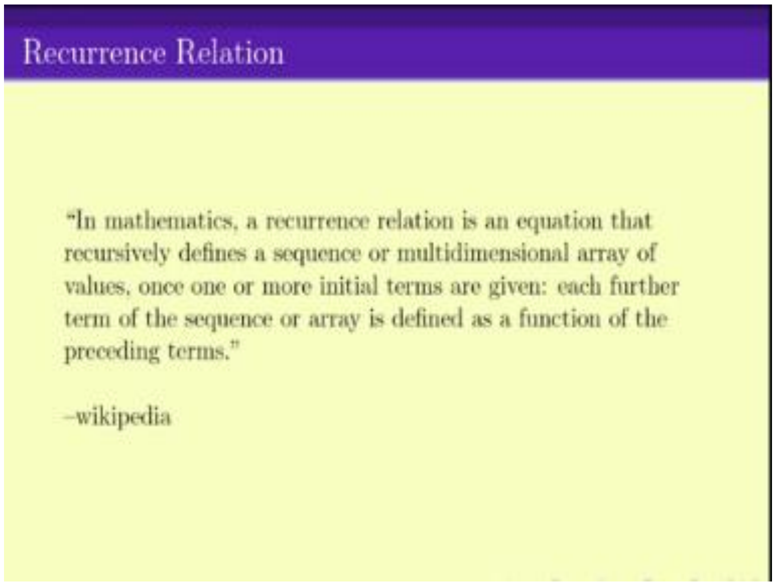
Discrete Mathematics

Prof. Sourav Chakraborty
Department of Mathematics
Indian Institute of Technology - Madras

Lecture 43
Asymptotic Relations (Part 4)

Welcome back. So we have been looking at recurrence relations and how to solve recurrence relation.

(Refer Slide Time: 00:06)



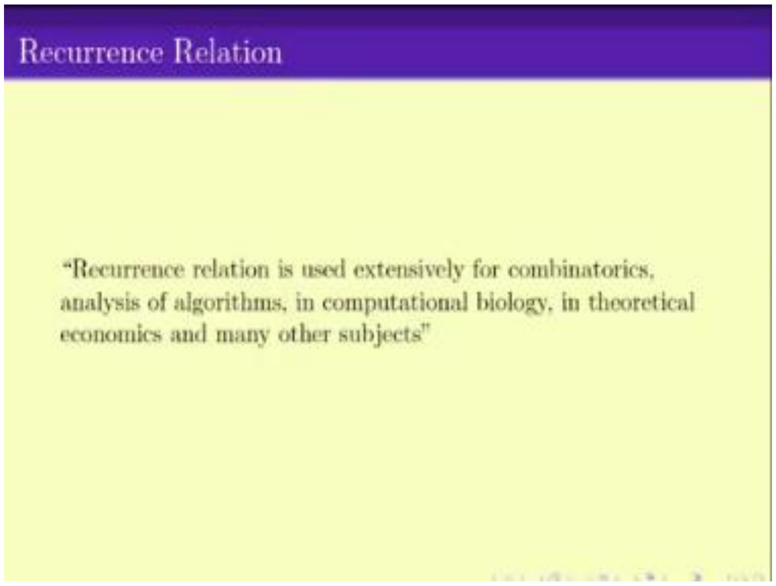
Recurrence Relation

"In mathematics, a recurrence relation is an equation that recursively defines a sequence or multidimensional array of values, once one or more initial terms are given: each further term of the sequence or array is defined as a function of the preceding terms."

–wikipedia

So, we have seen, we know what recurrence relation is, it is a sequence of numbers while we have been given the initial set of value and the nth term is determined as a function of the earlier terms.

(Refer Slide Time: 00:18)

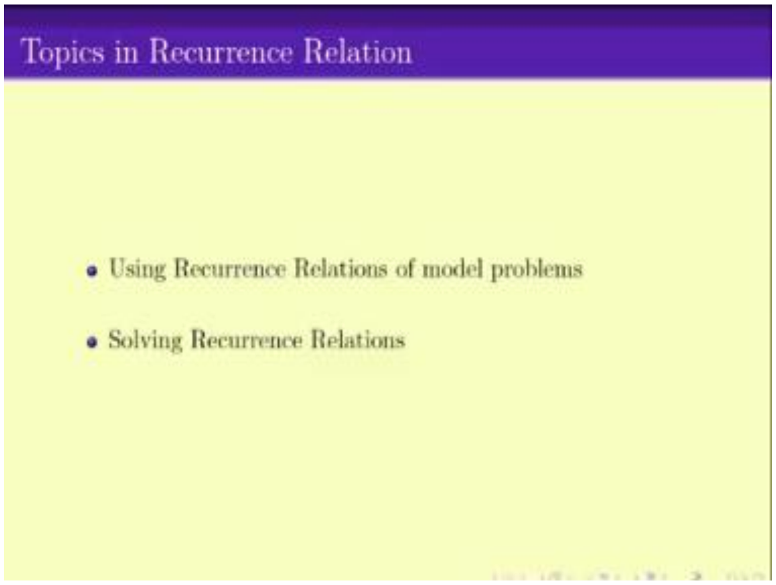


Recurrence Relation

"Recurrence relation is used extensively for combinatorics, analysis of algorithms, in computational biology, in theoretical economics and many other subjects"

Recurrence relations is used extensively for various topics in math and other related subjects.

(Refer Slide Time: 00:28)

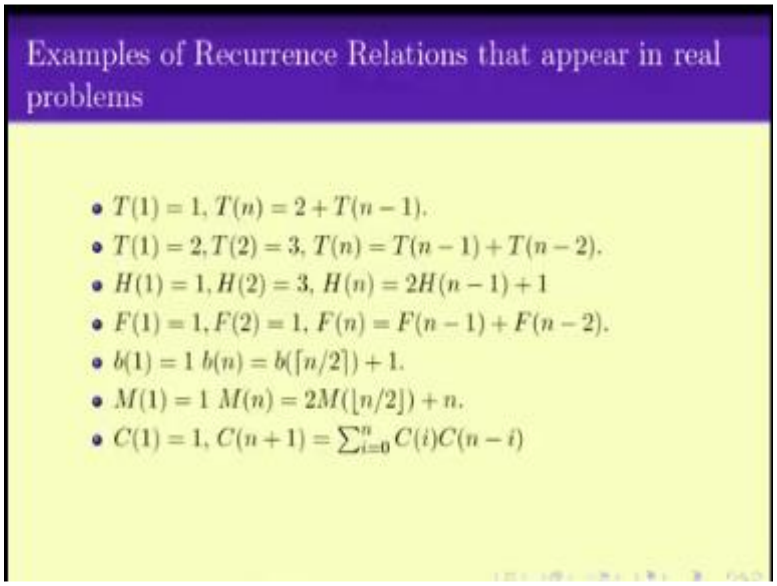


Topics in Recurrence Relation

- Using Recurrence Relations of model problems
- Solving Recurrence Relations

We have seen how recurrence relation can be used to model various problems particularly counting problems and we have been trying to see how to solve recurrence relations.

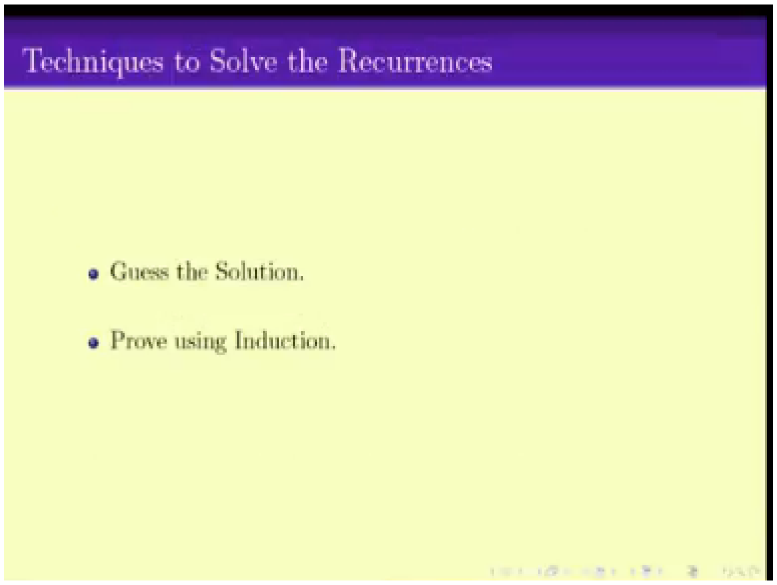
(Refer Slide Time: 00:43)


Examples of Recurrence Relations that appear in real problems

- $T(1) = 1, T(n) = 2 + T(n-1)$.
- $T(1) = 2, T(2) = 3, T(n) = T(n-1) + T(n-2)$.
- $H(1) = 1, H(2) = 3, H(n) = 2H(n-1) + 1$
- $F(1) = 1, F(2) = 1, F(n) = F(n-1) + F(n-2)$.
- $b(1) = 1$ $b(n) = b(\lceil n/2 \rceil) + 1$.
- $M(1) = 1$ $M(n) = 2M(\lfloor n/2 \rfloor) + n$.
- $C(1) = 1, C(n+1) = \sum_{i=0}^{n} C(i)C(n-i)$

Now, here are some of the examples that we have looked into and the question that always comes up is how to solve recurrence relation?

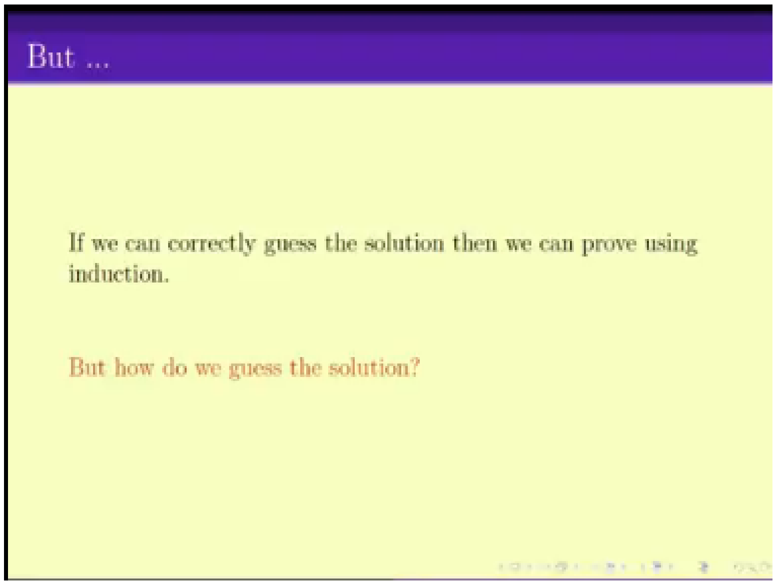
(Refer Slide Time: 00:56)


Techniques to Solve the Recurrences

- Guess the Solution.
- Prove using Induction.

So, till now, we have seen a few techniques of solving recurrence relation, the first one is guess the solution and they prove it using induction.

(Refer Slide Time: 01:11)



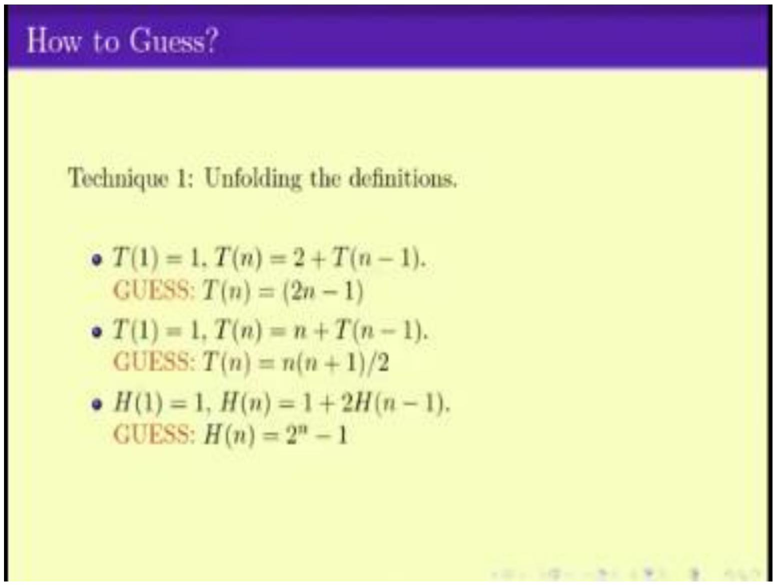
But ...

If we can correctly guess the solution then we can prove using induction.

But how do we guess the solution?

Now, if there exists a very nice case, a nice solution then possibly one can guess it and solve it by induction. Once you guess it correctly solving it by induction is quite a standard technique but how do you guess the solution? Now, again there are some techniques of guessing solution.

(Refer Slide Time: 01:28)



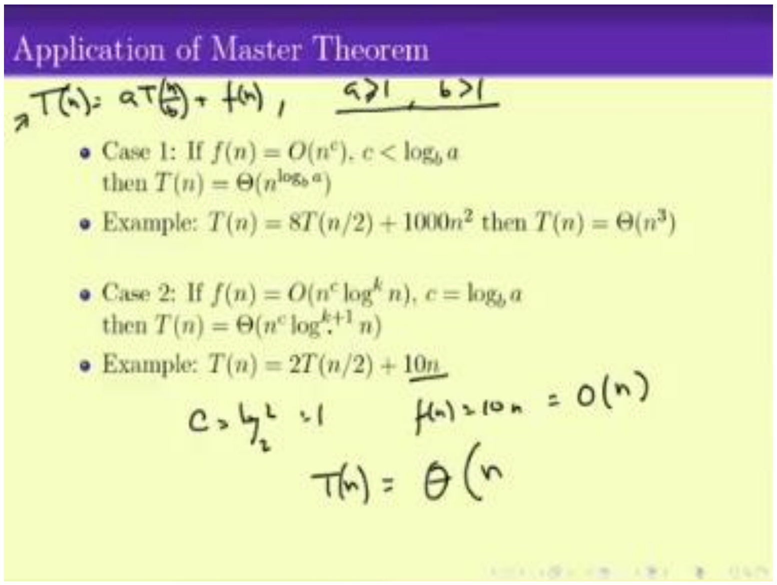
How to Guess?

Technique 1: Unfolding the definitions.

- $T(1) = 1$, $T(n) = 2 + T(n-1)$.
  GUESS: $T(n) = (2n - 1)$
- $T(1) = 1$, $T(n) = n + T(n-1)$.
  GUESS: $T(n) = n(n+1)/2$
- $H(1) = 1$, $H(n) = 1 + 2H(n-1)$.
  GUESS: $H(n) = 2^n - 1$

The first technique that we looked at was unfolding the definition, namely you keep on just unfolding the function, the expression possibly Tn = 2 + Tn of n – 1, you write Tn - 1 as 2 + Tn - 2 and in that case, you get Tn = 4 + Tn - 2 and so on. You keep on doing and at some point of time, you will be able to substitute after some ith iteration, you will be able to get the thing into a proper form and we have seen how this technique works and helps us to guess for recurrences like this.

(Refer Slide Time: 02.10)



Application of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$, $\underline{a \geq 1,\ b > 1}$

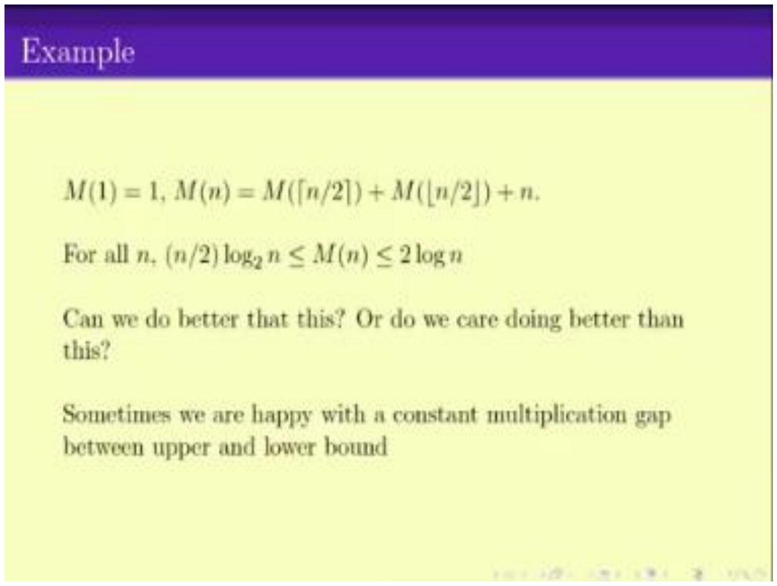
- Case 1: If $f(n) = O(n^c)$, $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
- Example: $T(n) = 8T(n/2) + 1000n^2$ then $T(n) = \Theta(n^3)$
- Case 2: If $f(n) = O(n^c \log^k n)$, $c = \log_b a$ then $T(n) = \Theta(n^c \log^{k+1} n)$
- Example: $T(n) = 2T(n/2) + \underline{10n}$

$c = \log_2 2 = 1$ $\quad f(n) = 10n = O(n)$

$T(n) = \Theta(n$

But now there are recurrences which are of this form Fn = Fn - 1 + Fn - 2 the Fibonacci number recurrence and guessing it is complicated because the actual formula is very complicated and the other kind of stuff that there is when you have things like Bn = Bn over 2 + 1 and because of the things like ceil, ceiling and floor and so on. It is very hard to get a clean, neat, simple formula for Bn and in that case, there is no guess that can exist.

(Refer Slide Time: 02:58)



Example

$M(1) = 1$, $M(n) = M(\lceil n/2 \rceil) + M(\lfloor n/2 \rfloor) + n$.

For all n, $(n/2)\log_2 n \leq M(n) \leq 2\log n$

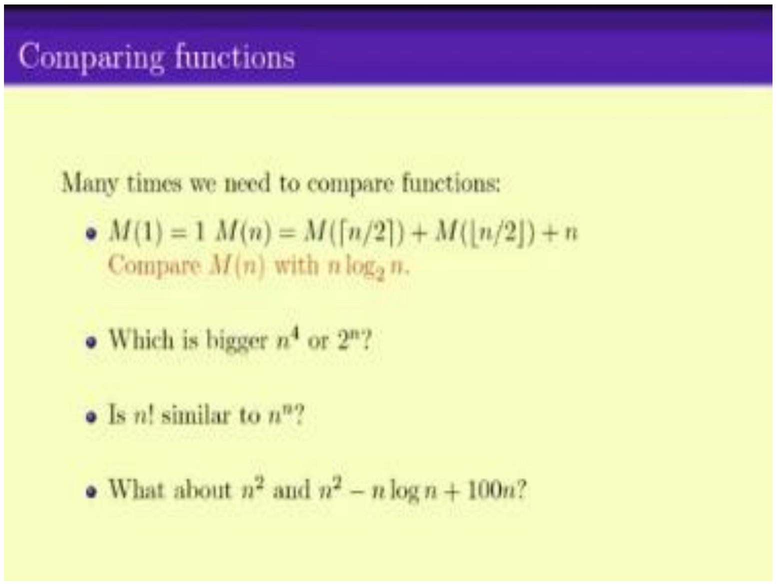
Can we do better that this? Or do we care doing better than this?

Sometimes we are happy with a constant multiplication gap between upper and lower bound

So, we saw that in such an example, for example, this Mn = Mn over 2 + n over 2 + n where the first one is a floor, second one is the ceiling. We instead of getting the exact solution can get an upper bound and the lower bound but upper and lower bound differ by some constant factor but that is possibly good enough for us. Sometimes, we are happy with a constant multiplication gap between the upper and lower bound.

Now, once you have something like this, you would like to say something like Mn is actually, something like n log n to formulae put it, we basically kind of compare the function M n versus n log n.

(Refer Slide Time: 03:45)



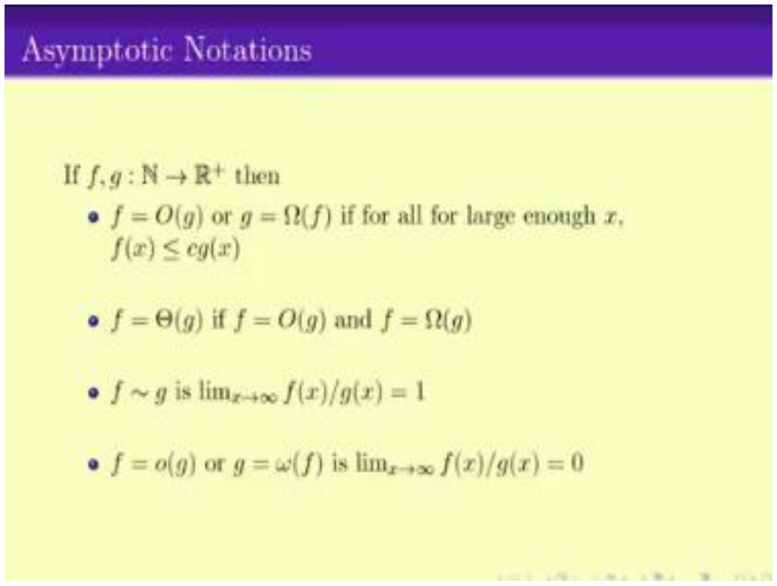
Comparing functions

Many times we need to compare functions:

- $M(1) = 1$ $M(n) = M(\lceil n/2 \rceil) + M(\lfloor n/2 \rfloor) + n$
  Compare $M(n)$ with $n \log_2 n$.
- Which is bigger n^4 or 2^n?
- Is $n!$ similar to n^n?
- What about n^2 and $n^2 - n \log n + 100n$?

And this kind of took us to a different course or different tool of how to compare functions. So how to compare Mn with n log n? How to compare n to the power 4 with 2 power n? How to compare n factorial with n power n and so on and so forth.

(Refer Slide Time: 04:12)



Asymptotic Notations

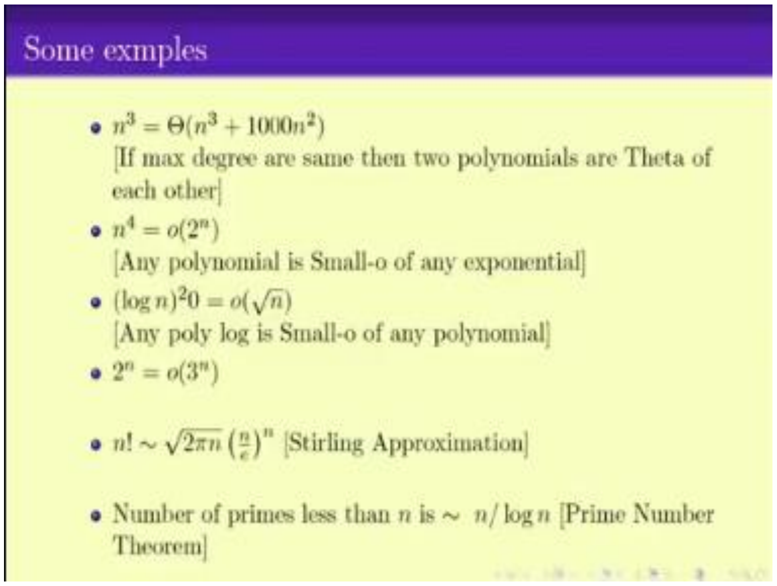
If $f, g : \mathbb{N} \to \mathbb{R}^+$ then

- $f = O(g)$ or $g = \Omega(f)$ if for all for large enough x, $f(x) \leq cg(x)$
- $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$
- $f \sim g$ is $\lim_{x \to +\infty} f(x)/g(x) = 1$
- $f = o(g)$ or $g = \omega(f)$ is $\lim_{x \to +\infty} f(x)/g(x) = 0$

And what we have seen is that there is this whole thing called an asymptotic notation that helps us to get a hold on how this function behaves. In short, f is equals to big-O of g, if f is less than constant times g is also told that g is equals to big Omega of f. If f is big-O of g and f is big-omega of g that if f is upper bounded and lower bounded by some constant time multiples of g. Then, we say f is theta of g and we are something like the asymptotic similar notation and the small notation which we saw and is extensively used in the literature.

So, it is very important to know be very familiar with this set of notations and also (()) (05:08) important to know how the various useful functions relate to each other.

(Refer Slide Time: 05:15)



Some exmples

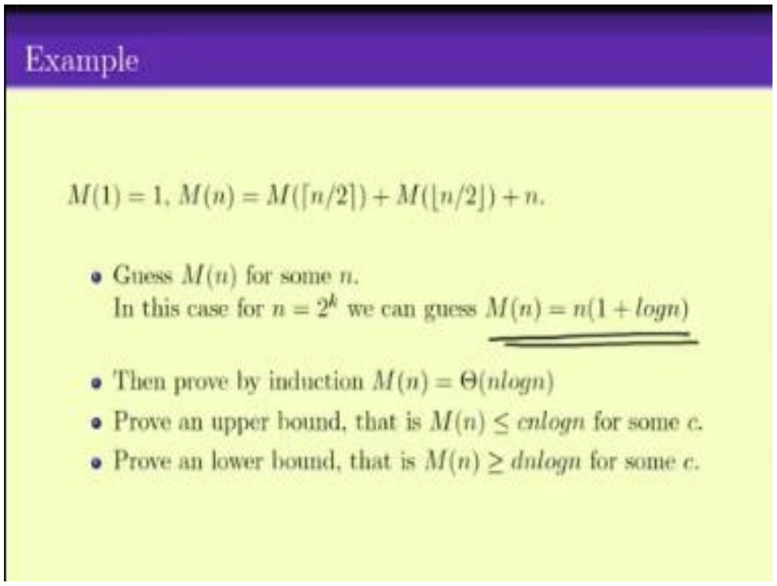
- $n^3 = \Theta(n^3 + 1000n^2)$
  [If max degree are same then two polynomials are Theta of each other]
- $n^4 = o(2^n)$
  [Any polynomial is Small-o of any exponential]
- $(\log n)^2 0 = o(\sqrt{n})$
  [Any poly log is Small-o of any polynomial]
- $2^n = o(3^n)$
- $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ [Stirling Approximation]
- Number of primes less than n is $\sim\ n/\log n$ [Prime Number Theorem]

For example, if you have two polynomials, then if you want compare them, all that matters is what is the coefficient of the largest degree? what is the larger degree and what is this coefficient and that helps us understand how (()) (05:32) largest degree is same that they are theta of each other. If the largest degree is same and the coefficient of largest degree is same then we have same of each other.

Similarly, we have things like of any polynomial is small-o of any exponential or in other words, as n goes to infinity, any polynomial is way with smaller than any exponential function. Similarly, any polynomial of a log, logarithmic function is small-o of any polynomials, n power square root n, and whatever you want to put it. So log n power 20, log n power 30 is equals to small-o of square root n.

We have things like 2 power n is small-o of 3 power n and we have some other expressions which are used for which we have some nice, cute expression which are much more handy. For example, n factorial is asymptotically similar to 2 square root 2 pi n, n over e by a power n and so on. Now that we know how to compare functions and now we have understood, we have got the language of how to write something like Mn is close to n log n and so on, so forth.

(Refer Slide Time: 07:05)



Example

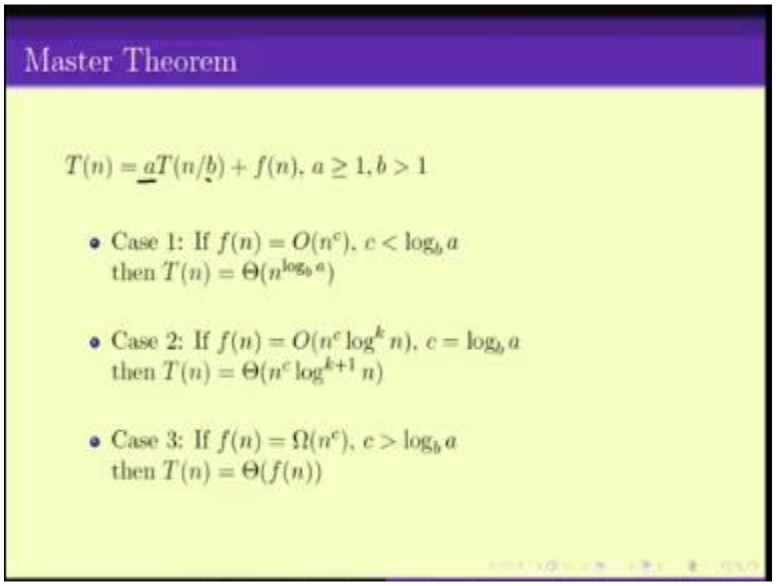
$M(1) = 1$, $M(n) = M(\lceil n/2 \rceil) + M(\lfloor n/2 \rfloor) + n$.

- Guess $M(n)$ for some n.
  In this case for $n = 2^k$ we can guess $\underline{M(n) = n(1 + logn)}$
- Then prove by induction $M(n) = \Theta(nlogn)$
- Prove an upper bound, that is $M(n) \leq cnlogn$ for some c.
- Prove an lower bound, that is $M(n) \geq dnlogn$ for some c.

Question is that how do we solve such a function, the recurrence? Now, the trick for that is first of all, you have to guess Mn for some n. You would get a feel for it. In the last video, we saw a few more examples of the same kind. The idea is to get a feel for it. So here, for example, if you put n to be power of 2 then, you can guess that Mn equals to n times 1 + log n which basically means it is n log n and then instead of trying to solve it exactly, you end up solving Mn equals to theta of n log n.

And you do it by induction by first proving an upper bound of Mn is less than or equal to constant time n log n and then proving a lower bound Mn is greater than some constant times n log n. Now, we have seen the last video, there can be quite a number of different expressions and at least even guessing this term is not necessarily easiest thing in the world. So what I am going to give you today is what is known as a Master theorem, it is a theorem that kind of talks a lot of various taking care of lots of differences.

(Refer Slide Time: 08:36)



Master Theorem

$T(n) = \underline{a}T(n/b) + f(n),\ a \geq 1, b > 1$

- Case 1: If $f(n) = O(n^c)$, $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
- Case 2: If $f(n) = O(n^c \log^k n)$, $c = \log_b a$ then $T(n) = \Theta(n^c \log^{k+1} n)$
- Case 3: If $f(n) = \Omega(n^c)$, $c > \log_b a$ then $T(n) = \Theta(f(n))$

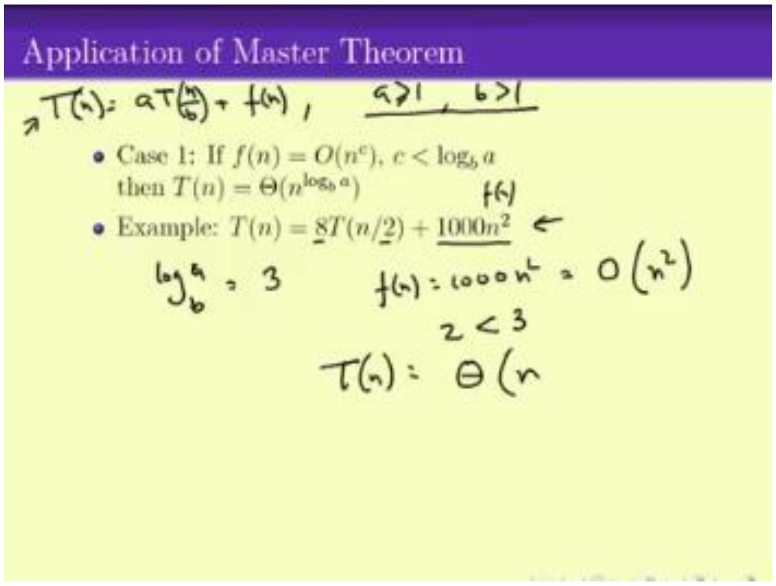
So, Master theorem is that if I have a term which is of the form Tn equals to a times Tn over b + f n where fn is some function of n, say n square, n q, n log n, 2 power n and something like that and if a is strict is greater than or equal to 1 and B is strictly greater than 1 then, first case, if f n is big-O of n power c where c is something less than log, log a base b where a and b are these two numbers then, Tn is actually theta of n power log a base b.

The second case is that if fn is n power c times log n power k and c is exactly equal to log b base a. So log a base b, then Tn equals to n power c log n power k + 1. Okay? And the third case is that if c is bigger than log b, log a base b and fn is omega of n power c then, Tn is theta of fn. So these are 3 cases, I am not going to prove you those three cases but again these are the three cases, I would ask you to go and prove it for yourselves. They are not hard.

Once I step this theorem, you have to just follow the induction technique to solve it. So I will give it as an exercise solve, I am sorry, not solve whereas prove the be master theorem. It will be useful for you to get a feel of this problem. This is a very generic theorem for solving. So instead of solving it, I will see how one can apply Master theorem to some of the recurrences that arrives.

The master theorem is a bit complicated to figure out what is going on because of this log a base b and so on and so forth. But we will see how can one can attack them.

(Refer Slide Time: 11:29)



Application of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$, $\underline{a \geq 1, b > 1}$

- Case 1: If $f(n) = O(n^c)$, $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
- Example: $T(n) = \underline{8}T(n/\underline{2}) + \overset{f(n)}{\underline{1000n^2}} \leftarrow$

$\log_b a = 3$ $\quad f(n) = 1000n^2 = O(n^2)$

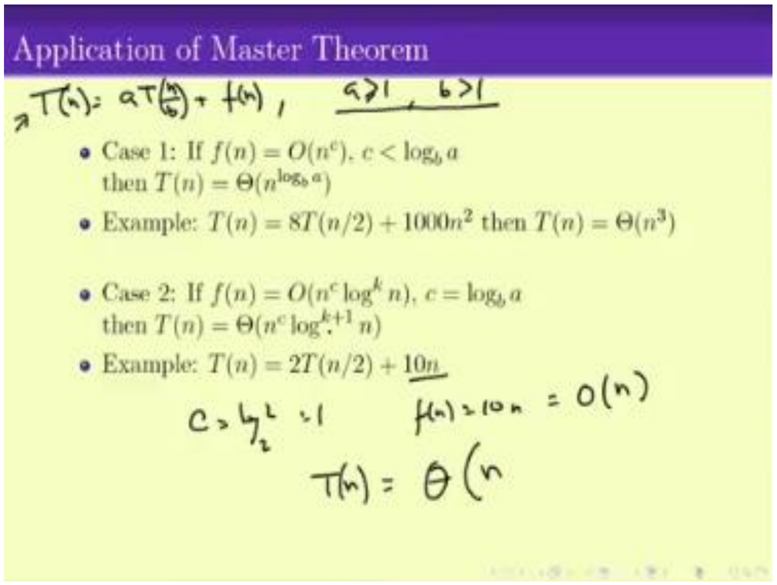
$2 < 3$

$T(n) = \Theta(n$

So, say here in the first case, so let me just write down here and so it was Tn = a Tn over b + fn where a was greater than equal to 1 and b is strictly greater than 1 and what it says is that f n is ordered n power c and c is less than log a base b then, Tn is equal to theta of n log a base b. Look at this example, Tn equals to 8 times n over 2 + 1000 n square. Now, here 8 is equals to a, 2 equals to b, clearly satisfies this expression.

It has the same maths, the same thing as this one where this number is of course fn that we are talking about and what is the log a base b? Log a base b is log 8 base 2 which is 3 and fn is 1000 n square which is of course less than n cube, Right? So correct? So actually sorry, I mean this is less than n square, I mean order of n square and 2 is strictly less than 3. So in fact, guess one fits perfectly and in that case, we can say that Tn equals to theta of n power log b, log a base b which is 3.

(Refer Slide Time: 13:55)



Application of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$, $\underline{a>1,\ b>1}$

- Case 1: If $f(n) = O(n^c)$, $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
- Example: $T(n) = 8T(n/2) + 1000n^2$ then $T(n) = \Theta(n^3)$
- Case 2: If $f(n) = O(n^c \log^k n)$, $c = \log_b a$ then $T(n) = \Theta(n^c \log^{k+1} n)$
- Example: $T(n) = 2T(n/2) + \underline{10n}$

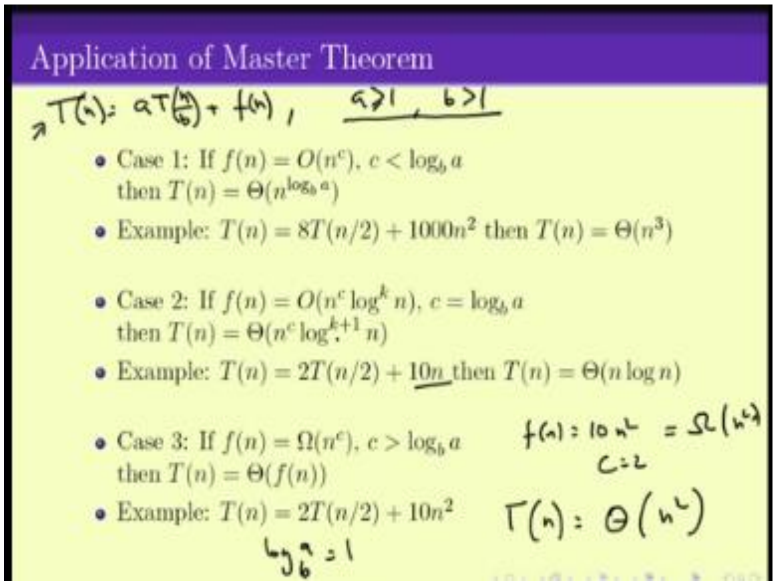
$c = \log_2 2 = 1$ $\quad f(n) = 10n = O(n)$

$T(n) = \Theta(n$

So, by applying this in a master theorem, we can easily get that Tn equals to theta of n cube. This also - It is a theorem, you do not really have to prove anything, once you have a proof of the theorem. But it also helps us to kind of - kind of guess what the value of Tn is. In this case, it helps us to understand that is n cube. The second one is of course, the case when c equals to log b base a as log a base b and fn equals to n power c log n power k.

Now, note that here, what is c in our expression? Tn equals to 2 times Tn over 2 + 10 n, c equals to log 2 base 2 which is 1 and clearly fn which is 10 n right? This is order of n, I mean it has nothing that there is no factor of law. So by doing so, I have satisfied this case two and hence what will T of a and b? So T of n will be theta of n power c so which is n and there is no log here and that the fold, I will get one more extra log here. So I get Tn equals to theta n log n. Okay?

(Refer Slide Time: 15:40)



Application of Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$, $\underline{a \geq 1, b > 1}$

- Case 1: If $f(n) = O(n^c)$, $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
- Example: $T(n) = 8T(n/2) + 1000n^2$ then $T(n) = \Theta(n^3)$
- Case 2: If $f(n) = O(n^c \log^k n)$, $c = \log_b a$ then $T(n) = \Theta(n^c \log^{k+1} n)$
- Example: $T(n) = 2T(n/2) + \underline{10n}$ then $T(n) = \Theta(n \log n)$
- Case 3: If $f(n) = \Omega(n^c)$, $c > \log_b a$ then $T(n) = \Theta(f(n))$
- Example: $T(n) = 2T(n/2) + 10n^2$

$f(n) = 10n^2 = \Omega(n^2)$

$c = 2$

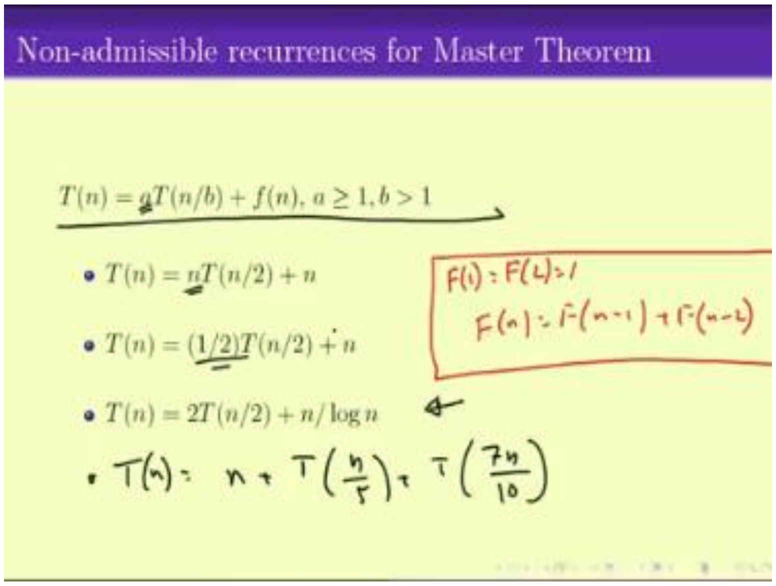
$T(n) = \Theta(n^2)$

$\log_b a = 1$

So, again you can quickly see that we do get Tn equals to theta n log n. Similarly, for the case three, if when I have this expression, Tn equals to n over 2 + 10 n square. Now here fn equals to 10 n square. So and what is c first of all? c is of course, sorry, what is log a base b? log a base b is 2 by log 2 base 2 which is 1 and 10 n square which is clearly bigger than - which is n square which is and this, this is c.

So c equals to 2 and 2 is clearly bigger than 1 and therefore, what I get is that Tn from by applying this Master theorem, I get Tn equals to theta of the fn which is n square, Right? So by just applying this, we get Tn equals to theta n square. So here is a theorem which helps us to get solutions to some class of recurrences very easily and very simply. But, there are some set of recurrences for which this Master theorem will not work.

(Refer Slide Time: 17:34)



Non-admissible recurrences for Master Theorem

$T(n) = aT(n/b) + f(n), a \geq 1, b > 1$

- $T(n) = nT(n/2) + n$
- $T(n) = (1/2)T(n/2) + n$
- $T(n) = 2T(n/2) + n/\log n$
- $T(n) = n + T(\frac{n}{5}) + T(\frac{7n}{10})$

$F(1) = F(2) = 1$
$F(n) = F(n-1) + F(n-2)$

For example, If I have Tn equals to n times Tn over 2 + n will not work because this a needs to be a constant where this is a n or if I have Tn equals to half of Tn over 2 + n is not work because this is a and it needs to be greater than or equal to 1 or this expression, you can say, you can convince yourself that this expression is not something that satisfied in any of the three cases or also the other one that we did which Tn equals to n + Tn over 5 + T 7n over 10.

This is something also does not work because it does not fit in this one. So, in other words, there are recurrences for which at Master theorem will not work and in those cases, we have to go back to our old method and there are recurrences for which this master theorem will work and in which case we have a nice easy example.

But, there are problems for which we all our technique fails, at least the techniques that we have seen till now namely what happens if I have this Fibonacci sequence and how to approach something like a Fibonacci sequence is what we will be doing in the next week's video. We will be focusing - We will be doing something, extremely important which is known as the generating functions. Thank you.