(n^k)$
3. $f \in \theta(g_1), h \in \theta(g_2)$
 $f + h \in \theta(g_1 + g_2)$
 $g = g_1 = g_2 \Rightarrow f + h \in \theta(g)$

Some, I will write down may be that $5 n \log n$ plus $10 n$, belongs to the class θ of $n \log n$. So, the important point over here is that, this $10 n$ is grows slower. So, as n becomes large this term is going to dominate this term. And therefore, its behavior

should be essentially the same as that of $n \log n$. And therefore, it is in the same class. We should really prove this. And I will leave that as an exercise.

We should really prove that, a function of n which is $5n \log n$ plus $10n$ belongs to this class $\theta(n \log n)$. And by that, I mean you should exhibit constant $c_1 < c_2$ and n_0 such that c_1 times $n \log n$ is less than or equal to this, which in turn is less than or equal to c_2 times $n \log n$, for all n greater than the n_0 that we defined. That is a fairly easy task, but if you certainly do it. So, as to make sure that you are fully conversant with this definition.

Let us take a slightly more complicated, but complicated looking example. So, let us say we have a polynomial a of n which goes something like. Say summation i going from zero to k of $a_i n^i$. And let us assume that, a_k is greater than 0. Other terms could be smaller, but a_k is bigger than 0. Then, I will claim that this function a of n in general, any polynomial of k th degree is in the class $\theta(n^k)$.

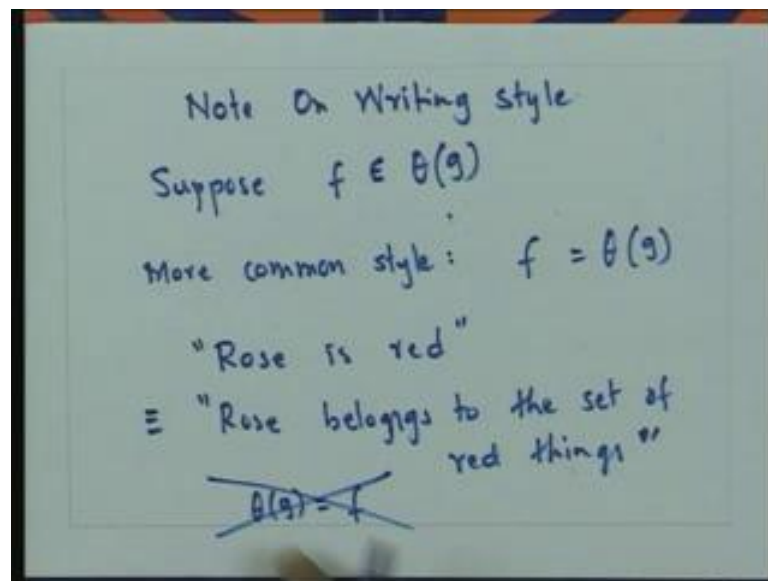
Again you should prove this. But, this proof is really similar to what we have done earlier. And there should not any difficulty what so ever. Again the message is the same that, if you have a function. You look at the most, the largest determinate. And that is really is, it is class. That really is its asymptotic complexity class. Let me now write down, some properties of this definition. So, suppose we have f belonging to $\theta(g_1)$. And say h belonging to $\theta(g_2)$.

Then, I claim that f plus h belongs to $\theta(g_1 + g_2)$. This also you should be able to verify, fairly straight forward. And furthermore, if say g_1 this is the same as g_2 . Then f plus h when I write f plus h , I really mean f of n plus h of n , the function which returns for every n , f of n plus h of n . And this function belongs to $\theta(g)$. This is also understandable, because if you believe the previous result. So, $g_1 + g_1$ equal to g_2 equal to g .

So, $g_1 + g_2$ will be equal to twice g . And the class $2n^3$ is $\theta(2n^3)$ is really, the same as the class $\theta(n^3)$. Again you should be able to verify this. You should verify this. And it is really not surprising, because we started off by saying that we really do not want to worry about constant factors. And therefore, it does make sense or it should make sense, to have $\theta(n^3)$ with the same as $\theta(2n^3)$.

We however, never ever write theta of $2n$ cube. And that is, because it is much simpler and much nicer to say, theta of n cube. So, when we write theta of g we do not, we drop off the constant multipliers, if any that might be present in g . So, we have defined the first class of functions that we wanted. And it does indeed, it is indeed consistent with our intuition. And the goals we set ourselves. However, although this is a class of functions, in computer science and mathematics there is a funny style evolved, as far as writing down these classes is concerned.

(Refer Slide Time: 25:38)



So, let me write that. Let me write that down. So, I am going to write down a note, on writing style. So, suppose f belongs to the class theta of g . Now, very often or most commonly this is not written in this manner. But, the more common writing style is, to write f is equal to theta of g . Unfortunately, the assignment operator is again being badly abused over here. This seems to be a tradition in computer science.

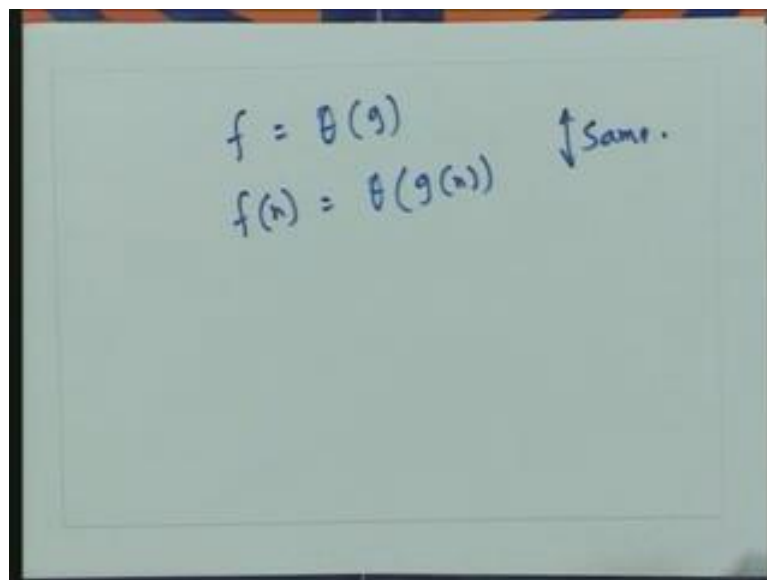
We use assignment to mean, we use the equal to operator to mean assignment. We use it to mean. Well, first of all it has the value equality. Then, it has this value. It has, it is used to indicate assignment. And here, we are actually using it to denote inclusion. However, you will see that you will not really be bothered by this. It will become very clear by the context that is, what we mean.

Actually the situation is really similar to our use of English language words. So, for example, we might write Rose is Red. When we write this, we really mean that Rose

belongs to the class of red things or the set of red things. So, as you can see even in the English language, the verb is used to indicate equality. That is perhaps the more common use. But, it is also used to indicate some kind of say conclusion.

So, anyway instead of saying f belongs to theta of g , it is very common to say f equals to theta of g . We never however, write theta of g equal to f . This is never written. Just as, it does not make sense to say red, well I guess in poetic English. It does make sense to say red is Rose, but we never write this in computer science. I will add one more note on the writing style. So, I have been writing functions as functions by their names directly. So, I might write something like f equals theta of g . But, from time to time I might also write f of n is equal to theta g of n .

(Refer Slide Time: 28:34)



A photograph of a chalkboard with two lines of handwritten text in blue ink. The first line is $f = \theta(g)$. The second line is $f(n) = \theta(g(n))$. To the right of these two lines, there is a vertical double-headed arrow pointing up and down, with the word "Same." written next to it.

If n is it is sort of understood, n is clearly understood as an unbound variable. The argument, the possible argument that f can take. So, these two really will be think of these as being the same. This I might write it in this manner. Just to emphasize the fact that, f is a function. But, if it is clear that, f is a function then I am it might be god to write it in this manner. Let me take one more example.

(Refer Slide Time: 29:07)

Example.

$$f_5(n) = 2 + \frac{1}{n} \in \theta(1)$$
$$g(n) = 1$$
$$2 \leq 2 + \frac{1}{n} = f_5(n) \leq 3$$

$c_1 = 2$ $c_2 = 3$ $n_0 = 1$ $\theta(1):$

Let me define f_5 of n as 2 plus 1 over n . So, what can we what class can we put this in. So, it turns out that there is actually a nice class into, which we can put this. And this class is simply the theta of 1 class. So, here let us say g of n is always equal to 1. And then, we have to argue that. In fact, f_5 belongs to theta of g , which is what I have written over here. Let us just do this. Just to make sure, we understand this.

So, clearly 2 is less than or equal to 2 plus 1 over n , which is equal to f_5 of n . And in fact, this is less than or equal to say 3. And therefore, we have c_1 equals 2 c_2 equals 3 and this is true for all n . So, we can have n_0 equals 1. And we have these three constants, satisfying our basic definition of theta. And therefore, we can write f_5 as belonging to theta of 1. So, theta of 1 is the class of all functions, which are essentially constant.

They may have some minor perturbations, but they really are like constants. So, now we will come back. We will come to our other two definitions. Our other two classes and we will define those. So, the first class is O of g .

(Refer Slide Time: 30:53)

$$\begin{aligned}
 O(g) &= \left\{ f \mid \begin{array}{l} f \text{ is non negative function} \\ \exists c_2, n_0 \\ f(n) \leq c_2 g(n) \text{ for } n \geq n_0 \end{array} \right\} \\
 \Omega(g) &= \left\{ f \mid \begin{array}{l} f \text{ is non neg. fn.} \\ \exists c_1, n_0 \\ c_1 g(n) \leq f(n) \text{ for } n \geq n_0 \end{array} \right\} \\
 \Theta(g) &: \quad \exists c_1, c_2, n_0 \\
 &\quad c_1 g(n) \leq f(n) \leq c_2 g(n)
 \end{aligned}$$

So, this is a class of functions f , where such that where f is a non negative function. And there exist c_2 and n_0 , such that f of n is less than or equal to c_2 times g of n , for all n greater than n_0 . Omega of g is the same thing as above. So, it is f but. So, f is non negative. But, now we are worried about c_1 and n_0 . Such that f of n is less than c_1 times g of n is less than or equal to f of n . And we do not have anything on the upper for all n greater than n_0 .

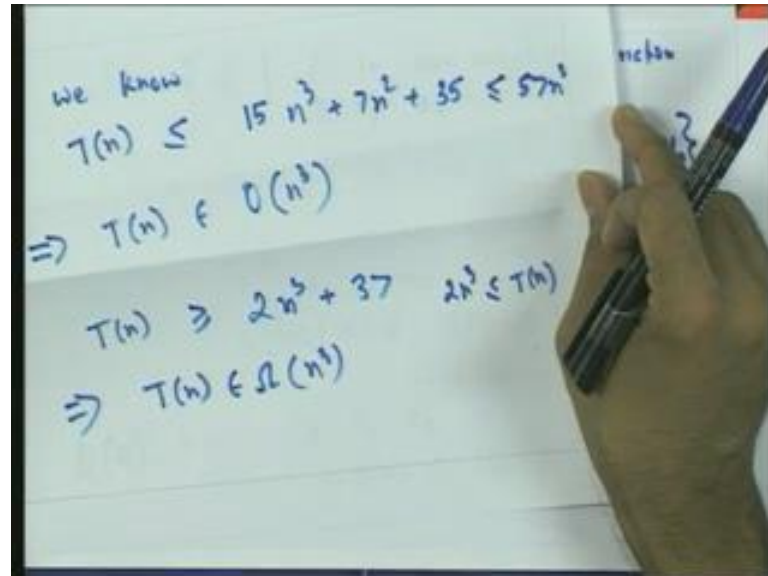
And let me just refresh you, what the theta of g definition was. The only difference was that, we wanted there to exist c_1, c_2 and n_0 . Such that, c_1 times g of n is less than f of n and is less than c_2 times g of n . So, you can see that the class omega relaxes one of the conditions, which was present in the class theta and so it does O . O relaxes the lower bound condition. And omega relaxes the upper bound condition.

So, in omega the lower bound condition is present. But, we are not saying anything about, whether f of n is bounded above by some g . Most of an algorithm analysis, we find it is easy it is reasonably easy to say that the time taken is at most something like this, at most this function. So, for example in last lecture. In the last lecture, we said something like if we look at this program. And we count the number of iterations.

Then, we can certainly argue that there are at most n^3 iterations or something like that. And therefore, the time taken has to be at most cubic in n . There we later on did argue that, the time has to be at least cubic. But, suppose we just had argued that it was

utmost cubic. Then, we would have used this O notation. So, we would have said that the time taken belongs to O of n cube. So, let me write that down.

(Refer Slide Time: 34:02)



The image shows a hand writing on a whiteboard. The text is as follows:

we know
 $T(n) \leq 15n^3 + 7n^2 + 35 \leq 57n^3$
 $\Rightarrow T(n) \in O(n^3)$

$T(n) \geq 2n^3 + 37$ $2n^3 \leq T(n)$
 $\Rightarrow T(n) \in \Omega(n^3)$

So, if we know say the time taken as a function of n is less than or equal to say 15 n cube plus 17 plus 7 n square plus 35 or something like that. Then, we can conclude that t of n belongs to O of n cube. If in addition, we prove that t of n is also greater than say 2 n cube plus 37. Then, this would imply let us go back to our definition. So, let me put it on top. So, let us see. So, here we are establishing that t of n is bigger than 2 n cube plus 37.

Or I can write this as, say 2 n cube is less than or equal to t of n. So, which is exactly the condition that, we wanted over here. And therefore, we could argue that t of n belongs to the class omega of n cube. Over here in the first case, in the first case we said that t of n we know that of, n is less than 15 n cube plus 7 n square plus 35 and which I can write down as in fact, less than 35 plus 7 plus 15, so 57 n cube.

And that is really satisfying this condition. And therefore, I can conclude that t of n is belonging to this class. But, what happens as a result of both of these. So, I have really established that, this t of n is bounded below by c 1 times g of n. Simply, this 2 n cube. And it is bounded above by c 2 times g of n, which is simply this 57 n cube.

(Refer Slide Time: 36:29)

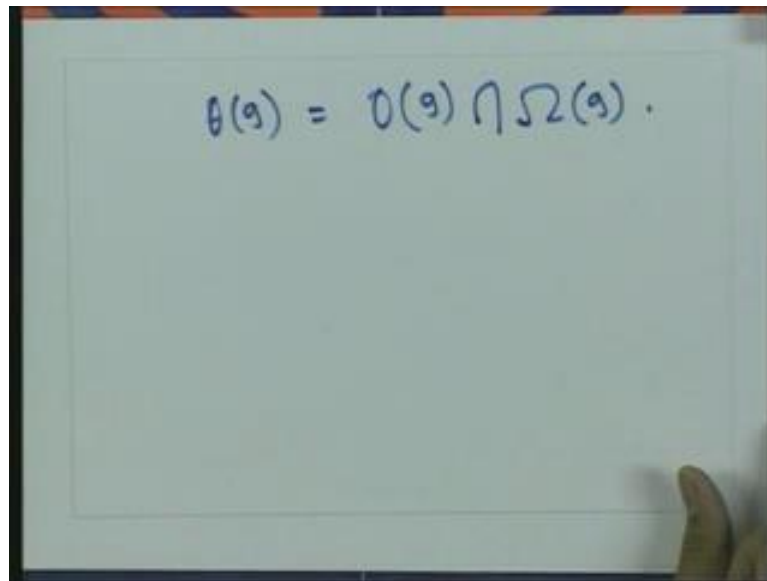
The image shows a whiteboard with handwritten mathematical derivations. At the top, it says 'If we know' followed by the inequality $T(n) \leq 15n^3 + 7n^2 + 35 \leq 57n^3$. Below this, it concludes $\Rightarrow T(n) \in O(n^3)$. A large curly brace on the right side of the board groups the upper and lower bound derivations. The lower bound derivation starts with $T(n) \geq 2n^3 + 37$ and $2n^3 \leq T(n)$, leading to $\Rightarrow T(n) \in \Omega(n^3)$. A horizontal line is drawn below this, and the final conclusion $T(n) \in \Theta(n^3)$ is written underneath.

$$\begin{aligned} \text{If we know } T(n) &\leq 15n^3 + 7n^2 + 35 \leq 57n^3 \\ \Rightarrow T(n) &\in O(n^3) \\ T(n) &\geq 2n^3 + 37 \quad 2n^3 \leq T(n) \\ \Rightarrow T(n) &\in \Omega(n^3) \\ \hline T(n) &\in \Theta(n^3) \end{aligned}$$

So, as a result what has happened is that, I can conclude from both of these things that T of n is belonging to Θ of n belongs to Θ of n cube as well. So, let me make this point again, because it is an important point. The class O of g is the class of functions, which are bounded above by g . So, if we know something about what is bigger than these, than the function that we are considering.

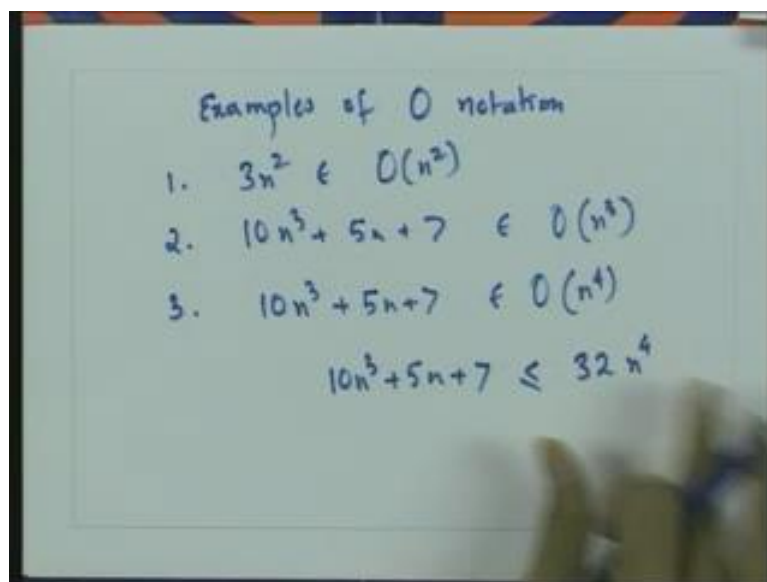
If we know a function, which is bigger then, we can say we can put it. Put the unknown function in this class, in this O of g class. So, if you know an upper bound on a function then, we should be looking at expressing that upper bound as O of g . If we know a lower bound on that function, we should be looking at expressing this knowledge as this f belongs to Ω of g . And if we know both upper bound and lower bounds in terms of the same function g then, we should write that this function f belongs to Θ of g .

(Refer Slide Time: 37:48)


$$\theta(g) = O(g) \cap \Omega(g).$$

So, as you must have already guessed the class theta of g is simply the union of the classes O of g, the intersection of the classes O of g and omega of g. So, let us take some examples of O.

(Refer Slide Time: 38:13)



Examples of O notation

1. $3n^2 \in O(n^2)$
2. $10n^3 + 5n + 7 \in O(n^3)$
3. $10n^3 + 5n + 7 \in O(n^4)$

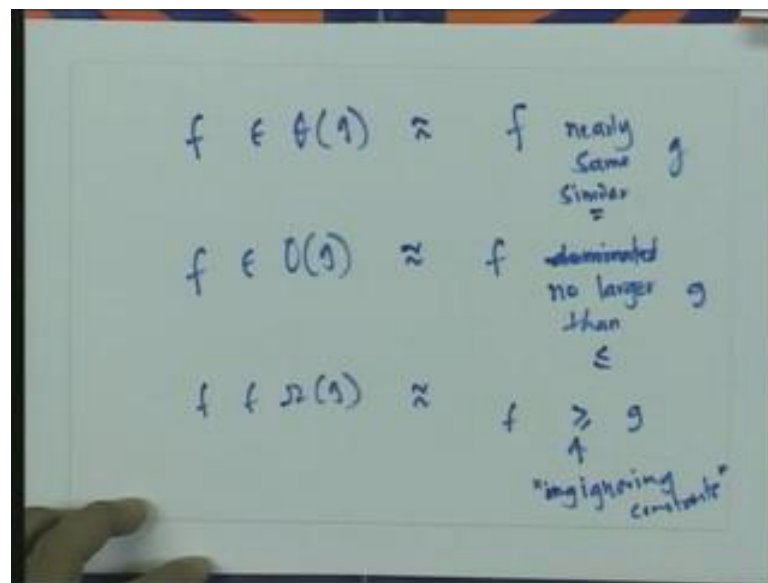
$$10n^3 + 5n + 7 \leq 32n^4$$

This is something like 3 n square belongs to O n square. 3 n square in fact, belongs to theta of n square. And therefore, it certainly belongs to O of n square, because O of n square in fact, is bigger than theta of n square. Here is another example. So, say 10 n cube plus 5 n plus 7 belongs to O of n cube. Similar logic, but something like 10 n cube

plus $5n$ plus 7 also belongs to O of n to the fourth. Why is this? Because, $10n^3$ plus $5n$ plus 7 is less than or equal to I can certainly write this as being less than or equal to say $32n^4$.

And that is all my definition of O really cares about. So, this also belongs to this. This function belongs to this class. No surprise, you should be surprised by this. Because, we are really saying that g serves like an upper bound on this function. And if I can, if n^3 is an upper bound then certainly n^4 is an upper bound as well.

(Refer Slide Time: 39:45)



Let me summarize that f belongs to theta of g , should be read as f is nearly the same. Or let me write it as similar to g . f belongs to O of g , should be read as f dominated by g . Actually, not dominated. f is no larger than g . Sort of like less than or equal to, but it is not exactly less than or equal to. Because, we are ignoring constants just as this is sort of like equal to. f belongs to omega of g , like y should be thought of as f greater than or equal to g . But, again we are ignoring constants. And also lower order terms.

So, we now have defined our three main functions classes, theta of g , O of g are also called big O of g and omega of g . I have defined these classes in the context of the times taken by algorithms. But of course, these are just plain old function classes. And the functions could denote, not necessarily the time taken, but any old thing. So, for example let me define a general function. Which just, it is the sum of n numbers.

(Refer Slide Time: 41:33)

The image shows a whiteboard with handwritten mathematical derivations. At the top, the sum $S(n) = \sum_{i=1}^n i$ is written, followed by the exact formula $\frac{n(n+1)}{2}$. Below this, an upper bound is derived: $S(n) = \sum_{i=1}^n i \leq \sum_{i=1}^n n = n^2$. This leads to the conclusion $\Rightarrow S(n) \in O(n^2)$. Then, a lower bound is shown: $S(n) = \sum_{i=1}^n i \geq \sum_{i=\frac{n}{2}+1}^n i \geq \sum_{i=\frac{n}{2}+1}^n \frac{n}{2}$. The final calculation shows $= \frac{n}{2} \cdot \frac{n}{2} = \frac{n^2}{4}$, which leads to the conclusion $\Rightarrow S(n) \in \Omega(n^2)$.

So, let me say S of n is equal to summation i going from 1 to n of i itself. Well you do know, from say some of the mathematics courses that you have done. That S of n is nothing, but n into n plus 1 upon 2. However, getting into a result like this, requires some amount of cleverness, this result is a very precise result. If you prove that S of n is exactly this, it is a very precise result.

But, sometimes you might say, you might not have enough time or you might not have enough cleverness to get on exact result like this, which is to say that S of n is exactly n into n plus 1 upon 2. But, suppose you might be happy with a weaker results. So, you might want to know well, does S of n grow or does S of n belong to n square into the class θ of n square or does it belong to the class θ of n .

At first glance just by looking at this, it should it is not clear at all whether S of n belongs to the class θ of n square. Or whether it belongs to class θ of n cube or anything like that. So, what I want to stress or what I want to give an example of right now is that, even without getting to this exact expression, you might be able to determine the class to which a function belongs. And in fact, we are going to prove that S of n belongs to the class θ of n square, without actually calculating S of n precisely.

And this is an instructive example, because something like this will happen when we analyze algorithms. So, S of n is equal to summation i going from 1 to n of i . But, note that, this i is always going to be at most n . And therefore, I can write this as summation i

going from 1 to n of n itself. But, what is this? This is just n plus n plus n n times, because every term in this sum is n. And therefore, this is nothing, but n square.

So, what have you established? We have established that S of n is less than or equal to n square. But, that right away puts S in the class O of n square. So, this implies that S of n belongs to the class O of n square. Can we argue that S of n belongs to the class omega of n square? That is, what we are going to do next. So, again we observe something very simple. So, S of n is summation i going from 1 to n of i.

Now, I am going to ignore the first in our two terms. So, this certainly is greater than or equal to. If I ignore the first in our two terms, this is going to be i going from say n over 2 plus 1 to n of i itself. But, now note that this i is always going to be at least n over 2, because it starts at n over 2 and goes all the way till n. So, therefore, I can write this as summation i going from n over 2 plus 1 to n of n over 2 itself.

So, term by term this series this sum is bigger than the corresponding term over here. But, what is this? This is simply n by 2 added to itself n by 2 times. So, therefore, I will write this as equal to n by 2 times n by 2, which is n square by 4. So, I have argued that S of n is bigger than or equal to n square by 4 and at most n square. So, we have bracketed S of n. Well before that, just once we argue that S of n is bigger than n square by 4. We can conclude that, S of n belongs to omega of n square. Just going back to our definition, we have proved that. Let us just do this once.

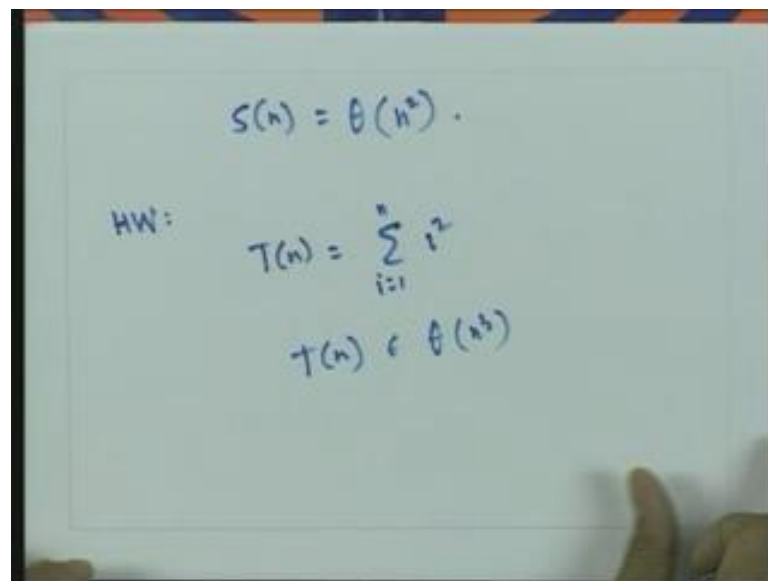
(Refer Slide Time: 45:59)

Handwritten definitions for asymptotic notations:

- $\Rightarrow S(n) \in \Omega(n^2)$
- $\exists c_2, n_0$
 $f(n) \leq c_2 g(n) \text{ for } n \geq n_0$
- $\Omega(g) = \{ f \mid \begin{array}{l} f \text{ is non neg. fn.} \\ \exists c_1, n_0 \text{ positive} \\ c_1 g(n) \leq f(n) \text{ for } n \geq n_0 \end{array} \}$
- $\Theta(g) : \begin{array}{l} \exists c_1, c_2, n_0 \text{ positive} \\ c_1 g(n) \leq f(n) \leq c_2 g(n) \end{array}$

So, we need to argue that f of n which is S of n now, is greater than or equal to n square by 4. And the c 1 is now 1 by 4, but that is. We do not, we did not necessarily say we did not say over here that, c 1 has to be greater than 1 or anything like that. Any real number, any positive real number is fine. There exist, I should write this as there exists c 1 on n naught positive all through. All the c 1 naught that, I have written should be positive. That is what, that is exactly what we have proved over here S of n . So, we have argued that S of n belongs to n square as well and from this and this, what can I conclude?

(Refer Slide Time: 46:48)



$$S(n) = \theta(n^2).$$

HW:

$$T(n) = \sum_{i=1}^n i^2$$

$$T(n) \in \theta(n^3)$$

Well from those two things, I can conclude that S of n must be θ of n square as well. So, notice that in doing any of this I did not actually exactly evaluate S of n . In this case evaluating S of n exactly is possible, but very often it is not. As I said, it may require exceptional cleverness once in a while to exactly evaluate a function. But, giving bounds on it is easy. And the bounds on it, can be nicely stated in terms of the class notation that we have right now.

So, if you know an upper bound we can state it as S of n belongs to O of something. If you know a lower bound, we can state it as S of n belongs to ω of something. And if we know both, we can state S of n as θ of something. In this manner, I would like you to prove. So, let me write this down as home work. Prove that, say t of n if t of n is equal to summation i going from 1 to n of say i square. Then, prove that t of n belongs to

theta of n cube. The proof is really more or less identical, but you should. But, again you should persuade yourself of this. Let me take one more example.

(Refer Slide Time: 48:34)

$$F(n) = F(n-1) + F(n-2) \quad F(1) = F(0) = 1$$

Claim: $F(n) \geq 2^{n/2} \dots$ Homework.

$$F(n) = \sum (2^{n/2}) = \sum (\sqrt{2}^n)$$

Actual Result: $F(n) = \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$ ↑
exponential growth

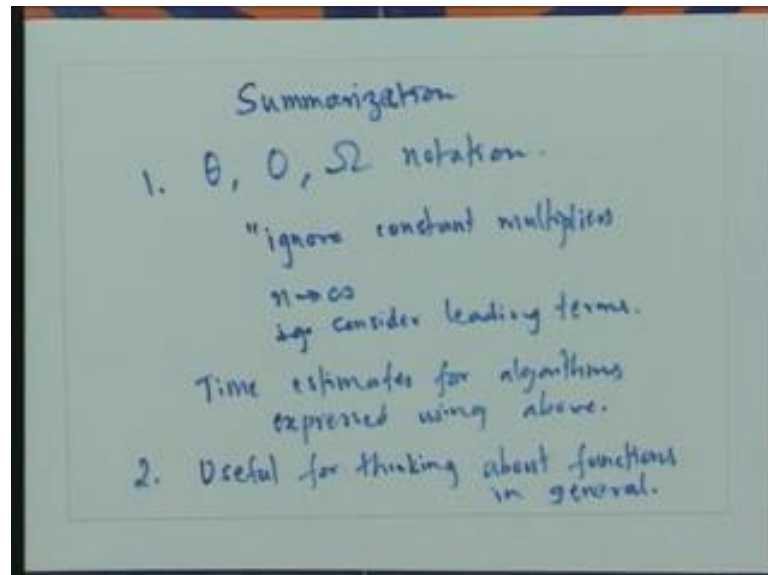
So, let us look at the Fibonacci series which is defined by f of n equals f of n minus 1 plus f of n minus 2. And f of 1 equals f of 0 equals 1. Let me claim that, f of n is always greater than or equal to 2 to the power n by 2. This is also home work. What can you conclude from this? You can conclude from this that, f of n is an omega of 2 to the n by 2. Let me make, one more claim. Well, let me not make that claim.

Let me instead tell you the real result, actual result. It is possible to show that f of n is theta of 1 plus root 5 upon 2 whole to the power n . So, it is not exactly this number, but it is within a constant multiplier of this. The n th Fibonacci number is within a constant multiplier of this. Proving this exact bound, takes a lot more work. But, by something really simple we have at least argued that f of n is actually going to grow at least as 2 to the n by 2. Or in fact, I can write this as omega of root 2 to the n .

So, something like this is commonly called exponential growth. Some by a very easy logic, by very easy reasoning we can argue that the Fibonacci number grow exponentially. By more complicated reasoning and by reasoning, which involves essentially involves finding out a precise formula for the n th term. We can get a much tighter results. But, the importance right now is that our theta notation and our omega

notation allow us to express our knowledge or lack of it, in a very compact manner. So, let me summarize now.

(Refer Slide Time: 50:59)



So, one we have defined theta, O and omega notation. These capture the idea that, ignore constant multipliers. Consider n goes to infinity, which is equivalent to saying that ignore consider leading terms. So, our time estimates of algorithms will be expressed using these notations. The second thing that I want to mention is that, in fact this notation is just a general notation on functions. And it can be used for other things as well and sometimes.

And it really allows us to express, what we know and what we do not know in a very compact manner. So, even if we do not know even if we know a little bit about it then, we can put it in a certain class and that, this partial information that we have can be nicely expressed. So, this is useful for thinking about functions in general.

Thank you.