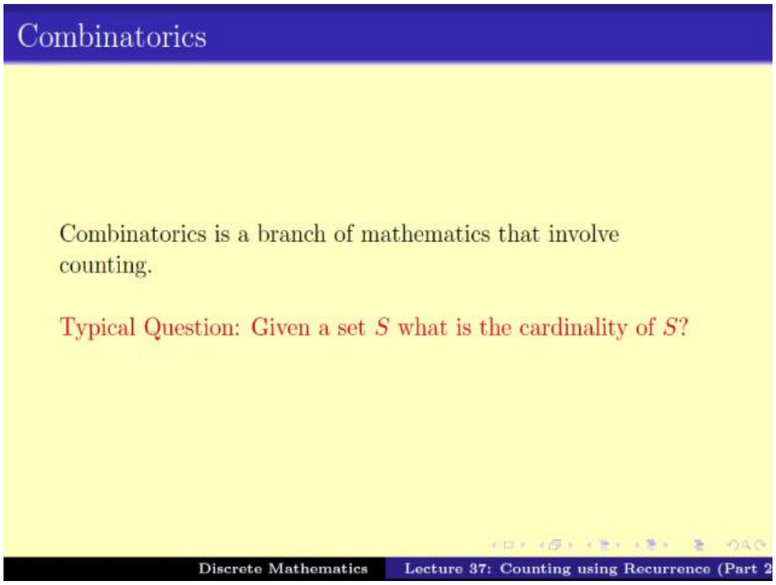
Discrete Mathematics
Prof. Sourav Chakraborty
Department of Mathematics
Indian Institute of Technology – Madras

Lecture - 37
Counting using Recurrence (Part 2)

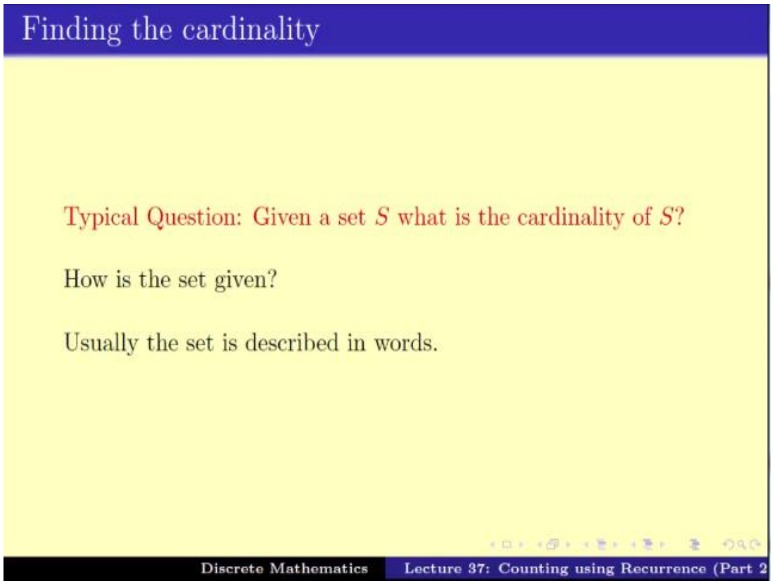
Welcome back. So we have been looking at how to count using Recurrences. So, stepping back a little.

(Refer Slide Time: 00:13)



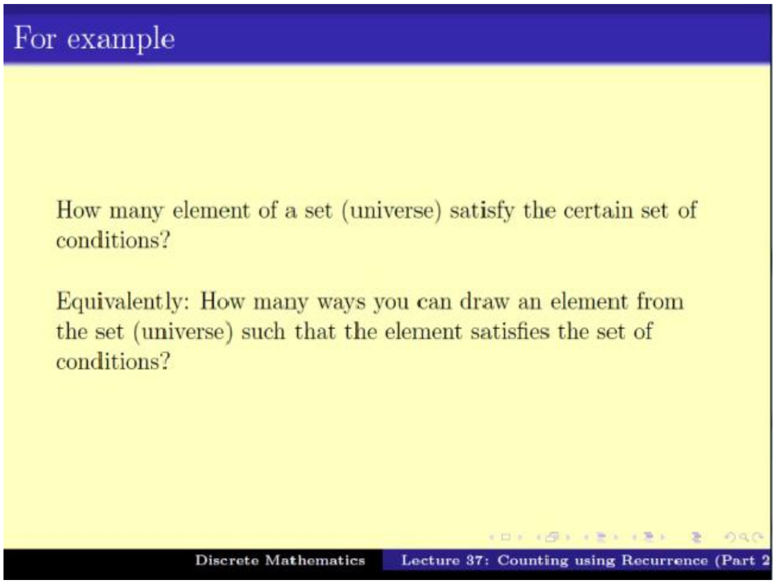
So we have been looking at Combinatorics which is a branch of mathematics involved in counting. And the typical question is how to count the cardinality of the set.

(Refer Slide Time: 00:28)



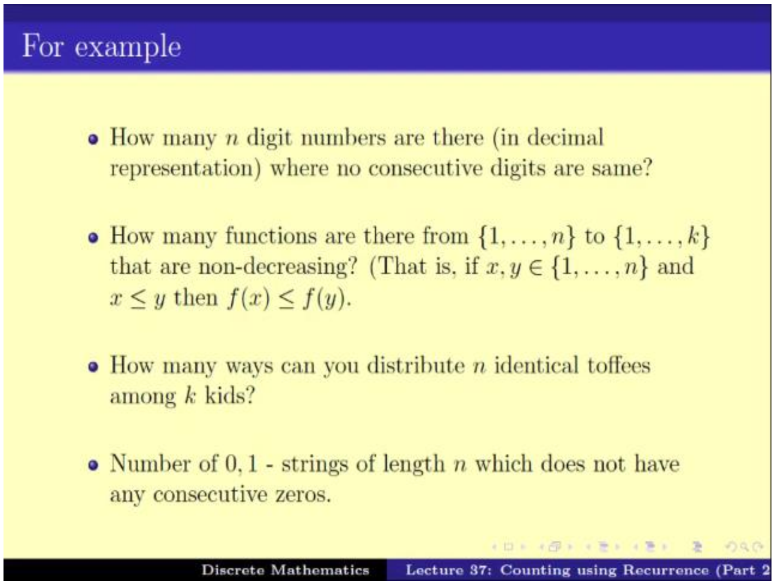
And here we mostly talk about the case where the set is given implicitly and not explicitly. Then the set is described in words and I want to know how many elements in the set.

(Refer Slide Time: 00:43)



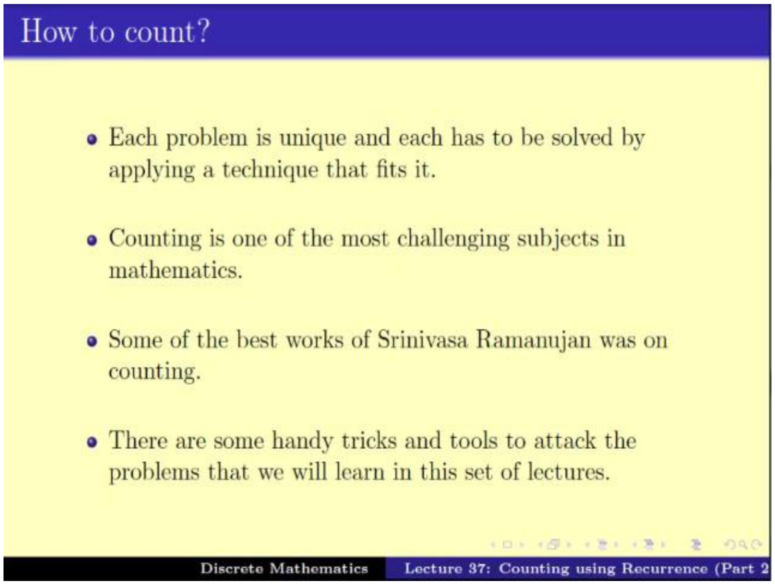
A typical example is how many elements of a universe satisfy a certain set of conditions? Or how many ways can I draw an element from the universe such that satisfy certain set of conditions?

(Refer Slide Time: 01:03)



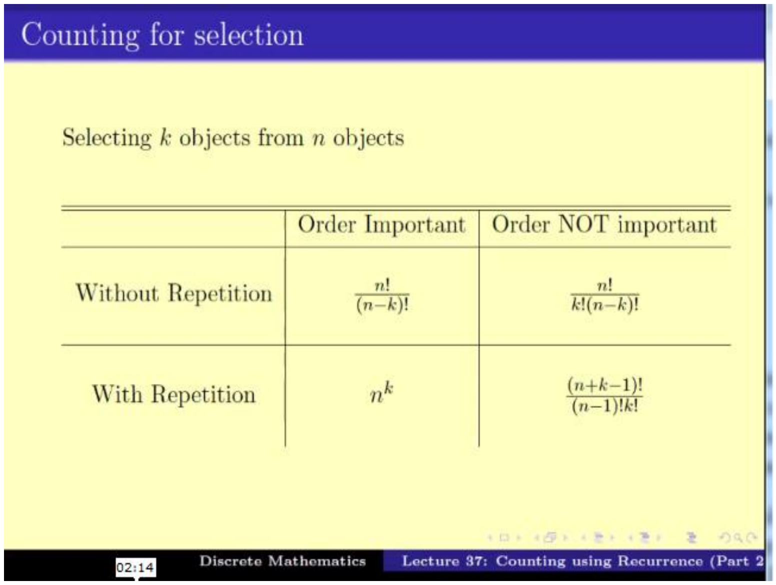
So here the some of the example that we already seen and how to tackle them.

(Refer Slide Time: 01:10)



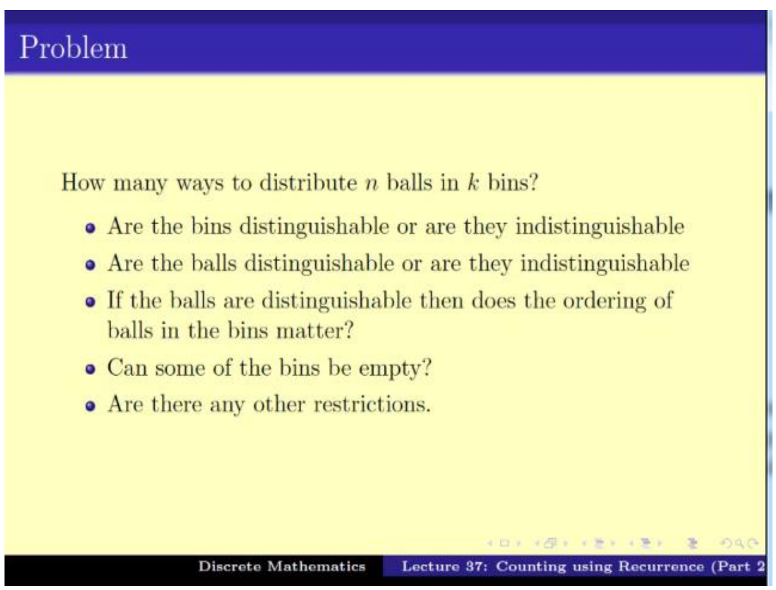
In fact, the big problem of counting is that the-- all the problems are unique and one need to apply a technique that fix it. Counting is in fact one of the most challenging subjects in mathematics and big names in mathematics as also works from the particular area. And we will be showing you some tricks and tools on such how to count.

(Refer Slide Time: 01:44)



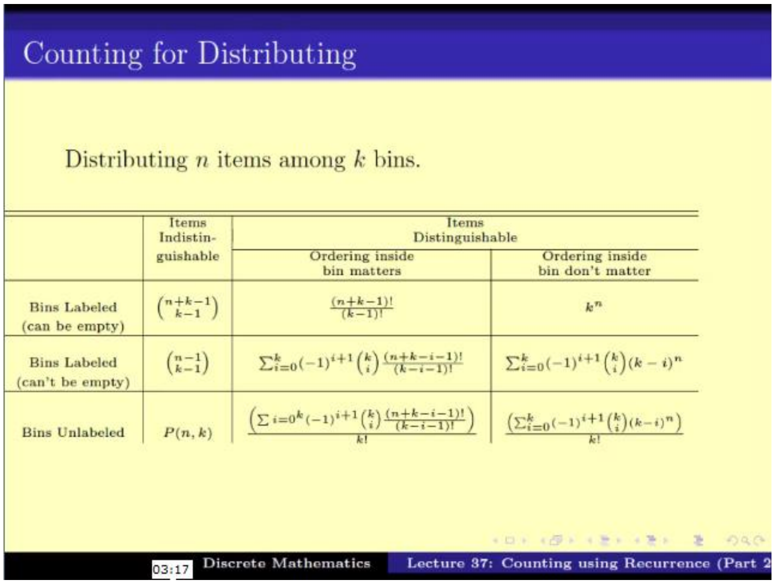
So one of the things that we looked at is how to select k objects from n objects and in that case we looked at whether the ordering which we select the elements matter and whether we can select a same object multiple times. And for all the four cases we came up with a solution which was a Compact nice beautiful solution something like the n! by k! times (n-k)! in this case.

(Refer Slide Time: 02:29)



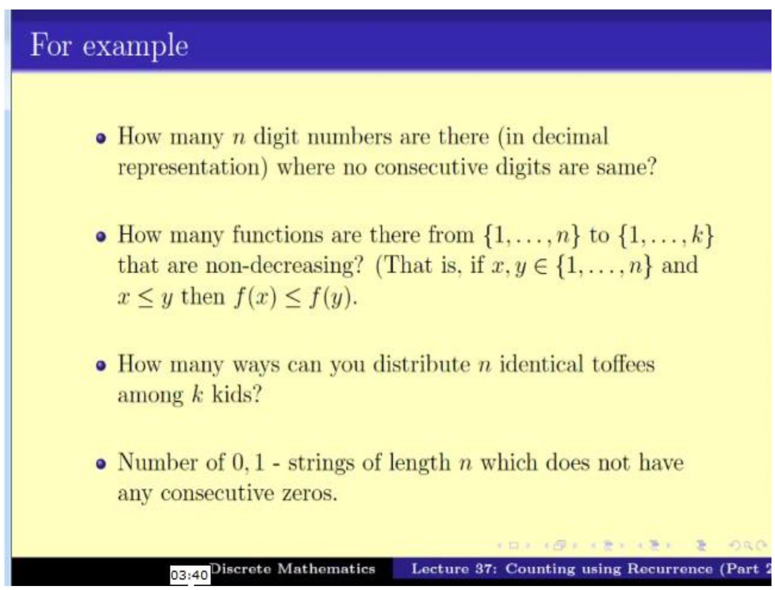
Now similar problem was how to distribute n balls into k bins. And here also there are multiple cases to be looked at namely whether the bins are distinguishable or the balls are distinguishable or whether the ordering in the bins matter or here some of the bins can be bin empty at all.

(Refer Slide Time: 02:55)



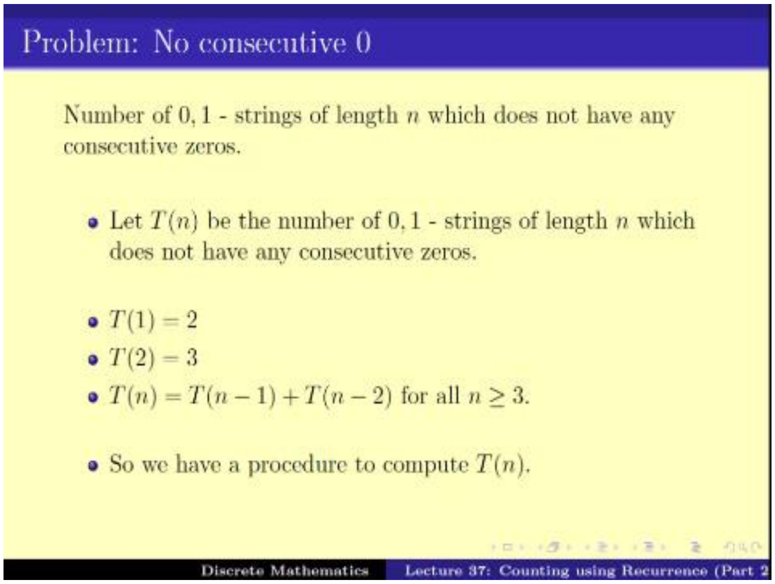
Now even in these cases also we came up with the certain set of answers to all the various cases. And they are mostly compact. Says that we have some idea about how to – what are these values are.

(Refer Slide Time: 03:15)



And we use all these tricks to solve this set of problems. The final problem was problem of how many 0,1 strings of length n are there that does not contain any consecutive zero.

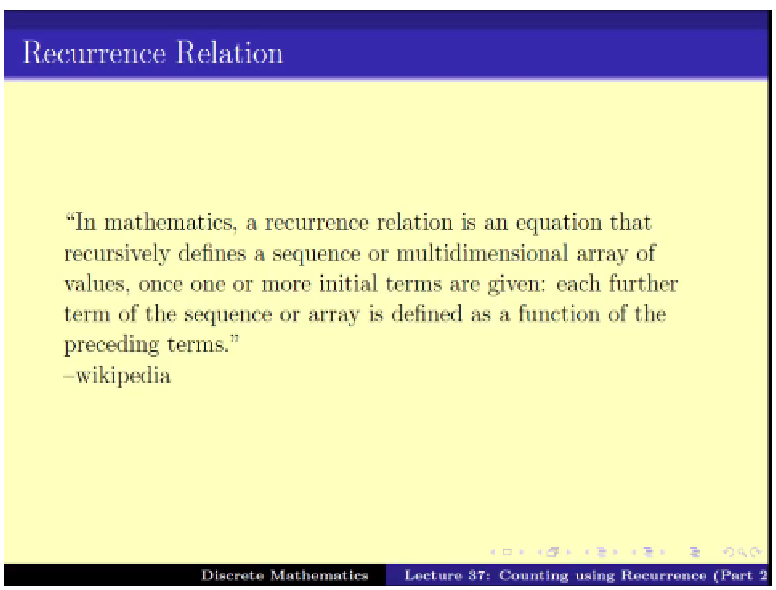
(Refer Slide Time: 03:35)



And if you take where which we did this problem is that we started with notation of T(n) be the number of 0,1 strings of length n which does not contain any consecutive zero. We proved that T(1) = 2; T(2) = 3 this is the way that you are counting and then you prove that T(n) is equals to T(n-1) + T(n-2) and this in fact gave us a procedure of computing T(n). Of course we left it in the case that can be find up nice expression for T(n).

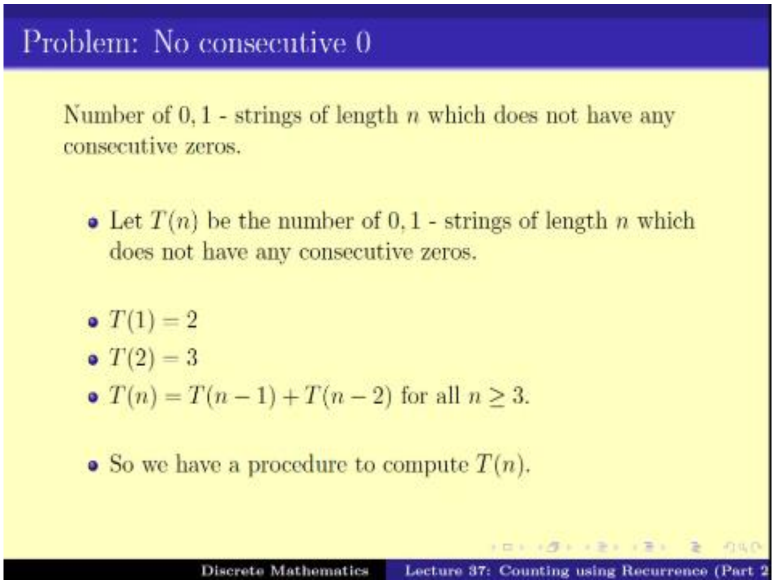
And we will come back to this particular question later on in the next videos. But the main idea was that we basically converted the problem of counting into a recurrence relation.

(Refer Slide Time: 04:34)



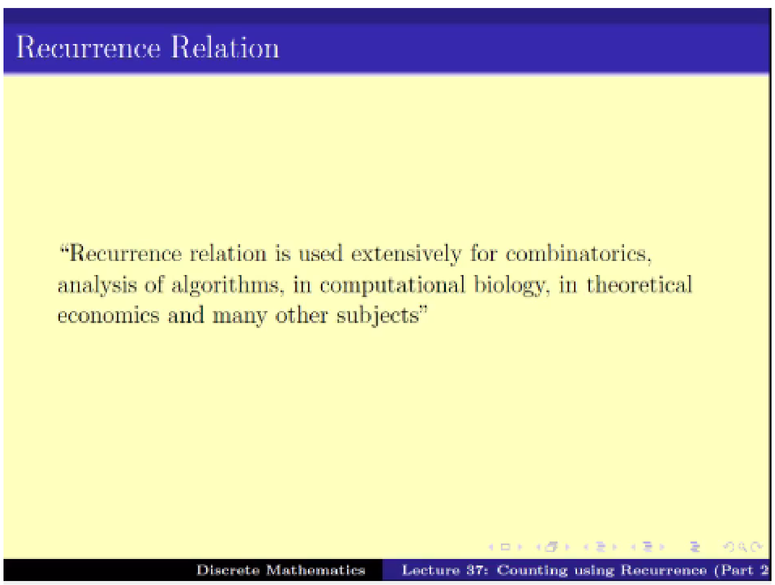
Or in other words we converted it into some sequence of numbers so that we have a set of initial conditions and we can any further turn is defined as a function of the preceding terms, right.

(Refer Slide Time: 04:51)



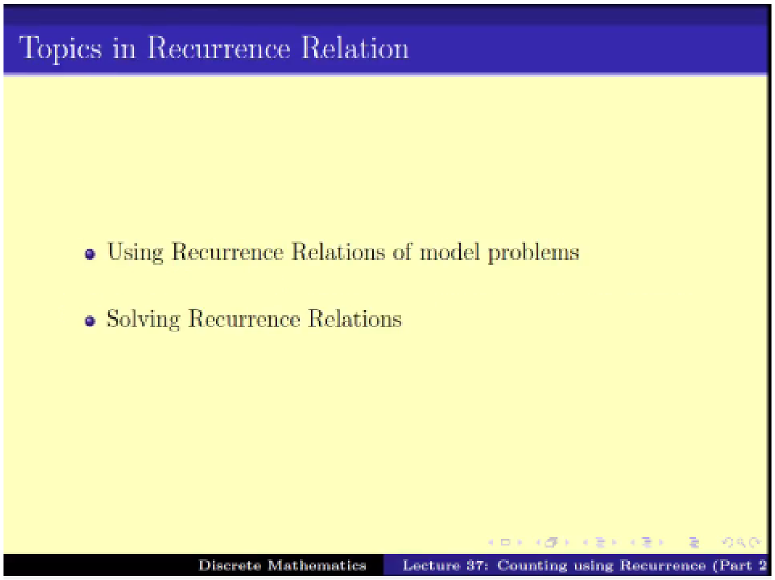
So here where the set of initial equations and here is the thing where it is recurrence relations where the T(n) is decide it is-- written as a function of T(n-1) and T(n-2).

(Refer Slide Time: 05:14)



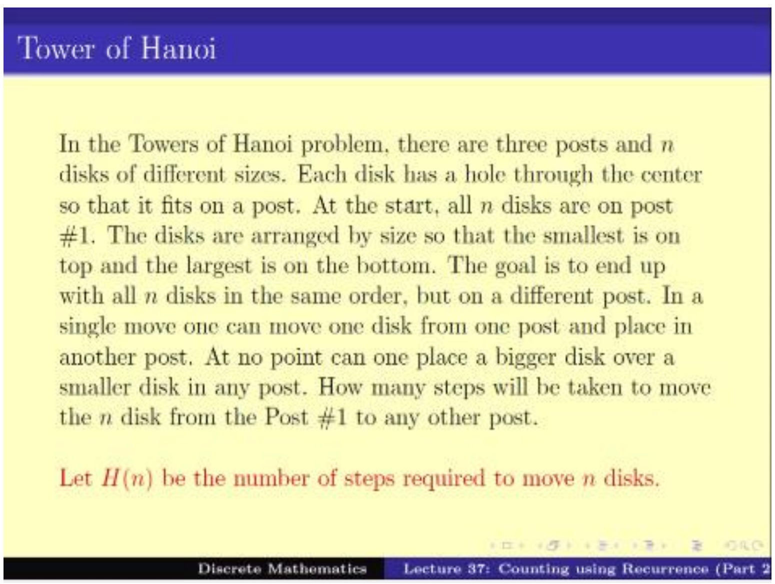
Now, Recurrence relations is extensively used in various subjects.

(Refer Slide Time: 05:23)



And in fact they have one of the very strong ways of modeling problems particularly counting problems and solving recurrences relations is also big part of this subject. Now in this video we will be looking at one more problem and see how we can model it using recurrence relations.

(Refer Slide Time: 05:47)



The topic of the problem is called the "Tower of Hanoi" problem. And the main idea is that you have three posts and n disks of different sizes.

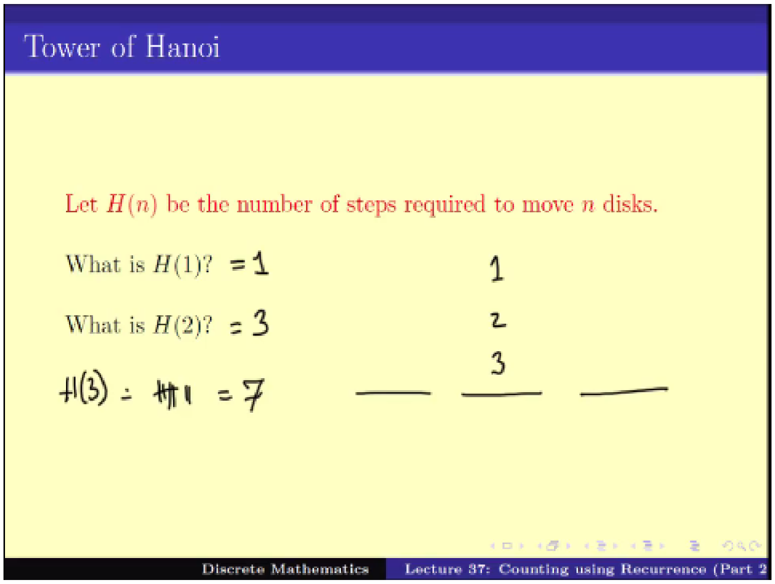
(Refer Slide Time: 06:06)

So let me show you the picture so something like this you have three posts you have this n disk here 1,2,3,4 and they are arranged in such a way that the smallest one is on top largest one is on the bottom and so the next-- this one is-- this one is base, so the one which is basically arranged in a decreasing form. And the goal is to somehow place all these things all these – all these n disks into the second wall into this wall this. And what we can do?

You can remove one at any point of time; you can remove one disk from here and put it anywhere. But you cannot put a bigger disk on a smaller disk, okay. So let me go on it. There are three posts and n disks of different sizes. Each disk has a hole through the center so that it fits on a post. At the start all n disks are on the post #1. The disks are arranged by size so the smallest is on the top and the largest is on the bottom. The goal is to end up with all the n disks in the same order on a different post. In a single move one can move one disk from one post and place it to the another post.

And at no point can one place a bigger disk over a smaller disk. Question is how steps will be taken to move n disk from post #2 to any other post. So here is the typical picture of Tower of Hanoi and it is picture like the game that it play but you cannot place a bigger disk on top of a smaller disk on any of the post. So question is that how many ways can you make the move. So for doing the counting the typical way of doing it is again let number of H(n) be the number of steps required to move n disks.

(Refer Slide Time: 08:51)



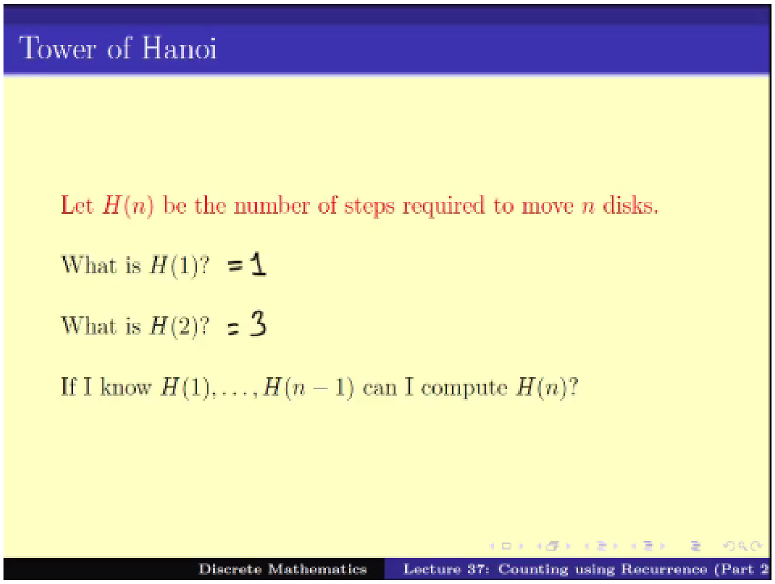
Now if H(n) be the number of steps required to move n disks, how many-- so can way say something above this H(n)? What about H(1)? So H(1), since I do not have I cannot got this so I just got as number okay so I have a three post and I have this number one. Now of course if I have to place this one in this next pole I just leave to pop this one out and lose it in the next part. So clearly H(1) equals to one right. And this is the (()) (09:37) thing.

What about H(2)? H(2) is interesting. Right, so I have these three poles there. In the first one I have 1 and then 2. How can I move this 1,2 to the second pole? So of course the first thing I have to do is I have to move this first and I move it to the first one. So I have one made one move then I move the second one to the second place so I make another move one. And in the third I move the first one back to the second one so which is one more move. So good.

So I make, so number of H(2) is actually equals to 3. Let us see one more example. What about H(3)? Here I have 1, 2 and 3. I need to move this 1, 2, 3 to the next pole. Now I know how to move 1,2 to the second pole maybe I can also move it to third pole right so I can move 1 to here, 2 to here then I move 1 from here to here, so I have till now made three moves right three then I make move this number 3 to this second one.

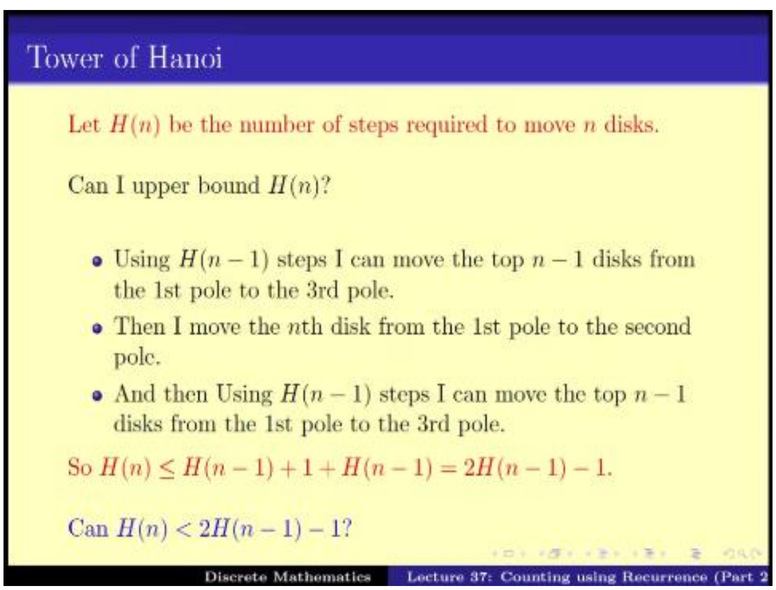
I have made four moves and now I move 1 to here then move 2 to here and I move the 1 to here. So I have made– initially I have made 1, 2, 3 then I made one more move then I made two more moves so namely I have made seven moves all total. So H (3) is 7.

(Refer Slide Time: 12:23)



Now like this if I keep on continuing and say if I know H (1) to H (n-1) can I computer H(n) and that is the question to be asked.

(Refer Slide Time: 12:33)



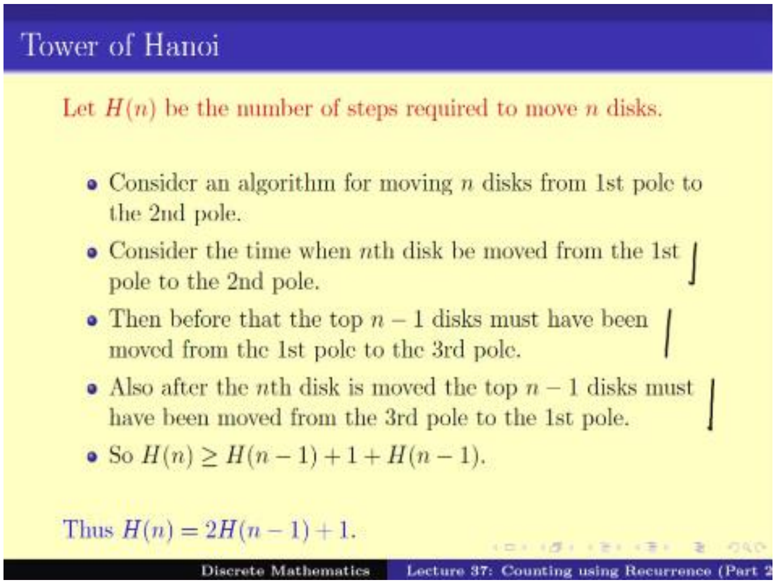
So let us see first here. First of all, can I get upper bound H(n)? If you remember what we did for the case of H (3) let us see what was done. We first used H(n-1) step to move the top n-1 disk from the first pole to the third pole right. That we can do because that is the definition of H(n-1).

Then I move the nth disk the top bottom was this—sorry for the first pole to the second pole. And then again I moved the n-1 one of them on top of the second pole.

So sorry this is not third pole this is second pole. This second pole right and this is also not the first pole this is the third pole. So I had it on the third from third pole I moved it to second pole. So in other words total number of steps I made was of course less than 2H(n-1) -1. So what we have is that the minimum number of steps required is less than or equal to 2H(n-1) – 1. Now unfortunately this is just upper bound.

Can we get a lower bound? Or in other word is it possible the number H(n) is strictly less than – sorry another mistake this is last one of course. Can it be strictly less than $H(n) < 2H(n-1) + 1$. And let us prove that it cannot be.

(Refer Slide Time: 14:47)

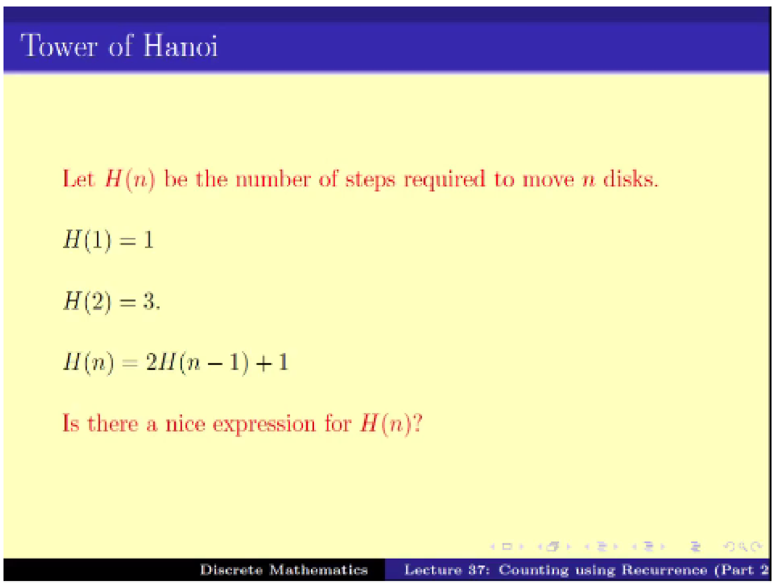


So consider an algorithm for moving n disks from the first pole to the second pole right. Consider the time when the nth disk was be moved, of course at some point of time the last is was moved from the first pole to the second pole. Before that we move it must have been the case that this top n-1 must have been moved from the first pole to the third pole. Right. Because when I move the biggest from the first pole to the second pole the first pole must have been empty other than that.

And till the biggest this cannot be placed on top of the any of the other cases. So the second pole must also have been empty, right. So the n-1 disk is must have been sitting in all of that in the third pole. Right. And of course to complete the game after I move the nth disk to the second pole the n-1 disk is must have been moved from third pole to the second pole—third pole to second pole.

So the number of steps that must have been made here is H(n-1) and H(n-1) and just 1. So H(n) must have made H(n-1) here H(n-1) here and the one move here so it is two times H(n-1) + 1. So in other word for the case that from the previous slide and this slide that H(n) is actually the equal to 2H(n-1).

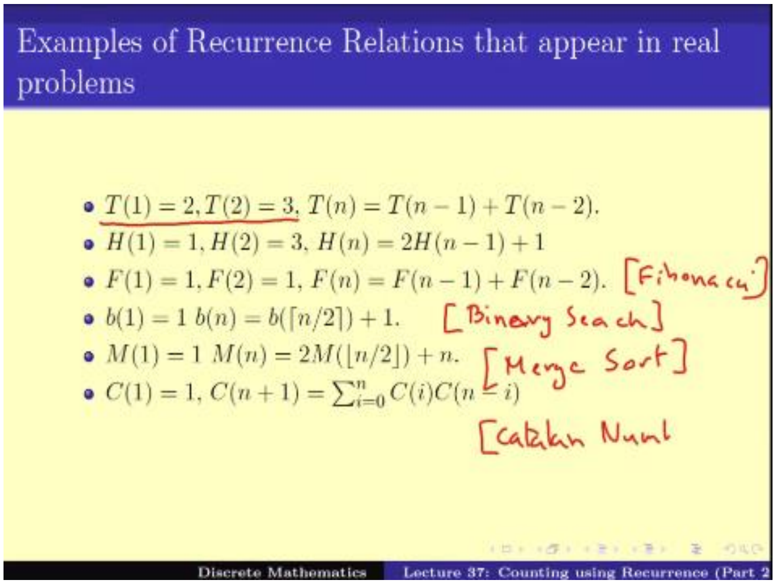
(Refer Slide Time: 16:44)



So thus for counting the number of steps required to move n disks I have H(n) = 1 and H(2) = 3 and H(n) = 2H(n-1) + 1. So as you can see here we can clearly write down here now that that okay H(3) = 2H(n-1) + 1 which is 7 number sorry—H(4) is 2H(3+1) is 15; H(5) is 31 and so on and so forth. So I can keep on doing this same again and again and get some values. Question is that is there a nice expression for H(n).

Or in other words instead of just like instead of just computing it H(n) in brute force manners like this is there a way of computing H(n). So we have seen two examples of how to solve this counting problem using recurrences relations.

(Refer Slide Time: 18:02)



Now there are many other recurrences relations that appears in real life, of course we have seen this the one this is the one that is saw for the case of 0,1 string without any consecutive zeros. We also saw this one. Now if I change the initial conditions, for example in the first case I have change the initial condition to something like this is one equals to one is two, this is the same as Fibonacci sequence.

Then we have seen slide for b(1) = 1 b(n) = b([n/2]) + 1 which is something that is arises from Binary search is called it is an algorithm. We have M(1) = 2M ([n/2]) + n this comes from what is called as Merge sort. So, various algorithms produced different types of such equations. And their equations are much more complicated like this where C (n) increases some of square of them and these are called Catalan numbers.

Now these are examples of recurrences relations. Recurrences relations as you can see is where the nth terms is written as the function of the previous terms as in this case. Now there are techniques of solving these recurrences relations that getting a compact for these recurrence relations. They are essentially not only for counting but also for algorithms for analysis of algorithms; for various other subjects of math. In the next video, we will be looking at techniques of solving this recurrences relations, thank you.