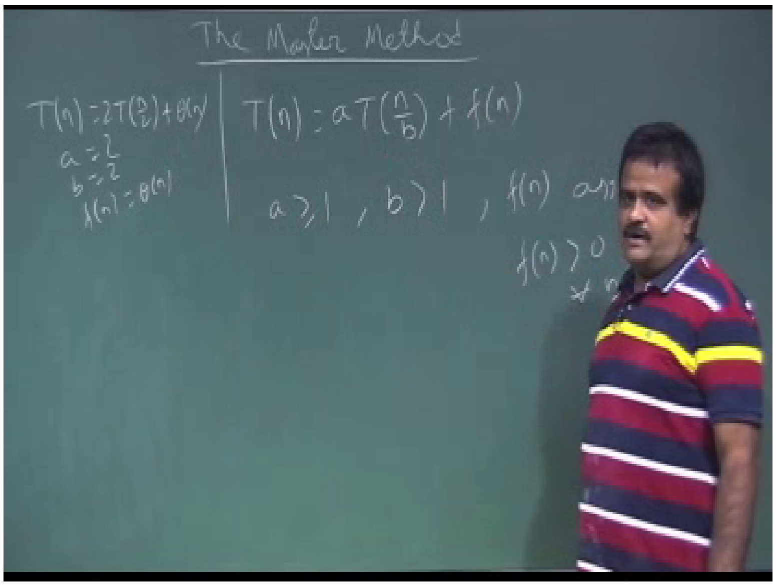
An Introduction to Algorithms

Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 06
The Master Method

So we talk about another method powerful method to solve the recurrence, which is called master method it has recurrence.

(Refer Slide Time: 00:36)



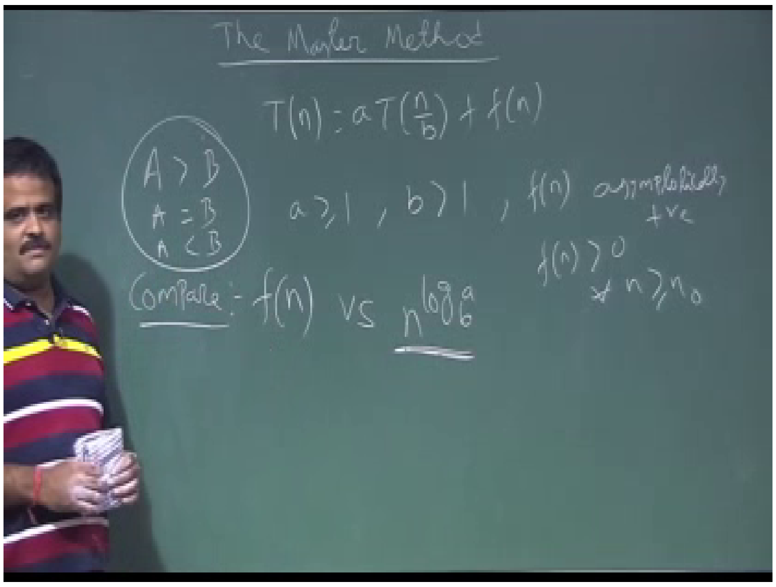
In this method the recurrence is a particular form line T n is equal to a T n by b plus f of n where a is greater than equal to 1 and b is greater than 1 and f n is f n is positive function f n is basically called asymptotically positive asymptotically positive so; that means, f n is greater than 0 for all a after starting value of n 0.

So, if it is negative then for the initial few values are it is negative otherwise it is positive in the most of the example we will see a finished throughout positive. So, if our recurrence is up this form then only we can solve it, we can we could it could be solved by master method for what it is there are three case of the master method we come to that what are the cases. So, now, the question is how come we can get this type of recurrence which algorithm can give us this type of recurrence. So, again we can think for a divide and conquer technique.

So, we have a problem of size n, we divide the problem into sub problems a means sub problems. So, at least we should have one sub problems and the size of the sub problems is n by b. So, that is it should able to reduce the size. So, that is why b is greater than one and this is the cost for combining the solution of the sub problems. So, this type of recurrence we can easily see like merge sort merge sort of this this type of recurrence. So, merge sort is T n is equal to what is the recurrence 2 T n by b plus.

So, merge sort a is equal to 2, b is equal to 2, and f n is equal to theta of n or c of n. So, this is merge sort this comes under this category.

(Refer Slide Time: 02:55)


The Master Method
$T(n) = aT(\frac{n}{b}) + f(n)$
$A > B$
$A = B$
$A < B$
$a \geq 1$, $b > 1$, $f(n)$ asymptotically +ve
$f(n) > 0$ $\forall n \geq n_0$
Compare:- $f(n)$ vs $n^{\log_b a}$

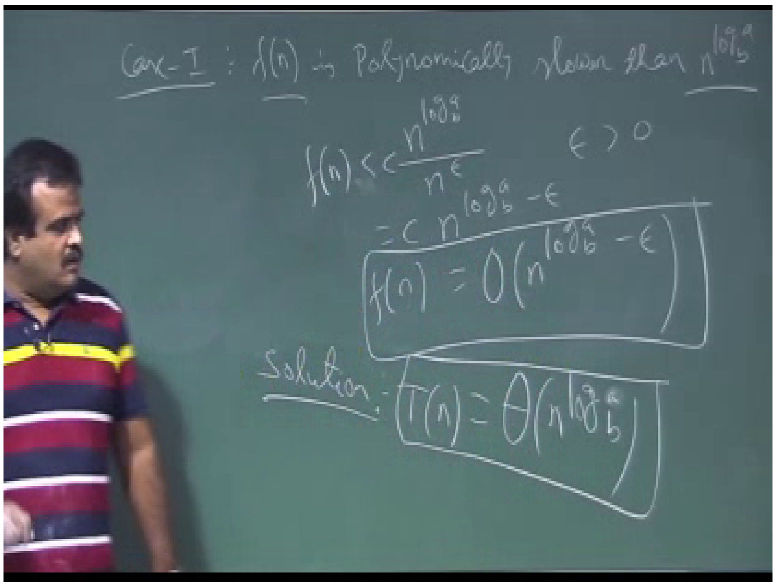
Now in order to get the solution of this master method we need to compare we compare basically 2 terms one is f n which is the merge cost. So, basically we compare f n verses n to the power log a base b. So, f n is the merge cost and as if this is the cost for recursively how much time you have spending here. So, we are solving this problem recursively we have aiming sub problems.

So, this is the time how much time we have spending here to solve this problem recursively, and this is the time for merging the solution. Now the run time will be the time which is dominating because run time is the how much time we have recursively spending here and how much is the merge time the total time, now if this time if as if for the time is let us take this is the time for how much time we have spending here solving this problem recursively, now if the end time is more if say that is a say for a example the 1000 seconds say for a example and say this is a 5 seconds.

So, what is the solution? Solution is thousand seconds so; that means, run time will be the time which is we compare this 2 now which is more which is dominating in a asymptotic sense; obviously, because we are talking about run time in the asymptotic with the help of asymptotic notation. So, the time run time will be the time which is the more among this 2, because this is the time how much we are spending here and this is the time for combining or merging now the run time will be the time which is dominating. So, if we compare 2 term there are three possibilities.

If we compare A B say 2 two terms A B. So, that (Refer Time: 04:51) A is greater than B A is equal to B, A is less than B. So, there are three possibilities. So, that is why we have three cases of the master method based on this three possibilities. So, first case is this is bigger, second case is they are similar and third case is this is bigger. So, now, in asymptotic sense how we justify, how we mention that how we represent that one is bigger than another. So, that is also a important issue. So, let me just. So, there are three cases based on this comparison.

(Refer Slide Time: 05:37)


Case-I : f(n) is Polynomially slower than $n^{\log_b^a}$
$f(n) << \frac{n^{\log_b^a}}{n^\epsilon}$ $\epsilon > 0$
$= c\, n^{\log_b^a - \epsilon}$
$f(n) = O(n^{\log_b^a - \epsilon})$
Solution: $T(n) = \Theta(n^{\log_b^a})$

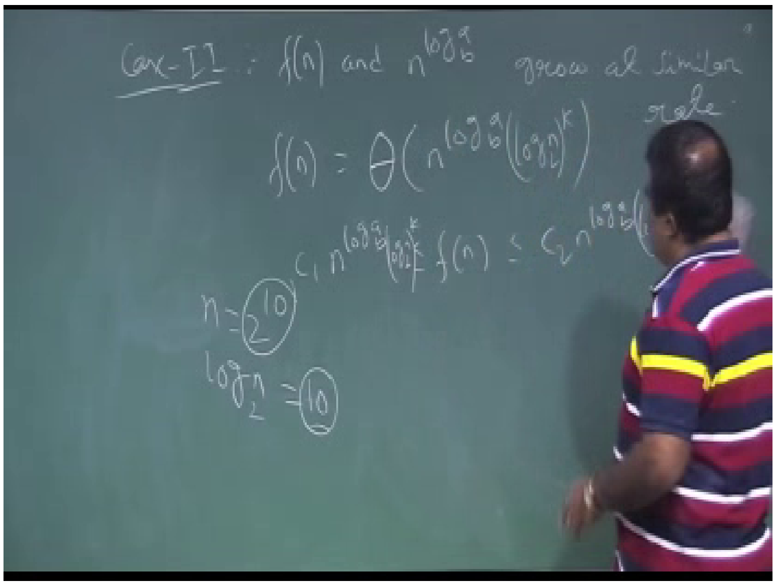
So, first is say case one; case one means we want this to be bigger this to be slower. So, how we can write this let us write this f n is polynomically a slower, f n is polynomically slower than this term n to the power log a base b. So, f n is polynomically slower than this term polynomically slower means what. So, that means, this is bigger so; that means, n to the power log a base b if we divide by a polynomial term, then also a f n slower or epsilon is greater than 0 if epsilon is this one this is n, epsilon is 2 n square. So, this is a polynomial time.

So, if we divide these with the polynomial term then also a f n is less than the that. So, that is the says that f n is polynomically slower than this term. So, this is basically what. So, this is basically in a (Refer Time: 07:00). So, this is basically c n log of minus epsilon. So, that is basically f n is big o of n to the power log a base b minus epsilon. So, this is the way we say one term is faster than the other terms. So, this is the meaning of f n is polynomically slower than n to the power log a base b.

So that means, even if we divide a polynomial factor with this, this is the n to the power epsilon, epsilon could be greater than greater than zero; that means, it could be one 2 like this if it is one then this is n n is a polynomial if it is 2 n square. So, these are polynomial term so that means, if you divide a polynomial term with this then also f n is slower so; that means, this term is bigger this is the dominating term then what is the solution? Then the solution will be this is called case one of the master method then the solution is the term which is dominating big theta of n to the power log a base b. So, this is called case one of the master method.

So, this is we have to remember. So, the n to the power log a base b is the dominating term. So, this is the sorry n to the power log a base b. So, so this is case one; now case 2 case 2 is they are similar their growth is similar.

(Refer Slide Time: 08:54)

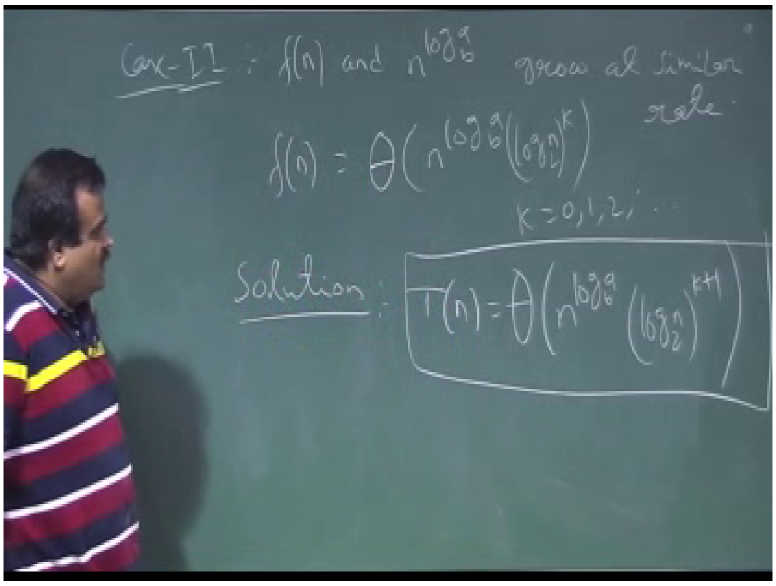

Case-II : f(n) and $n^{\log_b^a}$ grow at similar rate.
$f(n) = \Theta(n^{\log_b^a}(\log_2^n)^k)$
$c_1 n^{\log_b^a}(\log_2^n)^k \le f(n) \le c_2 n^{\log_b^a}($
$n = 2^{10}$
$\log_2^n = 10$

So, case 2 is basically they are similar. So, f n and n to the power log a base b the growth of f n and n to the power log a base b are similar. So, this and this grow at similar rate. So, say how we can represent this similar in a asymptotic sense.

So that means, if we can write this f n is equal to big theta of n to the power log a base b then they are similar because if we this means what this means there is 2 constant positive constant. So, f n is bounded both side by this so that means, they are excitedly similar. Now here we are allowing they are they could be differ in the log factor because log n is small quantity like in terms of n because if say n is 2 to the power 10 then what is log n? Base 2 is basically 10. So, 10 is very less than compare to 2 to the power 10. So, log n is quite less number compare to n.

So, in that sales we are allowing then will differ in a log factor way or power log factor. So, if this is a if they differ in the log factor way we can multiply this with log n log n base 2 or power of log factor. So, k, so this the general term so; that means, f n will be log n base 2 to the power k ok.

(Refer Slide Time: 11:17)



Case-II : $f(n)$ and $n^{\log_b^a}$ grow at similar rate.

$f(n) = \Theta\left(n^{\log_b^a} (\log_2^n)^k\right)$

$k = 0, 1, 2, \ldots$

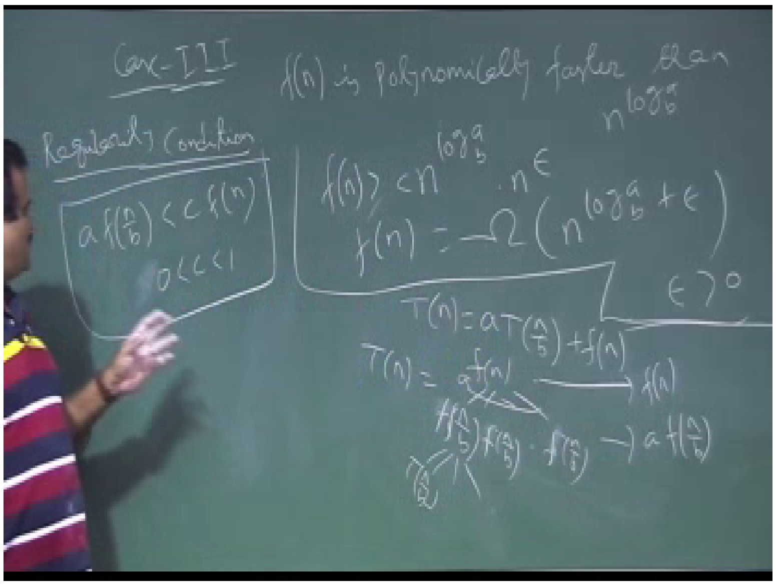
Solution : $\boxed{T(n) = \Theta\left(n^{\log_b^a} (\log_2^n)^{k+1}\right)}$

So, here this k could be if k is 0 then they are exactly their growth is exactly similar there is n log factor, but if k is 1 then there is log factor difference that is allow because log n is less than very much less than n ok.

So, then this is the case 2 then what is the solution? Then the solution will be. So, this T n is equal to big theta of n to the power log a base b and this log factor term will become to the power k plus 1. So, this is the solution for case 2. So, so if k is equal to 0 then we have only log n because k 0 means we have only log n here. So, we will take some example there. So, this we have to remember this is case 2 of master method. So, in this case their growth are similar now case 3 there is only one case left.

So, we have seen one is more one this we have seen this f n is less we have seen this f n and this n to the power log a base b are same or similar growth. Now f n is bigger f n is faster. So, again we have to use that polynomically faster way. So, let us just write that third case. So, this is the third case.

(Refer Slide Time: 13:08)


Case-III
f(n) is Polynomically faster than $n^{\log_b a}$
Regularity Condition
$a f(\frac{n}{b}) \le c f(n)$
$0 < c < 1$
$f(n) \ge c n^{\log_b a} \cdot n^{\epsilon}$
$f(n) = \Omega(n^{\log_b a + \epsilon})$
$\epsilon > 0$
$T(n) = aT(\frac{n}{b}) + f(n)$
$T(n) =$ f(n) → f(n)
$f(\frac{n}{b})\ f(\frac{n}{b}) \cdot f(\frac{n}{b})$ → $a f(\frac{n}{b})$

So, third case is f n is polynomically faster than n to the power log a base b. Now how we can denote the polynomical faster in the asymptotic notation.

So, we can write so; that means, if we multiply a polynomial term with this n to the power epsilon then also f n is faster then also f n is greater. So, f n is greater than c of this so; that means, f n is basically this is big omega n to the power log a base b plus epsilon where epsilon is greater than 0. So, this is the way we represent in a asymptotic notation this is the meaning of f n is polynomically faster than n to the power log a base b ok.

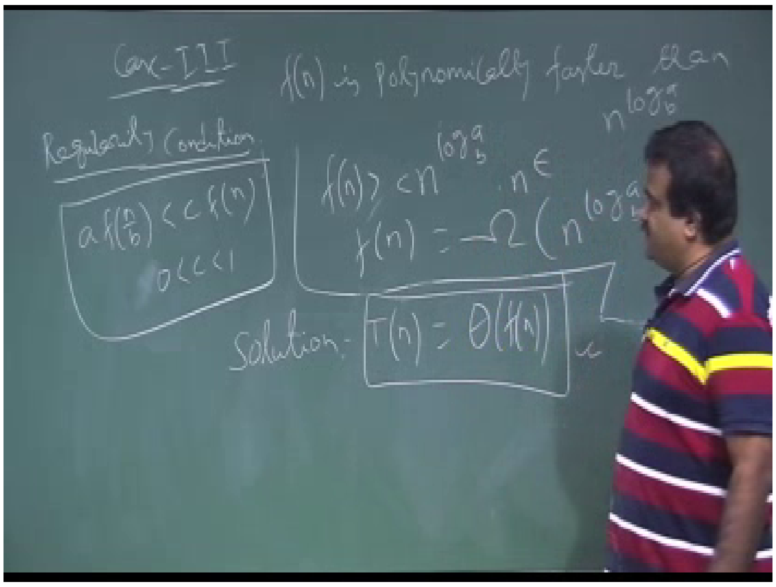
So, here in this case we have another double check we have another condition to check that is called regularity condition. So, this is a extra check point regularity condition this is only they are in the third case. So, how we can get this regularity condition? So, let us just draw. So, what is our recurrence? So, just this is their now what is our recurrence our recurrence is T n is equal to a T n by b plus f of n so that means, we have a problem of size n we divide the problem into n means sub problems each of size n by b and this is the f n is the cost for combining this.

So, now if we draw the recursive tree, T n is basically. So, f n is the merge cost we have T n by b, T n by b, T n by b, and this is a menu. Now again further we can divide this into this is the problem of size n by b we can further divided into sub problem of size n by b square by this and this will be the f of n by b. So, all of this will be f of n by b. So, our total cost is basically sum of the. So, this level sum is f n and this level sum is a f of n by b. So, first level sum is f n. So, we want f n should be more a f n should be bigger so; that means, it should be bigger than at least the next level.

So, it should be much more bigger than this so; that means, if you take a fraction of it f n then also this must be lesser so that means, this is the regularity condition so; that means, a of f n by b must be less than equal to c of f n for and this c must be lies between we must get a c; that means, if you take a fraction of if you take a fraction of this then also f of n by b must be less than that. So, that is the extra check point. So, that is the regularity condition if this is more this is more means if you take a fraction of this then also the next level cost will be lesser than that. So, that is that means, f n is really dominating.

So, this is sort of extra checking this is called regularity condition. So, in that case the solution is. So, this is the case 3.

(Refer Slide Time: 17:05)



Case-III

f(n) is Polynomially faster than $n^{\log_b a}$

Regularity Condition

$a f(\frac{n}{b}) < c f(n)$

$0 < c < 1$

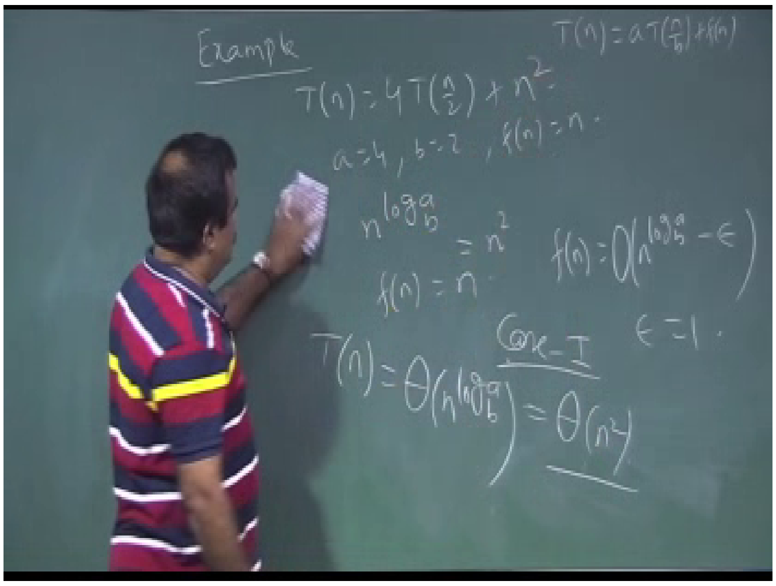
$f(n) > c n^{\log_b a} . n^{\epsilon}$

$f(n) = \Omega(n^{\log_b a}$

Solution:- $\boxed{T(n) = \Theta(f(n))}$

So, if we have case three then the solution will be. So, solution is which dominating f n is dominating f n. So, this is the solution for case 3 so, but case three we have to remember we have a extra condition regularity condition so.

(Refer Slide Time: 17:42)


Example
$T(n) = aT(\frac{n}{b}) + f(n)$
$T(n) = 4T(\frac{n}{2}) + n^2$
$a = 4, b = 2, f(n) = n$
$n^{\log_b a} = n^2$
$f(n) = n$
$f(n) = O(n^{\log_b a - \epsilon})$
$\epsilon = 1$
Case-I
$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

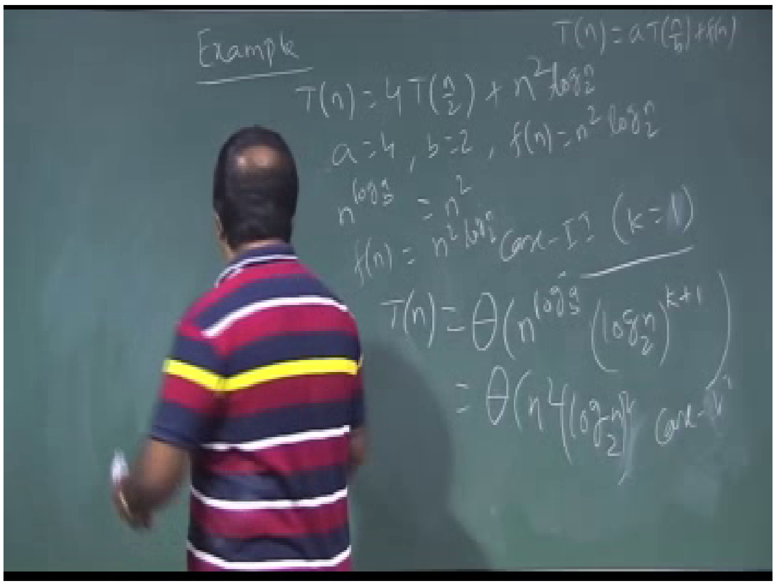
Now, we will take some example of these cases. So, we will take some example. So, let us take. So, if we take this T n to be 4 T n by 2 plus n, we want to use master method on this recurrence.

So, what is the general form of master method? T n is equal to a T n by 2 n by b plus f n. So, what is a over here for this example a is 4 b is 2 and f n is n. So, now, we need to compute that n to the power log a base b. So, what is the n to the power log a base b? So, this is basically n to the power this is a n square. So, log a base b, b is 2 this is 4, 2 square. So, this is 2 basically. So, this is n square and what is a f n? F n is a n. So, which case which dominating n square is more n square is polynomically bigger than n.

So; that means, f n is polynomically slower than n to the power log n a base b so that means, we can write f n is basically big o of n to the power log a base b minus epsilon here epsilon is one case one for master method. So, case one of master method case one means what is the solution. So, T n is basically big theta of n to the power log a base b. So, what is this? Big theta of n square. So, this is the solution for this recurrence with the help of master method now will check the case 2. So, so case 2 for case 2 will take the recurrence n recurrence may be you can change it to the square ok.

So, just rub it. So, a is 4 b is 2, but here f n is n square.

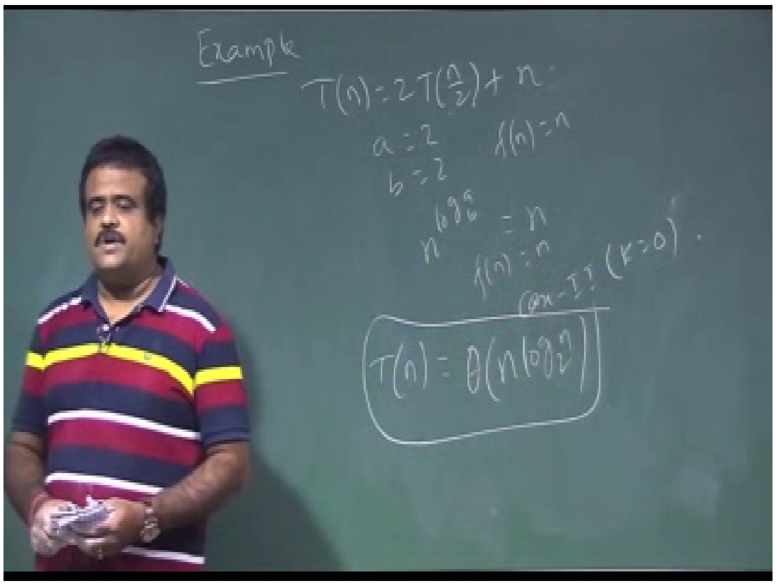
(Refer Slide Time: 20:08)


Example
$T(n) = aT(\frac{n}{b}) + f(n)$
$T(n) = 4T(\frac{n}{2}) + n^2 \log_2^n$
$a = 4, b = 2, f(n) = n^2 \log_2^n$
$n^{\log_b^a} = n^2$
$f(n) = n^2 \log_2^n$ Case-II? $(k = 1)$
$T(n) = \Theta(n^{\log_b^a} (\log_2^n)^{k+1})$
$= \Theta(n^2 (\log_2^n)^2)$ Case-II

Now, what is n to the power log a base b, n to the power log a base b is basically n square. So, now, this is which case now f n is also n square. So, they are exactly similar. So, this is case 2 with k is equal to 0 they are exactly similar, there is no log factor there no logarithm term is involve there. So, case 2 with k is equal to 0 then what is the solution? Solution is basically T n is equal to big theta of n to the power log a base b into log n base 2 to the power k plus 1. So, here k is 0.

So, it will be just log n. So, this is basically this is n square log n base 2. So, this is the solution of this recurrence. Now if we have a log n over here base 2 then this is this is also case 2 this is if we have a log n over here this is also case 2 with k is equal to 1 and in that case solution would be this square. So, this is basically all of this basically case 2 yeah case 2 with particular value of k now we check the merge sort recurrence.

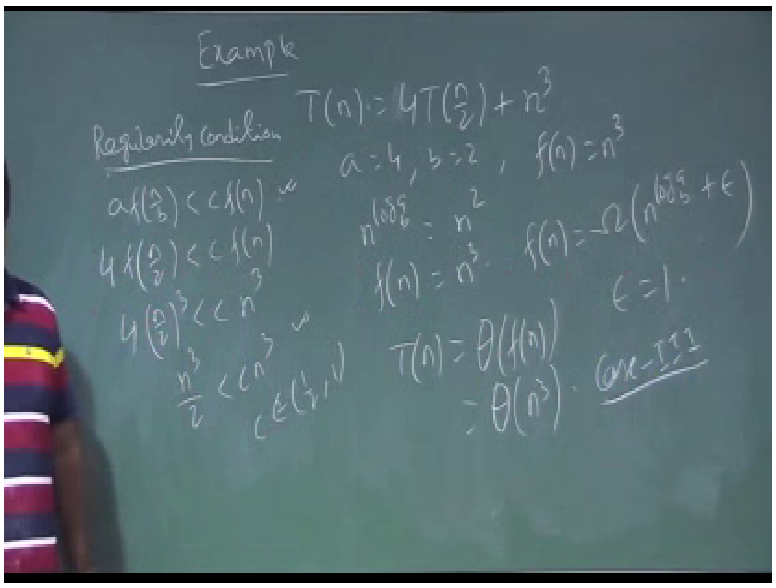
(Refer Slide Time: 21:55)



So, merge sort recurrence is T n is equal to 2 T n by 2 plus n or c of n. So, here a is equal to 2 b is equal to 2 f n is equal to n.

So, what is n to the power log a base b? It is basically n and f n is also n. So, which case case 2 with k is equal to 0. So, case 2 of master method with k is equal to 0 then what is the solution? Solution is basically big theta of n to the power. So, log a base b. So, this is the solution we know for merge sort. Now we will talk about case three we take an example which will feed on the case 3.

(Refer Slide Time: 22:53)


Example
$T(n) = 4T(\frac{n}{2}) + n^3$
$a = 4, b = 2, f(n) = n^3$
$n^{\log_b^a} = n^2$
$f(n) = n^3$
$f(n) = \Omega(n^{\log_b^a + \epsilon})$
$\epsilon = 1$
$T(n) = \theta(f(n))$
$= \theta(n^3)$
Case-III
Regularity condition
$af(\frac{n}{b}) < cf(n)$ ✓
$4f(\frac{n}{2}) < cf(n)$
$4(\frac{n}{2})^3 < cn^3$
$\frac{n^3}{2} < cn^3$ ✓
$c \in (\frac{1}{2}, 1)$

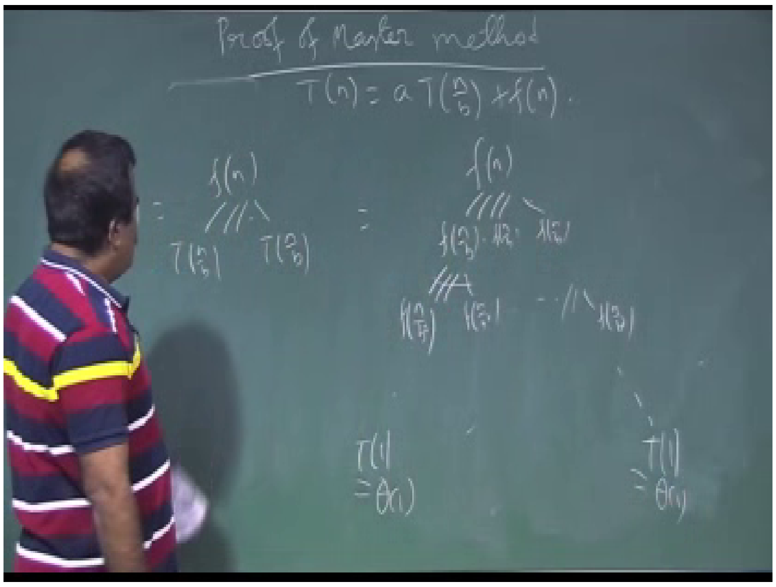
So, if we take this recurrence T n is equal to 4 T n by 2 plus say n cube. So, here a is 4 b is 2, f n is n cube. So, n to the power log a base b basically here how much.

So, it is a square and f n is n cube. So, it is basically f n is polynomically faster. So, this is this could be a candid of case 2 case 3, but we need to check the regularity condition. So, this is basically f n is basically big omega of n to the power log a base b plus epsilon. So, here epsilon is 1. So, this could be a candid of case three provided we convince with the regularity condition. So, just let us check the regularity condition. So, this is the extra check point for the case this case regularity condition. So, this condition is telling we should have a of f n by b is less than sum c of f n c of f n.

So, we should get a c which is lies between 0 to 1 such that this will satisfy. So, a is 4, a for b is 2 n by 2 is less than yeah we should get a c. So, this is basically f n now f n is n cube. So, we will just write this 4 n by 2 cube less than c of n cube. So, this is basically n cube by 2 less than c of n cube. So, for which one is of c this will occur. So, if we choose a any value c lies between half and one then this will satisfied. So, the regularity condition is also satisfying. So, then what is the solution? Solution is basically big o of f n big theta of f n. So, order of n cube.

So, this is the solution for this, this is by case this is the case three of the master method, but we have to be careful regarding this extra check which is the regularity condition ok.

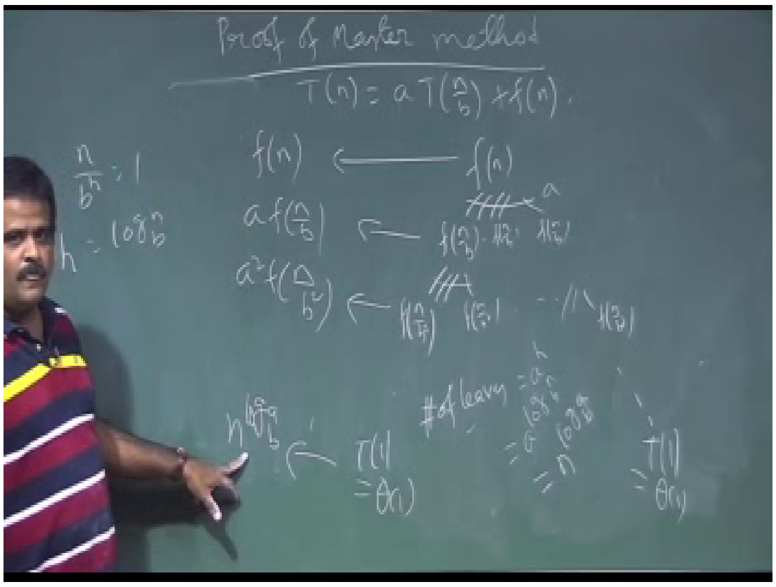
(Refer Slide Time: 25:38)


Proof of Master method
$T(n) = aT(\frac{n}{b}) + f(n)$.
$f(n)$
$T(\frac{n}{b})$ $T(\frac{n}{b})$
$f(n)$
$f(\frac{n}{b})$ $f(\frac{n}{b})$
$f(\frac{n}{b^2})$ $f(\frac{n}{b^2})$ $f(\frac{n}{b^2})$
$T(1) = \Theta(1)$
$T(1) = \Theta(1)$

So, now, we will talk we will see rough proof of this master method why this how this case is are coming; proof of master method. So, this proof i d is coming from the if you draw the recursive proof. So, this is basically the recurrence T n by b plus f n. So, we tried to draw the recursive tree. So, T n is basically f n and then we have T n by b, T n by b. So, this we continue.

So, f n. So, each of this is f n by b and each of this again is basically f of n by b square like this. So, this way you will continue until we reach to T 1. So, this is also f of n by b dot dot dot. So, f of n by b. So, all are f of n by b square like this, f of n by b square like this. So, all the branch will stop at T 1 when the size is one. So, T 1is basically theta of one. So, now, the cost is basically total cost, now this is the now what is the height of this tree? If the height is h then the height is coming like this n to the n by b to the power h is one.

(Refer Slide Time: 27:12)



So, height is basically log n base b, now then what is the number of leafs? Number of leafs is basically.

So, at this level how many notes are there n an a at this level a a a square. So, there the; that means, at the level h the number of notes will be that is the leaf number of height. So, a to the power h. So, this is nothing, but a to the power log of n base b. So, this is our n to the power log a base b which we have comparing with f n. So, if you take the sum this is f n this is basically a of f n by b, this is a square f of n by b square dot dot dot then this will be n to the power log a base b.

So, this is the sum now the case one is basically this is more. So, this is geometrically this is more so that means, the solution will be this one and case 2 is the they are similar case 2 with k is equal to 0 means they are similar. So, all the level they have similarity similar value of this.

So, the total cost will be these into height. So, height is this one. So, this is basically we can make it base two. So, that will be cost and factor again. So, that is why for case 2 k is equal to 0 is coming by there. So, this into log of the height I mean height the log of 2; that is why the case 2 and case 3 is this is dominating. So, that is why this is more even this is more than this. So, that is why solution is this. So, this is the rough proof of the master method, but in the book in our text book Cormen.

We have details proof. So, by the induction method, if you in case you can have a loop.

Thank you.