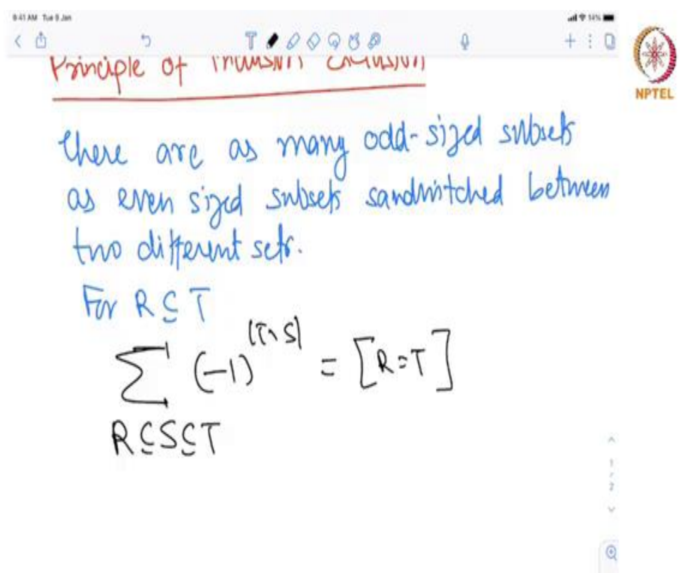
Parameterized Algorithms

Prof. Neeldhara Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Gandhinagar
Prof. Saket Saurabh
Department of Theoretical Computer Science
Institute of Mathematical Science, Chennai

Lecture – 39
Algebraic Techniques: Inclusion Exclusion (Colouring)

Welcome to the week 10 of the course, up until now we have been seeing a several new techniques in parameterize algorithms. And in this week we will learn another new technique in parameterize algorithms called algorithms based on algebraic techniques. And one of the first techniques that I would like to talk about is what is called inclusion exclusion based algorithm. So, now what is the principle of inclusion exclusion? Let us try to understand that.

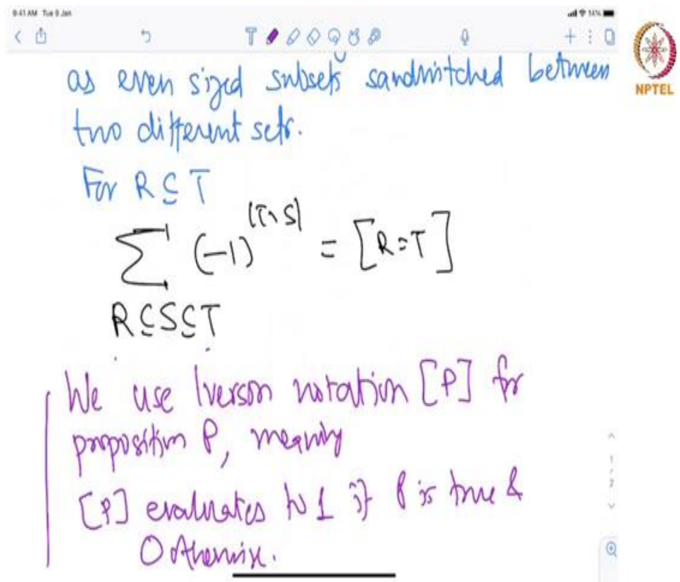
(Refer Slide Time: 00:42)



The principle of inclusion exclusion, there are several formulation we will see the more classical formulation later. But let me tell you something slightly different formulation and may be this will help you. So, basically what we would like to show is that there are as many odd sized subsets as even sized subsets that are sandwiched between 2 different sets. So, what I mean by this, so let us spell it out what does it mean?

So, this basically means so, we will prove this for R subset of T, look at R is there, S is there. And I do minus 1 then this is equal to. So, let me introduce this notion. This is also like this notion basically what I mean. So, what is this notation means R = T.

(Refer Slide Time: 02:51)

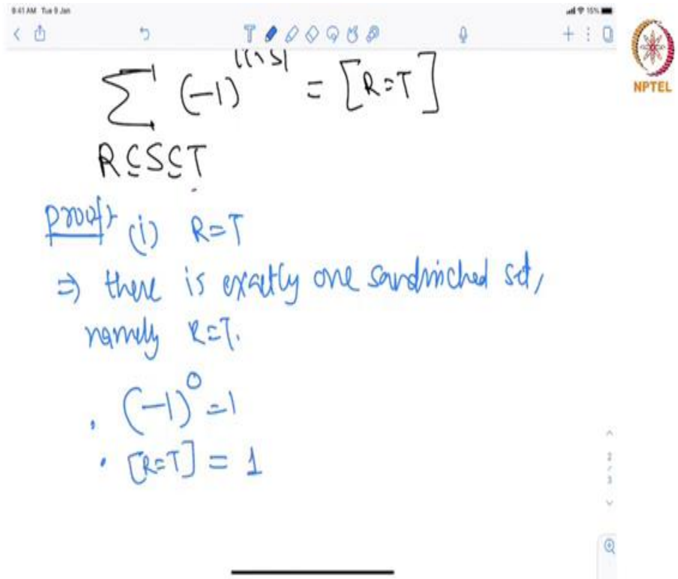


This is basically what is called we used what is called Iverson notation. Iverson notation P for proposition P, meaning this evaluates to 1 if P is true and 0 otherwise. So, in the context of this, what does this mean that this expression the left hand expression is equal to 1 if R and T are equal otherwise 0 here. Now, look at this expression what does this tells us? This expression tells us that you look at you fix an R, you fix a set T and you look at all the sets S which contains R.

But it is contained inside T and then you look at the minus 1 to the power there Cardinality. So, notice that all the odd sets here is going contribute minus 1 and all the sets S which contains R and contained inside T and they are that are even are going to contribute 0 because minus, they are going to contribute minus 1. So, the odd sets are going to contribute plus 1 even sets are going to contribute minus 1.

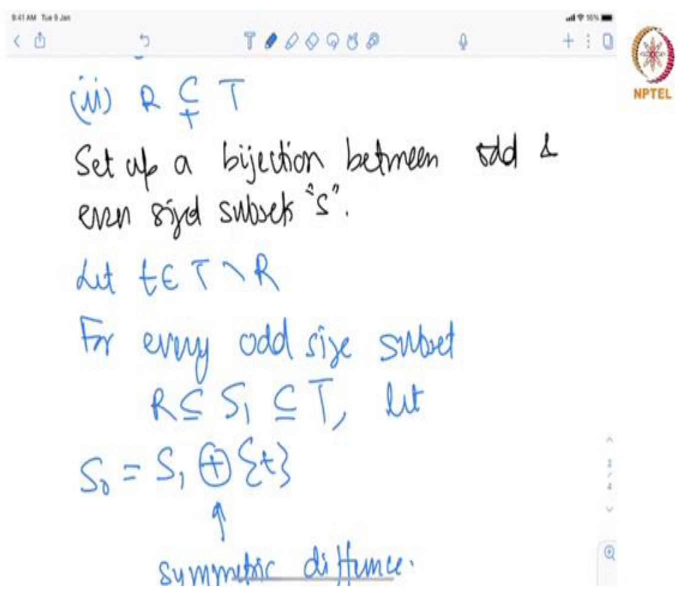
And if R and T are not equal, then the number of contributions that will come from all the odd size sets and even size set they will cancel out each other. And this is exactly what it means that there are as many odd size subsets and even sites subsets sandwiched between 2 different sets. So, now let us try to prove this. This is not very difficult. So, let us try to do this.

(Refer Slide Time: 05:14)



So, let us try to prove this. So, how are we going to prove? Let us take first R = T then in this case there is exactly 1 sandwiched set and what is that? Namely R = T. And then the left hand side and in this case namely R = T and so, the left hand side will become there is only 1 contribution that is minus 1 to the power 0 which is 1. And in the right hand side what is it? In the right hand side you know that R = T. So, this notation also evaluates to 1 so, this is fine.

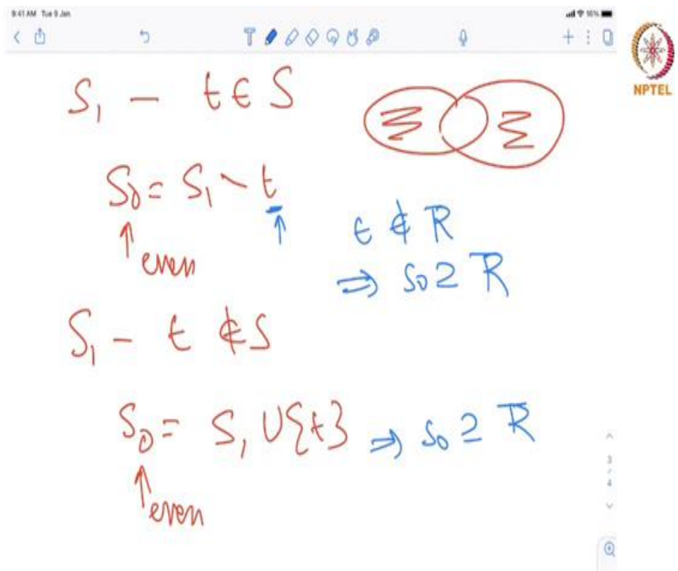
(Refer Slide Time: 06:17)



So, now let us assume that R is a proper subset of T. So, now what are we going to show? Now to show this we are going to how do we prove this in this case? We are going to set up a bijection between odd and even sized subsets. So, we are going to set up a bijection between odd and even size subsets S. And how do we set up such a bijection? That is very easy you know that R is a proper subset of T, so let t be an element of T – R. Because you know, now for so, what is the bijection we are going to give for every odd size subset S 1.

So that is great, it is slightly better. So, for every odd sized set S 1 that contains S and contained inside R, T contained inside R we are going to setup S 0 = S 1 and what is this notation? This is basically notation for symmetric difference.

(Refer Slide Time: 08:15)

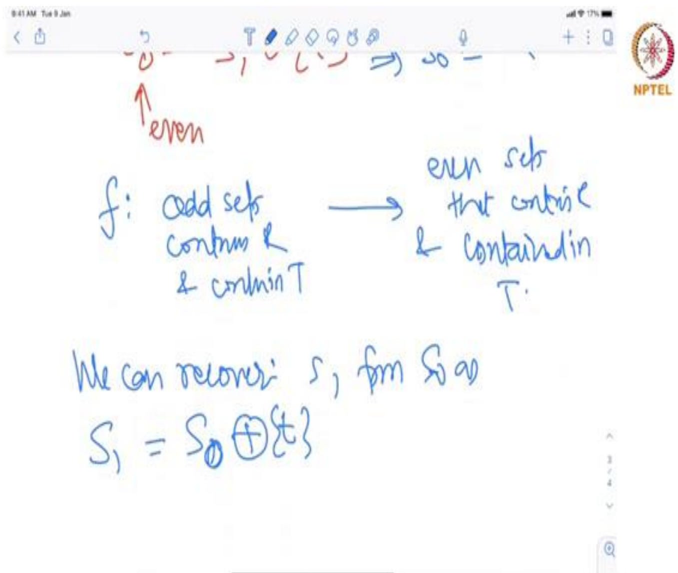


So, now notice. So, every odd size subsets; so, now look at S 1 what kind of set S 1 can be t could belong to S, if t belongs to S then what is S 0? S 0 is S 1. So, what is a symmetry difference? Symmetry difference means this and this. Now if t belongs to this then what is S 0 it is basically S 1 - t and then in this case what is S 0 is even. I will look at S 1. Now t does not belong to S then S 0 is what? It is S 1 and union t. So, since we started with S 1 as odd and we have added a new element to this this is still even.

So, notice that we have been so at least the mapping from odd to even is fine. So, every odd set is mapped to an even set and why this is a bijection because you can recover. And so, this is one thing, we also have to show that the set R is contained inside this. So, now we started look at the first case S 1 we deleted t but where the element t was? t was in t did not belong to R. So, what does it mean? So, if you looked at an odd set you deleted an element who did not belong to R. So, this implies that even in this case S 0 contains R.

And since this is anyway you are adding an extra element from a set, so this in this case also we can show that S 0 contains R. So, we have been able to show not R where T. We picked up T so, I made a mistake here this is what, so t does not belong to we, this implies that this is fine. This is so either way.

(Refer Slide Time: 11:30)

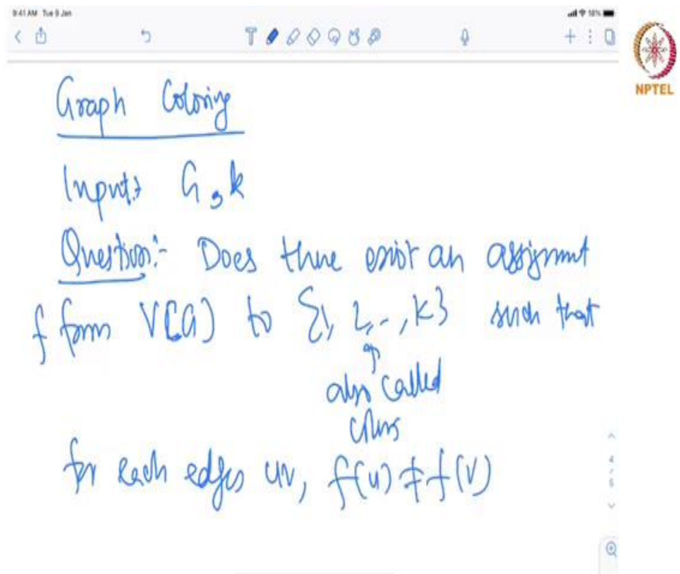


So, the basic fact of the matter is that we have been able to give a bijection from or we are given a function that maps even sets that maps odd sets containing R and contained in T to even sets that contains that R and contained in T. So, function is established. Now why do the bijective? That is bijective because we can recover S 1 from S 0 as again you can do the same thing. So, what is S 0? It is nothing but S 1. So, this is what so we can recover S 1. So, I object from you do.

So, you can be covered so look so it is like I can get S 1 by again doing the symmetric difference between 2 by with respect to the set T. So, this is why this forms this establishes bijection. And once you have been able to show that in the second case the number of sandwich sets of even size and all sides are equal. When R is a proper set then the left hand side contributions are the even number of plus 1’s and even an equal number of plus 1’s and equal number of minus 1’s.

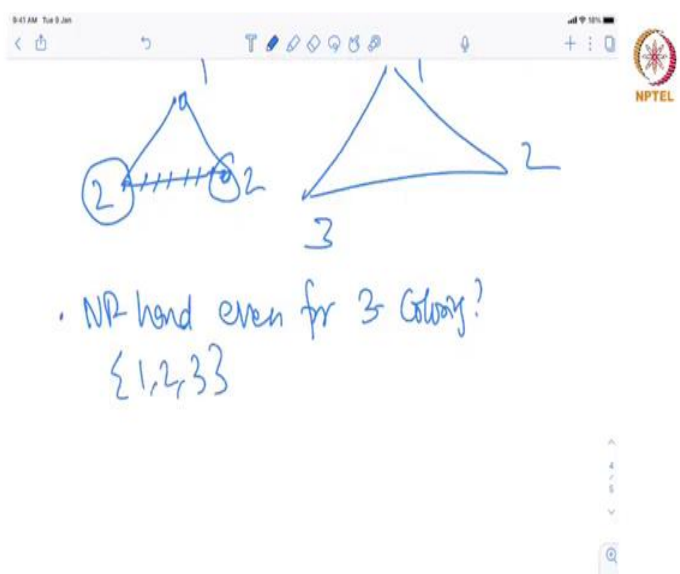
So, they will cancel out each other so left hand side is 0 and since R is not equal to T so this right hand side notation will evaluate to 0. So, you are done. So that is it. Now let us so why did I tell you this? So, it is a very important and interesting algorithm you will make out of this. So, first example I would like to give is in graph coloring.

(Refer Slide Time: 13:54)



So, what is graph colouring? So, graph colouring is nothing but it is so, input is a graph G integer K and question is, does there exist an assignment from vertex set of G to an integer 1 2 K also called colours. Such that for each edge UV assignment, fu is not equal to fv.

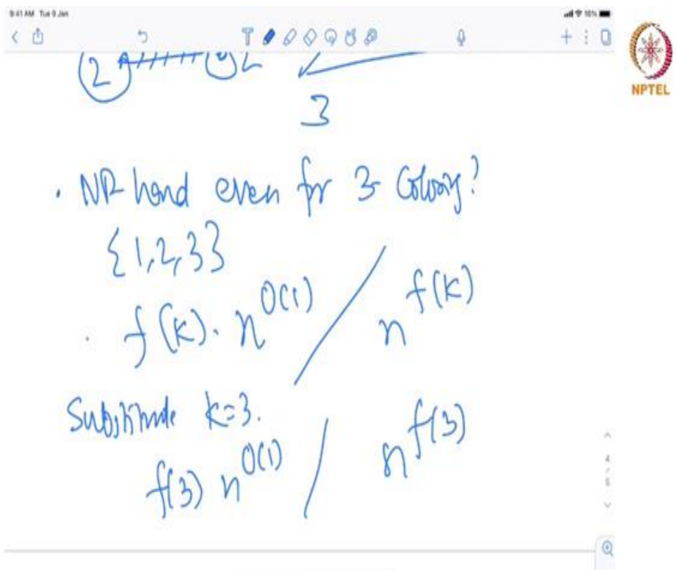
(Refer Slide Time: 14:56)



So, basically you have a graph and you want to assign numbers. So, for example, this is not a proper colouring because there is an edge here and both end points have been assigned the same colour. But this is the proper colouring because you look at the any edge the assignment maps different integers. So, this is what is called K colouring of the graph. And the question is there exist an assignment there exist the colouring of the vertices with colours 1 2 K such that no edge is monochromatic meaning that like same colour is not assigned to the end point of any edge.

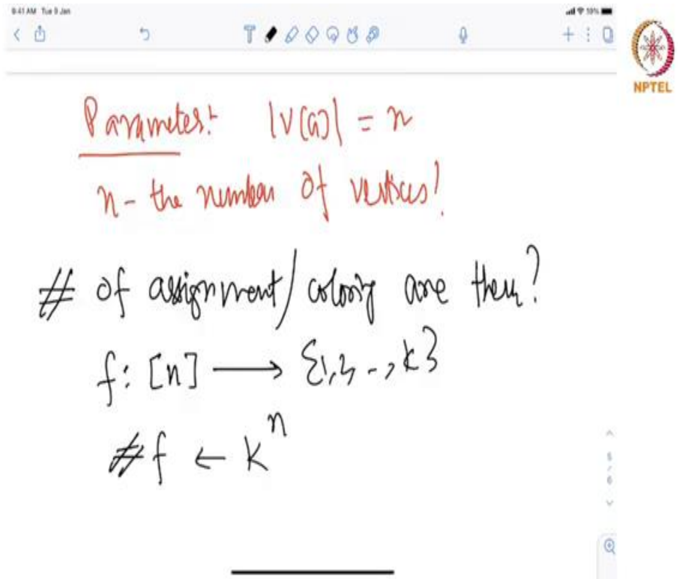
All of you know that this problem is NP hard even for 3 colouring. Meaning, if you are looking for an assignment from colour 1, 2, 3 even then the problem is known to be NP hard.

(Refer Slide Time: 16:05)



So, hence expecting an algorithm of running time say f of K n to the power O of 1 or even an algorithm with running times some n to the power f of K is not possible. Because if you could design such an algorithm then what it will imply? Substitute K = 3 then what you will get f of 3 n to the power O of 1 or even n to the power f of 3. Whatever it is for value 3 this is either polynomial both of them are polynomial time algorithm. So, unless p equal to NP we do not expect that p will be able to design such an algorithm.

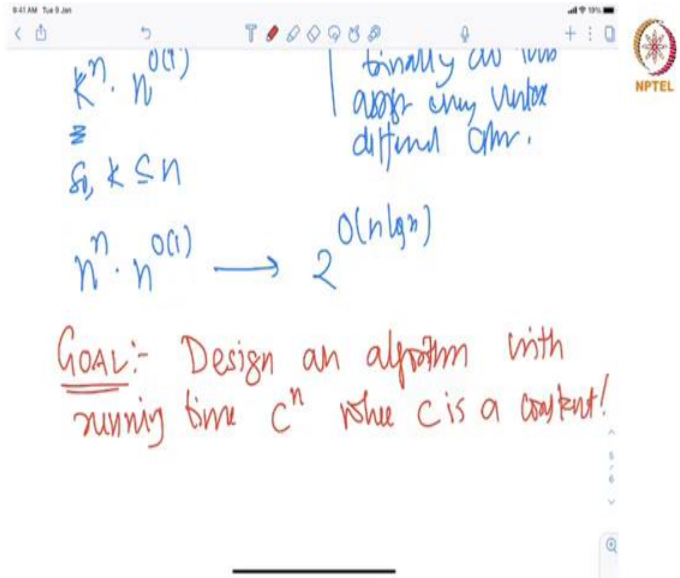
(Refer Slide Time: 16:45)



So, for this we are going to use a parameter which is cognitive of VG = n. So, we are going to use our parameter n the number of vertices. So, we can ask is this K coloring problem, is there K coloring problem? FPT parameterize by n, but now let us see how many number of assignments are coloring are there? How many assignments number of colorings are there? Number of assignments or coloring are how much. So, you are looking for a total number of functions from 1 to n to 1 to K.

So, every vertex could be assigned one of the K. So, the number of such functions number of f's are upper bounded by K power n. So, you can enumerate each of this function and check whether they form valid coloring or not.

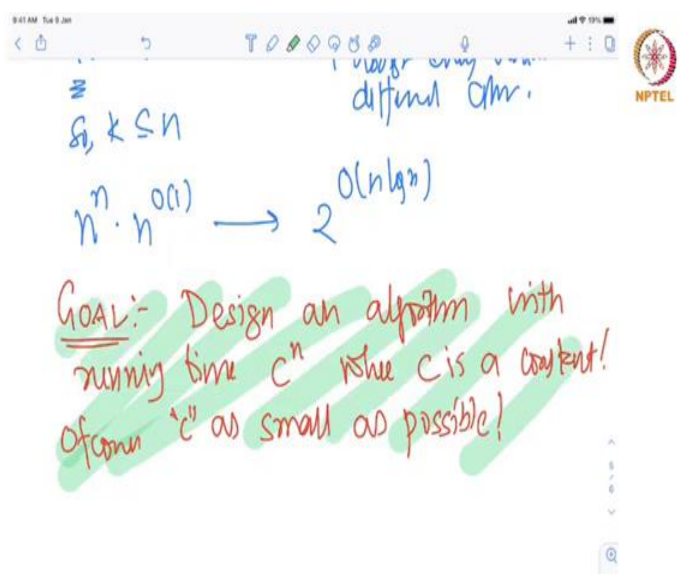
(Refer Slide Time: 18:06)



So, the 1 trivial running time of an algorithm could be K to the power n and n to the power O of 1, but now you know that like so, this is 1. So, if you are looking for a proper colouring and generally this is like your look. So, for example, if K is greater than equal to n then you can trivially do this assign every vertex different color and you are done. So, we can assume that K is less than equal to n. And hence n to the power n, n to the power O of 1 which is 2 to the power O of n log n and we are happy.

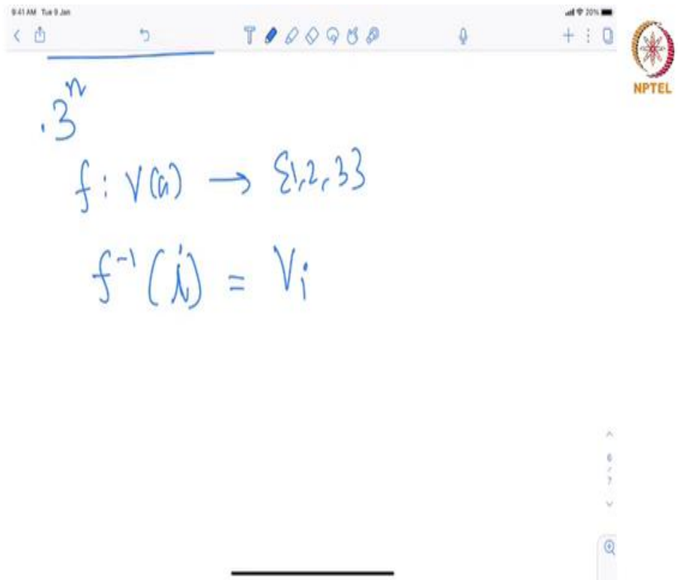
So, now as we have been trying to do. So, what is the goal? So, goal is fixed. So, the goal is to design an algorithm with running time C power n where C is a constant.

(Refer Slide Time: 19:38)



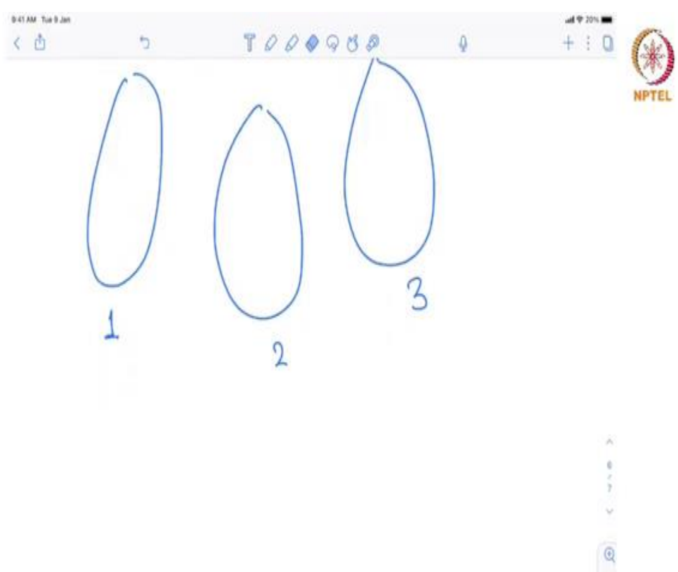
And of course C as small as possible; so that is the goal or in this area of computer science. So, just to give you a motivation let us start with 3 colouring.

(Refer Slide Time: 20:08)



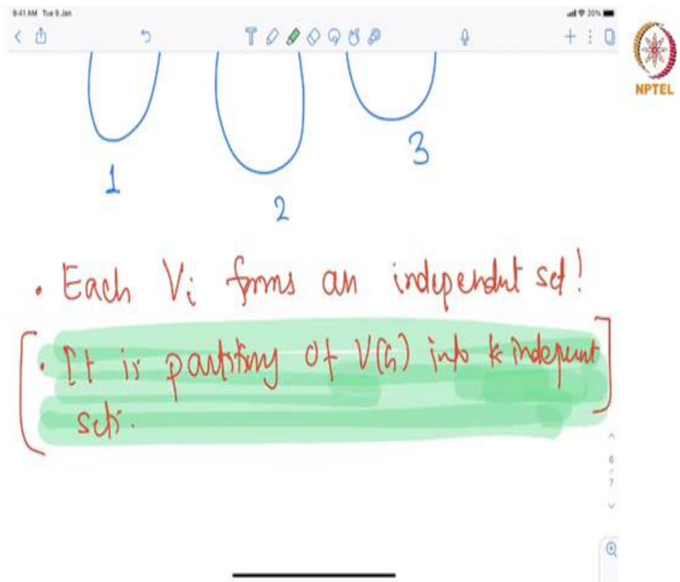
So, I want to assign from 1 to 3 and I want to do better than 3 to the power n. Because the trivial algorithm is 3 power n and the question is can we do better than 3 power n. So, now let us look at a valid colouring. So, valid f in an assignment 1, 2, 3 now notice f inverse of 1, let us call this V 1, so let us call this we will use V i.

(Refer Slide Time: 20:54)



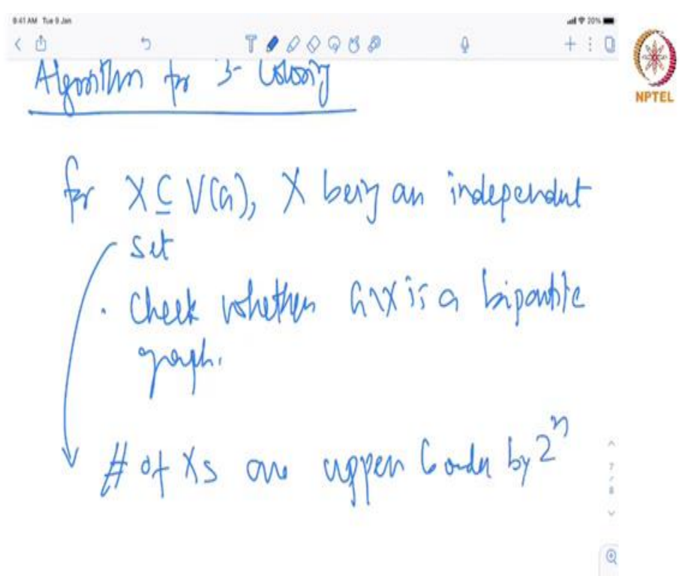
So, now look at this graph. Now all these vertices have been assigned 1, all these vertices has been assigned 2, these has been assigned 3. Notice that there are no edges here because otherwise this will be monochromatic. So, basically what is what is the property?

(Refer Slide Time: 21:23)



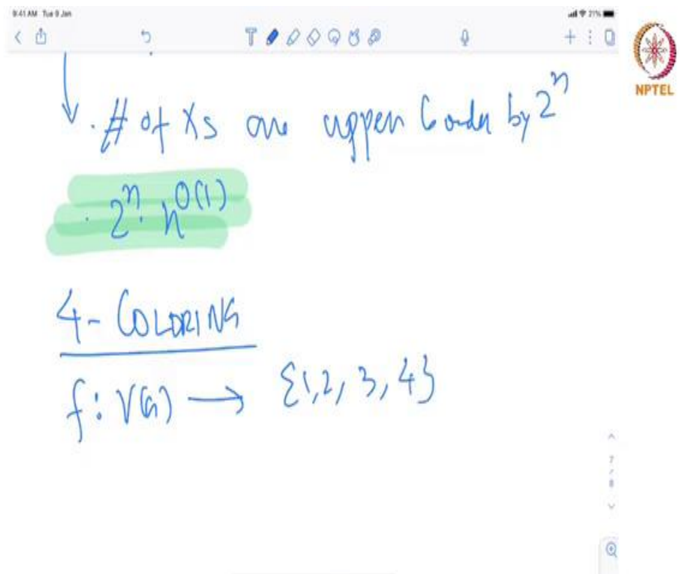
The properties that each V i forms and independent set. So, that is one of the observations. And secondly, in fact it is partitioning of vertex G into K independent sets. So, we will come back to this principle in a minute. So, once you have seen this there is a very simple algorithm for testing with a graph with 3 colourable how. So, the algorithm for 3 colouring is very simple.

(Refer Slide Time: 22:28)



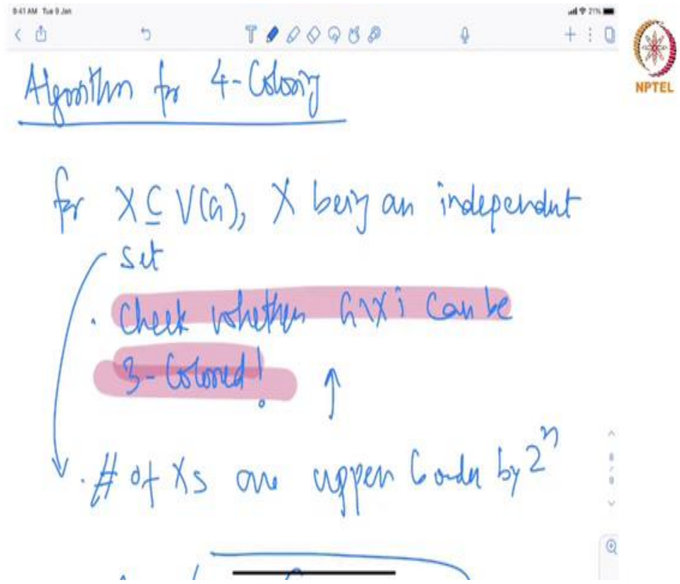
Here is an algorithm for 3 coloring. So, what is an algorithm for 3 colouring? First enumerate an independent set. Rather like let us look for all X subset of VG, X being an independent set check whether G - X is a bipartite. Now, notice that if I have guessed the colour class 1 correctly then the graph induce 1, 2 and 3 is nothing but 2 colourable or nothing but graph induce some color class V 2 union V 3 should form a bipartite graph. So, this is why I checked. So, notice so this algorithm will run in time, the possibility of X number of X's are upper bounded by 2 power n.

(Refer Slide Time: 23:47)



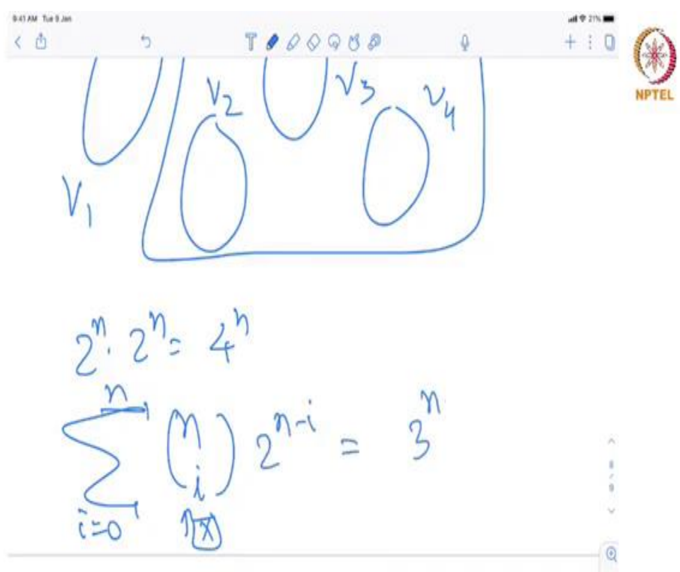
So 2 to the power n and n to the power 1, so this is 1 algorithm and but now you can ask yourself, what about 4 colouring? So, now you are looking for an assignment from VG to 1, 2, 3, 4. So, now what I can do? So, now let us make an algorithm for 4 colouring.

(Refer Slide Time: 25:10)



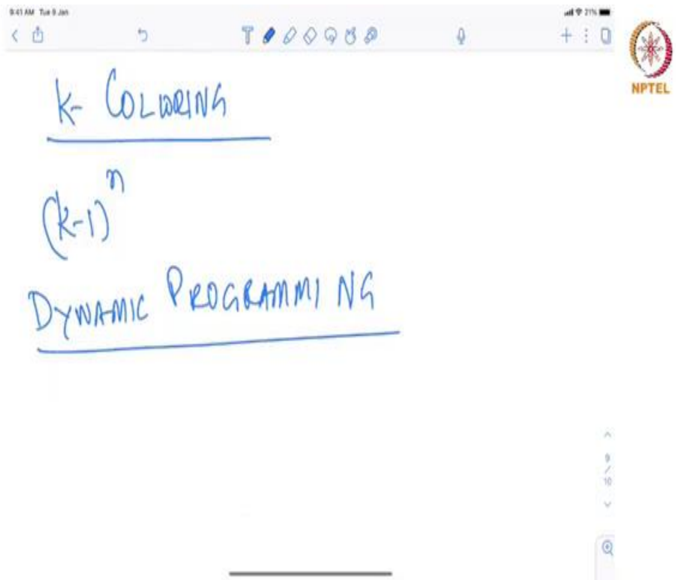
The X being an independent set, check whether G - X i can for 3 colours. So, now we made changes here. So, rather than checking whether, we said it is fine. Now what is an algorithm for 4 colouring? Algorithm 4 colouring is nothing but you. So, once you have guessed V 1 the remaining V 2, V 3 and V 4 the graph is 3 colourable. So, in this case now, in the first case we have 2 power n and we could do this checking in poly time, But now we cannot do this so, but for this we have an algorithm with running time 2 power n.

(Refer Slide Time: 26:06)



So, if you do naively it will become 2 power n times 2 power n 4 power n. But that is not how you should be doing. So, basically you have to do in choose i, i going from 0 to n. So, this is possibilities of x and then in the remaining how many vertices are left? In the remaining the total number of vertices are left is n - i and n that you run your 2 to the power n - i algorithm. So, this is going to be 3 to the power n.

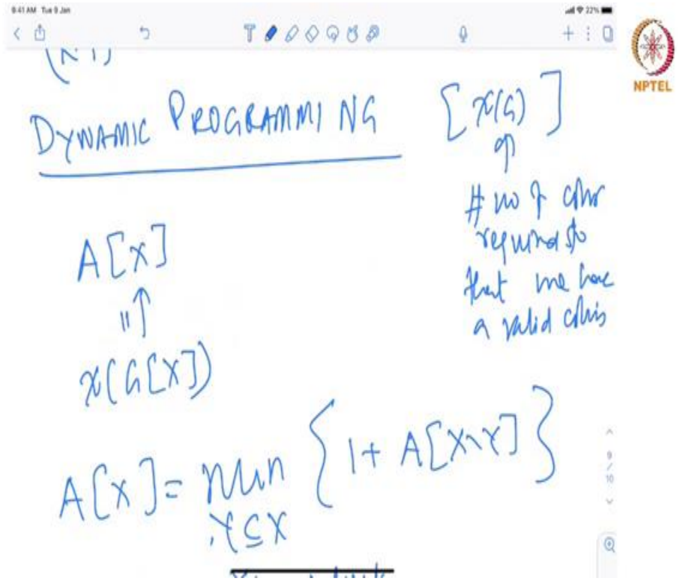
(Refer Slide Time: 26:35)



So, just generalizing this you can get a K colouring algorithm in running time but, this tells us something. I mean this tells us this idea of an algorithm tells us something that X being an independent set with a check whether G – X is 3 colourable. So, once I know the first set in the remaining I need to know the chromatic number of G – X. So, that just implies that this is like this immediately implies that maybe we should look for a dynamic programming algorithm. And in fact you can design a dynamic program a simple dynamic programming algorithm based on the following recommends?

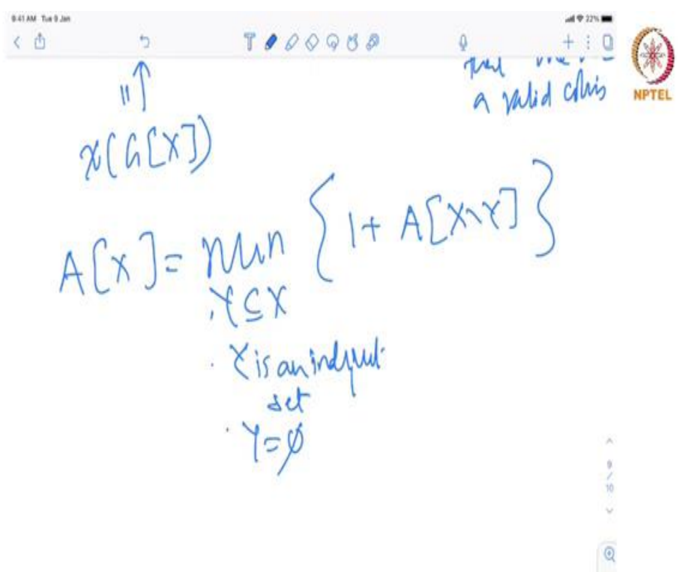
What is the following recommend? So, if you are looking for a following K colouring or chromatic number you can do anything.

(Refer Slide Time: 27:41)



So, look at AX. So, what is A of X? It is going to write down the chromatic number of graph induce. So, now let us try to compute chromatic number of G. And what is the chromatic number of G? Minimum number of colours required so, that we have a valid colouring. So, this is what chromatic number is like minimum number of colors you need so that every edge can be non mono chromatic. So, this is what A of X is going to store? Chromatic number. So, what is A of X? Graph induces. So, I am going to look at A of X look at it.

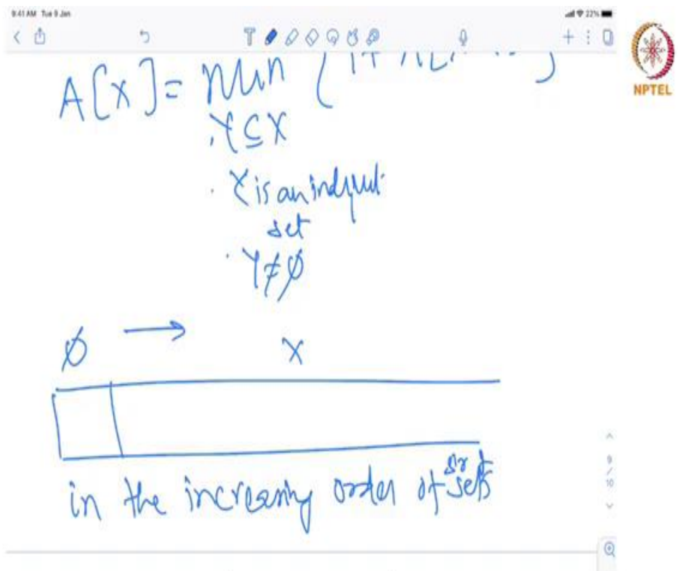
(Refer Slide Time: 28:32)



So, I guess the first independent set, so this is going to be minimum over Y subset of X, Y is an independent set, that is the first thing. Y is a subset of X, Y is an independent set and what is Y? Y is an independent set and Y is not equal to phi. So, you have guessed the first thing. So, it is 1 plus the chromatic number, the minimum number of colors you need for the graph induced on G - Y. So, this is AX - Y and that is it. So, I am trying to have an array A of X. So, basically A of X is indexed by a subset of my vertex set and what does A of X stores.

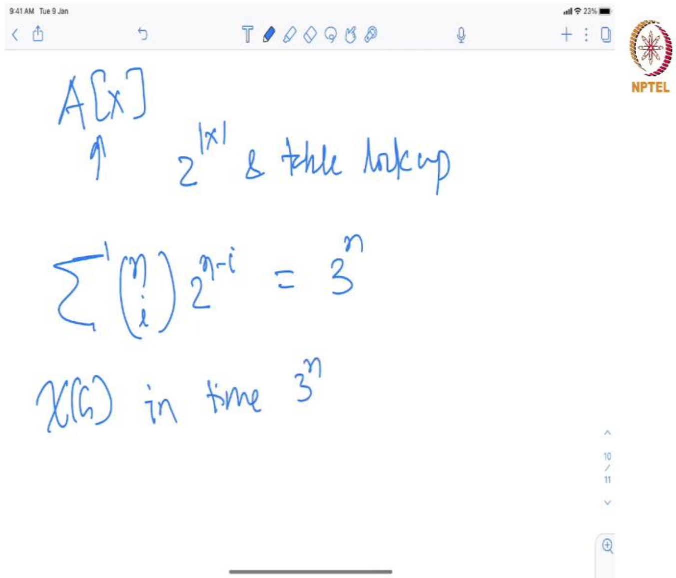
A of X stores the chromatic number of that and that can be given us minimize Y subset of X. Y is an independent set Y is not equal to phi and 1 plus this. So, now what?

(Refer Slide Time: 29:51)



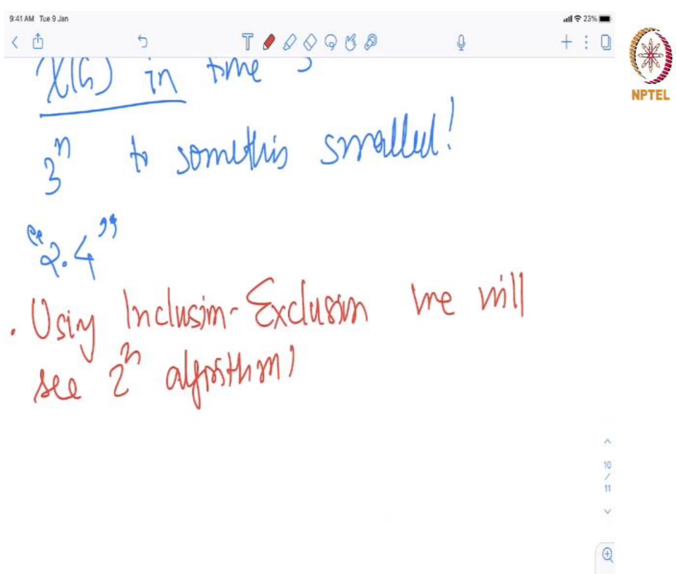
So, this is a big array, and you index them with subsets okay and you fill this up in the increasing order of sets and if you fill this up increasing order of size of set. So, you first fill up the 0 size set then 1 side sets then 2 sides sets then 3 sides so on and so forth and what is this? This is 0 A of phi is it.

(Refer Slide Time: 30:29)



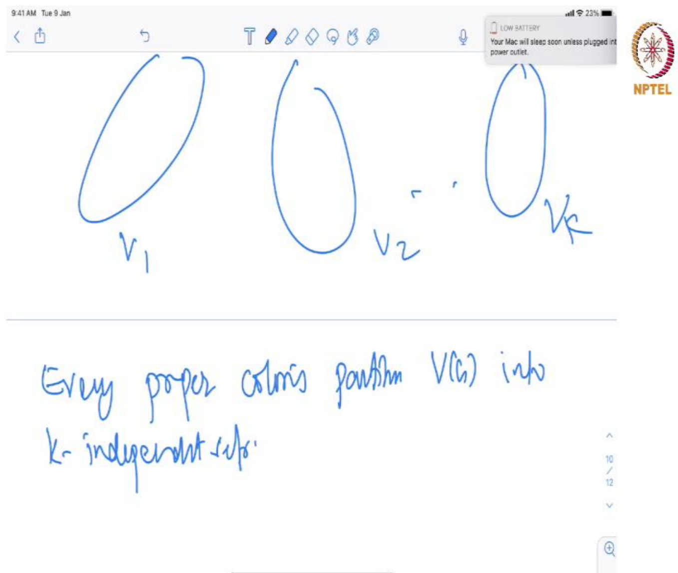
And now to evaluate A of X, how much time will it will take to evaluate A of X? Time to evaluate A of X is basically 2 power cardinality of X and table lookup. So, you take you like so you take 2 power X entries and you take minimum of that and you put that in. So, the running time of this algorithm is again going to be n to i and 2 to the power n – i that is what and this is going to be 3 power n. So, you can compute chromatic number of a graph in time 3 to the power n.

(Refer Slide Time: 31:13)



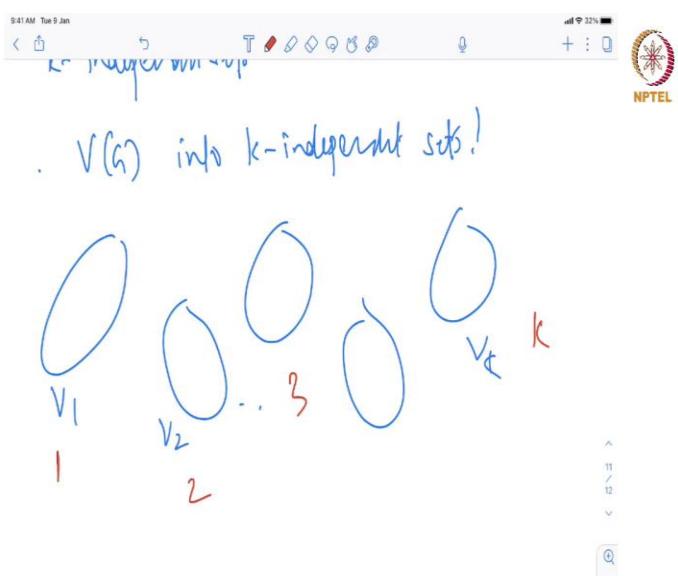
So, there was lots of work happening in this area to get 3 power n to something is smaller. And with lot of effort it was stuck at roughly I would say 2.3, let say 2.4 it is not accurate, but this is 2.4 roughly. It was stuck and then there was a very beautiful algorithm which we are going to see now. Now using inclusion exclusion we will see 2 power n algorithm. So that will be our goal using inclusion exclusion to see 2 power n algorithm. So, now let us go back to our colouring perspective. So, what is a coloring?

(Refer Slide Time: 32:30)



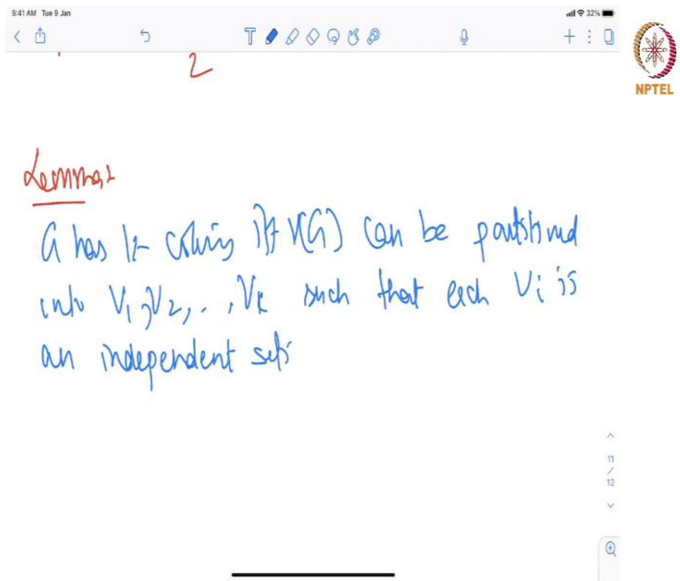
So, as I said, so like let us look at any color assignment colouring. So, basically as I said it is a basically every valid coloring or every set proper colouring partitions V of G into K independent sets that is very clear that every proper coloring partition VG into K independent set but what about the other way around?

(Refer Slide Time: 33:27)



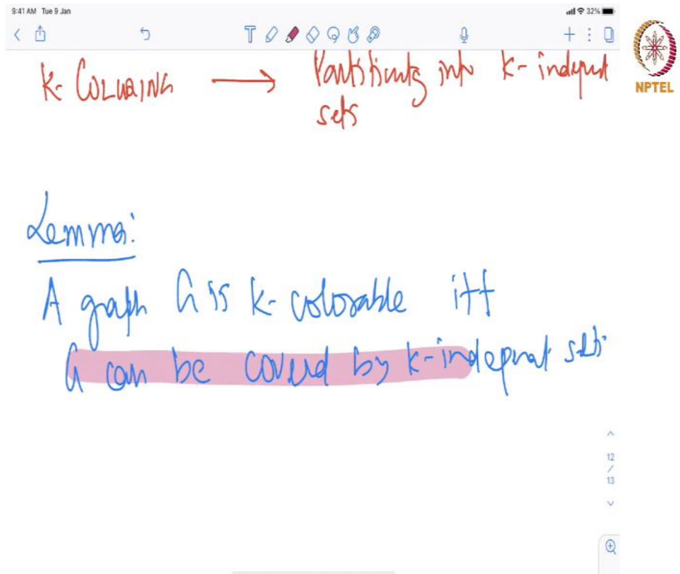
So, if I have a partition of say V of G into K independent sets what does that imply. So, I have this is partition of VG into K independent sets V K, the moment I have a partition of VG into K independent sets I could assign color 1 2 3 K and if you notice, because these are independent said there are no edges here which implies that this is a proper K coloring. So, now what we are to be able to prove?

(Refer Slide Time: 34:03)



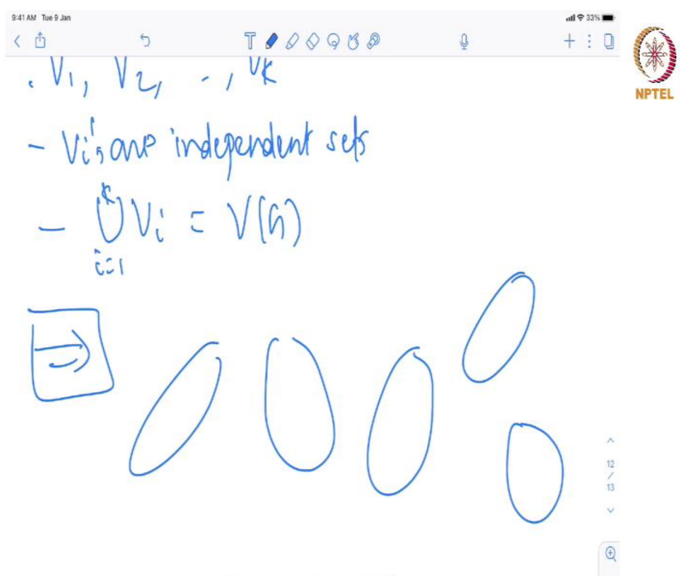
We are able to prove that the following lemma that G has K coloring if and only if V of G can be partitioned into V 1, V 2, V K can be such that each V i is an independent set and that is perfectly fine. In fact so, this is so we have reduced what we have done is we have reduced K coloring to partitioning in partitioning into K independent sets and we will utilize this fact to do inclusion exclusion. But before that let us make it even better in fact what we can say is the following.

(Refer Slide Time: 35:25)



We can prove a better slightly more general lemma and we say a graph G is K colorable if and only if G can be covered by K independent sets what is the meaning of this. Let us try to focus there can be covered with K.

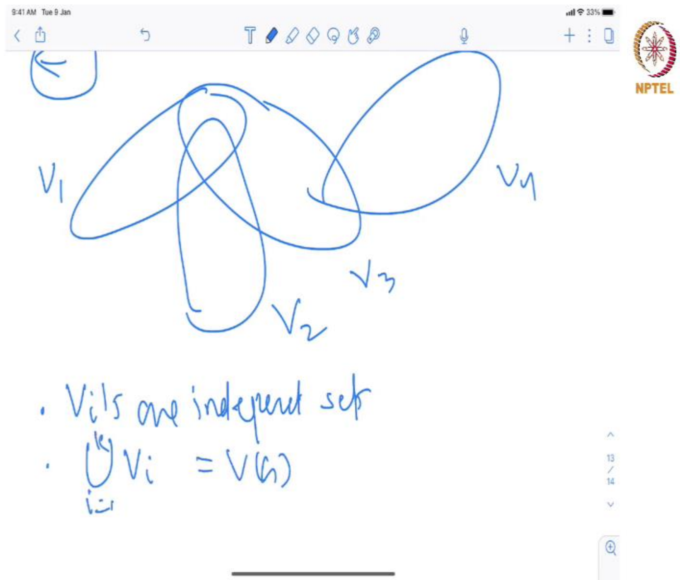
(Refer Slide Time: 36:01)



So, basically what I am saying a graph G is K colorable if and only if I can find V 1, V 2, V K the V i's are independent sets and union of V i, i going from 1 to K = VG. I am not claiming that they should form partition. What is the meaning of partition? A vertex will not appear in 2 different sets. So, I am saying that look of course K coloring is same as partitioning into K independent set but in fact what I am saying that it is K colorable even if you could cover the all vertices with K independent sets. So, these are basically 2 differences.

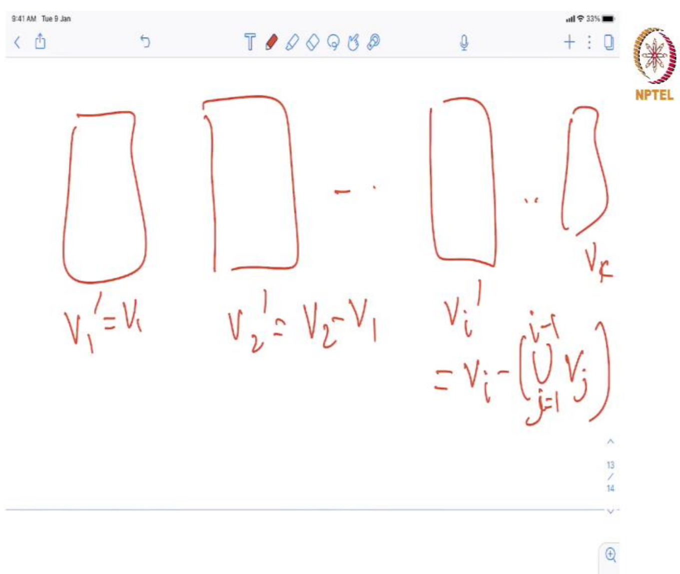
So, now let us prove forward direction is very easy, if graph G is K colorable then you know that they can be partitioned into K independent set. And hence they can also be covered into K independent set in fact; even better no vertex appears twice.

(Refer Slide Time: 37:13)



So, forward direction is very simple, but what about the backward direction now, we know that we have the sets maybe like this V 1 say V 2, V 3, V 4. What is the property? V i's are independent sets and union of V i's, i going from 1 to K = V G.

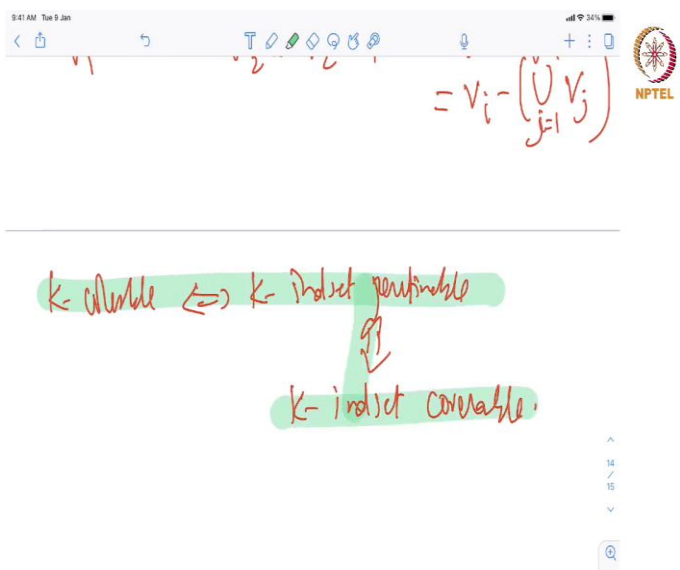
(Refer Slide Time: 37:50)



Now given this covering we will from covering we will recover a partitioning. How you fix V 1? And now what is V 2? So, what is the new so V 1 is V 1 we will make V 2 prime which is nothing but V 2 – V 1 fine. So, similarly so basically what when you are trying to construct V i prime it is nothing but V i minus union of let us call this V 1 prime = j going from 1 2 i - 1 V i V j. So, whatever previously you have selected just remove all those vertices. So, V j only contains those set of vertices which does not appears in V 1, V 2, V i – 1. That is it.

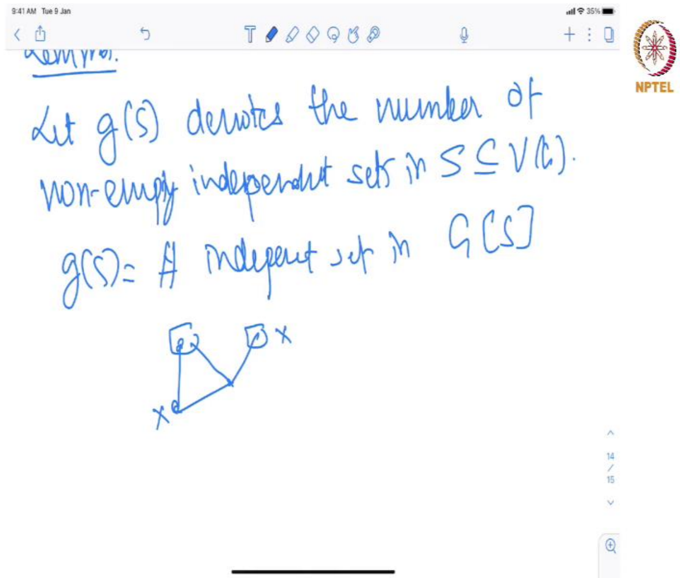
So, now notice that because of this we have and this is how you can get V K now notice that every vertex is in some set and in fact every vertex is exactly1 in 1 set. Because we have make sure that no vertex appears in more than 1 set and all of these are basically a subset of an independent set so, they are also independent set. So, from a given covering we have been able to obtain a partitioning word of independence set. So, what does this imply? So now, we have actually reduced our problem from if you want to test for the graph is K colorable we are asking can be covered the whole graph with K independent sets. So that is it.

(Refer Slide Time: 39:51)



So, we will use so notice what does this implies K colorable if and only if K independent set partitionable if and only if K is independent set coverable that is it. So, let us we will use this for our purpose so, the lemma which we will like to prove is the following.

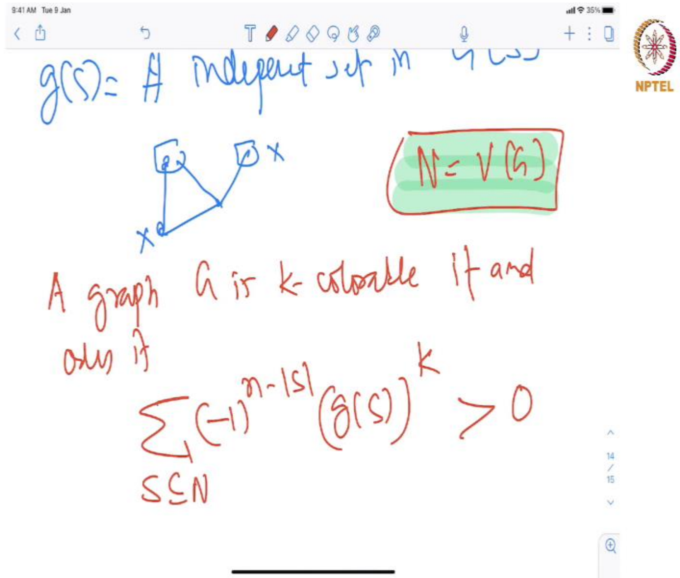
(Refer Slide Time: 40:22)



And we will see how we can use that to design our algorithm. So, to do this to be will before this let g of s denotes the number of non-empty independent sets in non-independent sets in s subset of VG. So, basically the rather you should say that g of s is number of independent sets in graph induced on s but non empty one. So, this is what g of s means. So, for example so let us call this. So, this is one independent set of this graph.

All Singleton's are one independent set and if you look at the 2 independent set it can only be this and this or it can be this and this that is it. So, but we are only talking about non independent non empty independent set we are not talking about empty independent set. So, now what we can say about this

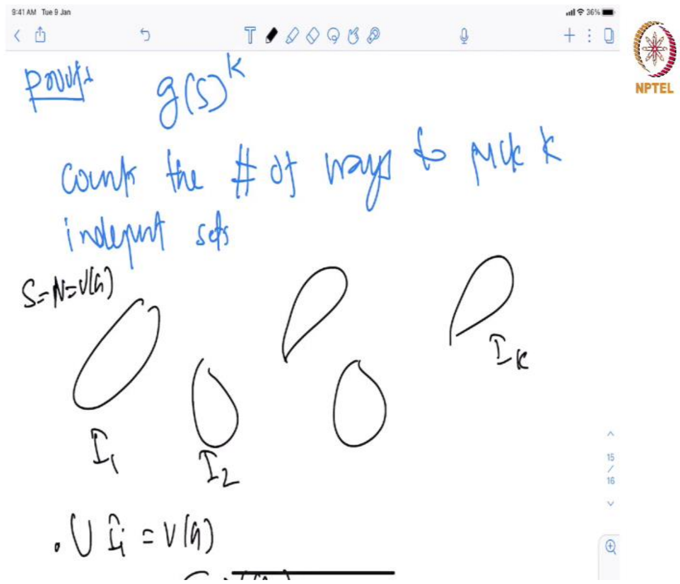
(Refer Slide Time: 42:37)



So, the lemma is the following. Now that we have set up the notation, a graph G is K colorable if and only if summation s subset of N rather than say VG. Let N = VG just so that it is easier to use notation, s subset of N – 1, N minus cardinality of s. So, remember what is g of s? g of s is number of independent sets non empty independent sets in my graph and why we are talking about non empty independent set is because you want V 1, V 2, V 3 each of them to be non-independent.

So, now to prove this, will prove this so, what is this g of s K counts? So, basically notice what g of s K counts is basically it counts you number of ways to pick K independent sets, so I am not saying that like look number of ways. So, let us not well do a proof for this.

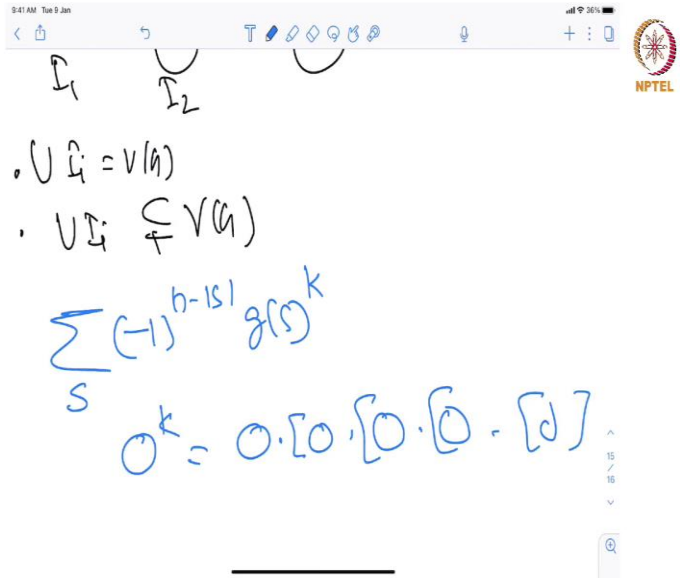
(Refer Slide Time: 44:11)



So, to prove this first let us what does it tells us that look what is g of s to the power K is number of ways to pick K independent sets. So, our intuition is that so, in particular what is a good way of picking a independent set for our purposes if for us good will mean so, let us try to understand what is g of s to the power K means it comes the number of ways to pick K independent sets. But let us say if I selected some K independent sets. And suppose S = N which is equal to vertex set of G when some K independent sets will be great for us.

Because if you picked up K independent sets say I 1, I 2, I K and if the union is equal to V of G great for us, because once it means I have picked up K independent said that can cover a whole vertex set. Because then covering implies, partitioning implies calculate so, this is great. But when it is not this great?

(Refer Slide Time: 45:45)

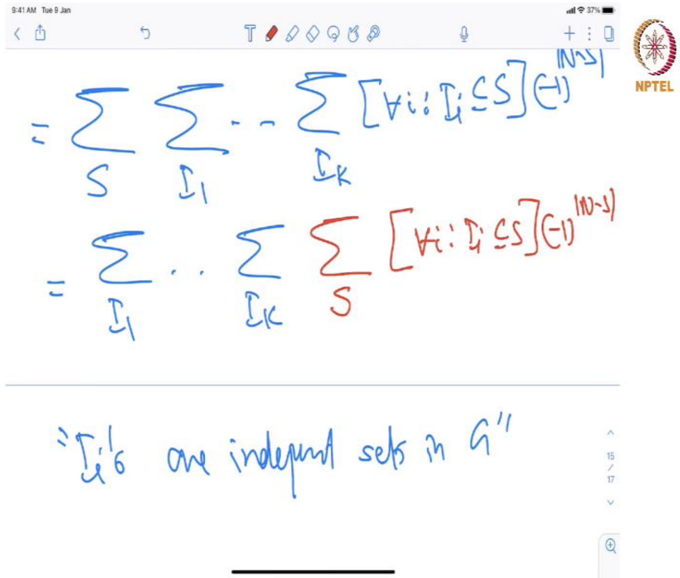


If union of V I is a proper subset of it, because the union can only be. So, if I look at the whole vertex set and like for S = N let us pick up K independent set. These independent sets are great if I have picked up K independent set which is which can cover the whole my vertex, but it is not good for me if it is proper subset. So, now what we are what basically I will tell you that look in this expression only thing which will matter is those counting for which like.

So, the only those ways of selecting K independent set will contribute who could equal to VG and rest they will like rest of them are counted equal number of times with minus 1 equal number of times with plus 1 and hence they will cancel out each other so, that is a goal of doing this. So now, let us try to express left hand side slightly better slightly differently rather so, what I can say so you are what is this summation minus 1 n minus cardinality of s g of s to the power K.

What is g of s to the power K it is some number. So, some number to the power K is nothing but some number times some number, times some number, times some number and that number is nothing but summation of 1 + 1 + 1 + 1 + 1.

(Refer Slide Time: 47:36)

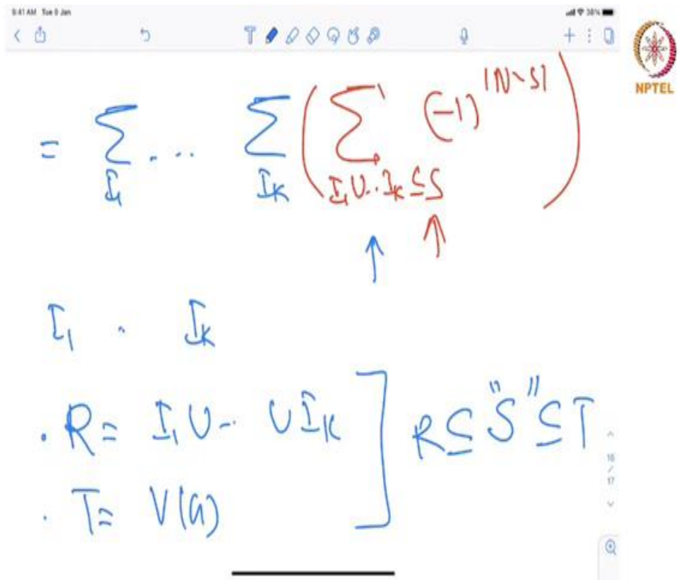


So, we can write this left hand side is equal to summation is going from s summation I selected an independent set I dot dot dot summation I selected K independent set. And what is the property that for all i this is your that notation for all i, I i belongs to set S and minus 1 N - S exactly the same thing. That is it so, like what did this means look at all the independence. So, suppose like among all the independent sets of my graph look at those who belongs to S. So, you are summing over those so that so, what is this left hand side?

So, it is nothing but minus 1 to the power N - S into g of s. And what is the summation again you are going over the same thing so, it is minus 1 g of s to the power square. So, if you do K times you will get summation over s this. Now but I could also do this slightly differently, I could go over all independent sets of my graph K independent sets of graph and only count this when each of these I 1 to I K belongs to S. So, what I mean by this so, this is I could write this summation I 1 dot dot dot summation I K so where these I i's are independent sets in graph G.

So, I am going to do summation over K independent sets but they should only contribute when but now so I first collected K independent sets and now I am going to sum this is important point I am going to sum over S and now so I took up K independent set now these K independent sets could like this K independent set could be contributing to several S? So, this is summation for all i I i is a subset of S - 1 N - S Now, if you picked up K independent set now you ask yourself if we have picked up K independent set how many s will you contribute to? You will contribute to those K those S is for which I 1 to I K is a subset of S. So, we can write down.

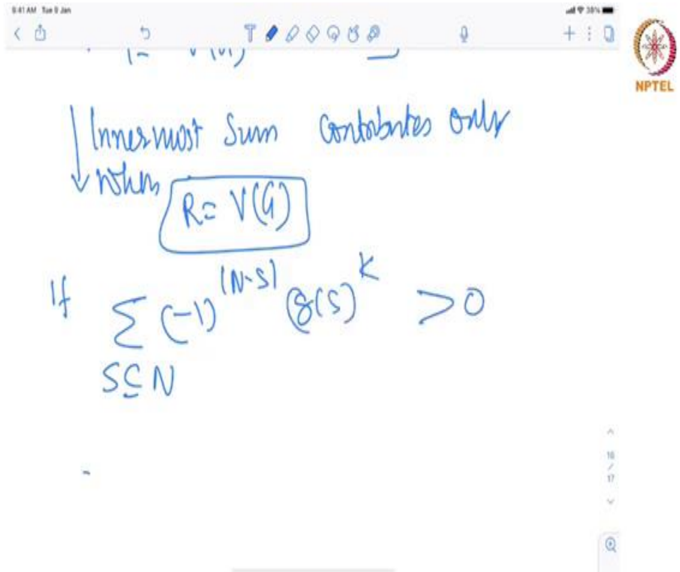
(Refer Slide Time: 50:27)



So, this could be again written as I K and summation over. So, this is going to contribute to all but look I 1 you also part of S, I 2 is also part of S, I K is also part of S. So, this is only going to contribute to those sets S right for which their union is contained inside this. And then this is now look at this inner some you so now let us look at fix some I 1 and I K, let us fix this I 1 to I K let us call this set they are fixed some cases. Let us fix I 1 to I K some K independent sets. Now let us call R = I k. So, this is my R and my T it is vertex set of G.

So, notice that these K independent sets are going to contribute when this; K independence this I 1 to I K are going to contribute for all S which contains R and which is properly contained inside T. So, this is vary. Now we know that this so, you fix I 1 to I K then once you are fixed I 1 to I K then their summation is for all s that contains this.

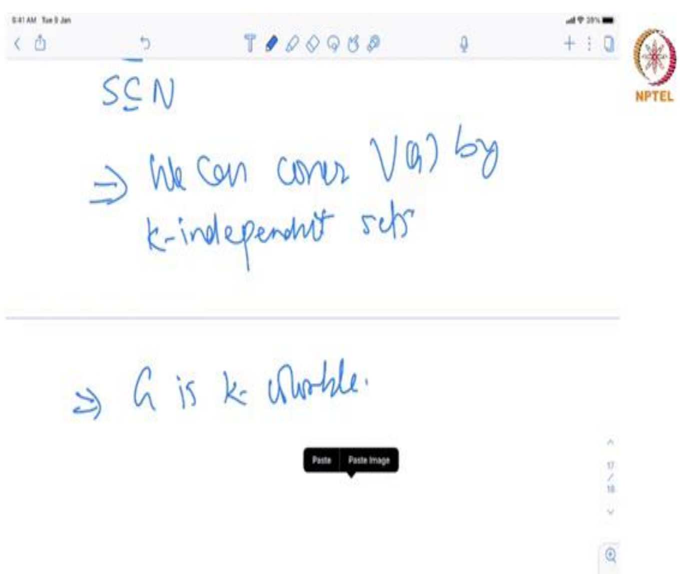
(Refer Slide Time: 52:29)



So, the inner most some contributes only when R = VG. So for all so, now what we have learned? We have learned that look at this inner sum. Now any covering of K independent sets that is not equal to vertex set of G that is counted equal number of times with negative sign and equal number of sign with positive sign and hence that guy contributes 0. So, the only people who contribute here are those only people who contribute here are precisely those.

So, only thing who is contributes are the covering of like only things which contribute to this like non 0 way or yes it is when I have I 1 to I K which covers the whole vertex. So, what we know so if so we have learned that if summation this S subset of N - 1 N - S g of s K is greater than 0.

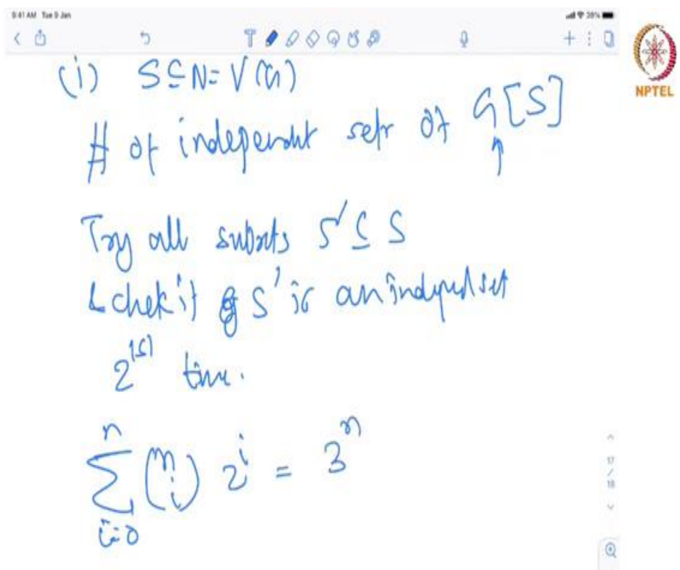
(Refer Slide Time: 54:14)



Implies we can cover vertex set of G into R like this you get the 0 if and only if; not rather we should not say that. We can cover VG by K independent sets great. And now we know that if we can cover VG by K independent set implies G is K colorable for the reverse direction is very simple if G is K colorable implies we can cover VG by K independent set if we can cover VG by K independent set then this sum is always going to be non 0. In fact, when we have G is K colorable then we can cover VG by K dependent set check disjoint partition great.

So, we have been able to show that if graph is K colorable if and only if following combinatorial identity is true. Now let us try to see how we can use it this combinatorial identity algorithmically. Now let us try to so, what we would like to do we would like to compute this quantity what did this come computing this?

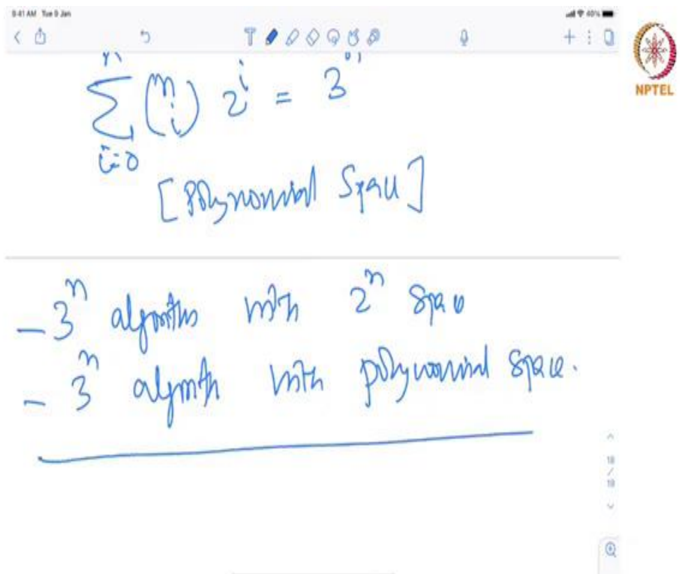
(Refer Slide Time: 55:45)



So, basically what this quantity computes let us first write it down it computes for all S subset of N which is vertex set of G number of independent sets. What is it come so, it is basically takes number of independent sets of graph induced on S fine. I mean and so, we can do this very easily you look at graph induce. Try all subsets S prime subset of S and check if check S prime is an independent set. So, you look at all like non trivial independent sets S prime subset of S and check how many of them turn out to be independent.

So, this can be done in time 2 to the power S time. So, in 2 power S time we can compute g of S and take it K is power and look at this summation so, we can evaluate this summation in time submission again we can do the same trick n to i, i going from 0 to n 2 power i and that is like 3 power n. So this is so, we did not get anything new when we try this inclusion exclusion formula then what we already knew before because of this but notice that this algorithm is a polynomial space algorithm.

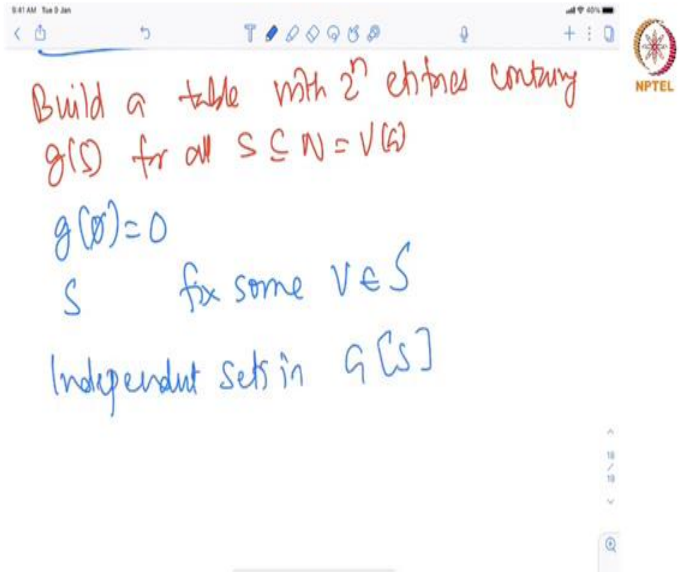
(Refer Slide Time: 57:32)



Because all we need at any point of time is like you go through S 1 by 1 and keep a sum of what is the current evaluation is so that is like 1 memory and for you look at the current graph current S and you enumerate all subsets 1 by 1 again you have a counter and that is how you can compute g of S in poly space. So in fact, so what we get before 3 power n algorithm but with 2 power n space, so what did we get before? So, we gave 3 power n algorithms with 2 power n space.

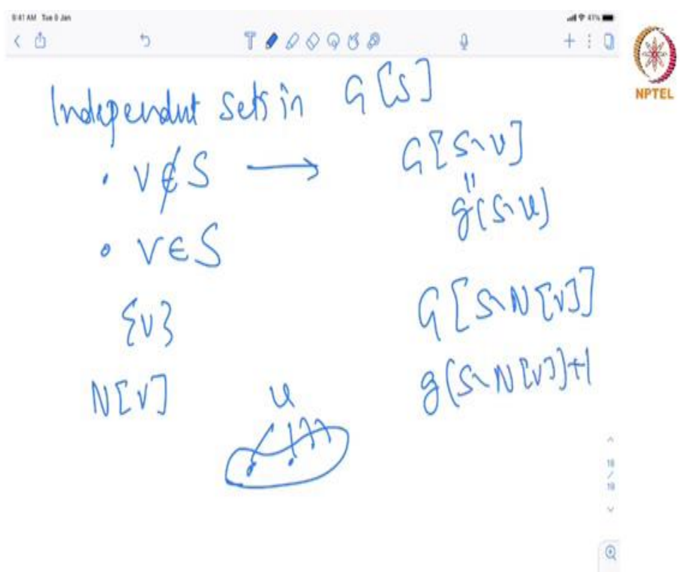
Now we have given another 3 power n algorithm but with polynomials space. Now what I am going to tell you is that actually we can do this computation much more easily. So, rather than doing this what we will do is that.

(Refer Slide Time: 58:48)



We are going to before even running this algorithm, we will build a table with 2 power n entries containing g of S for all S sub set of N which is equal to VG. And to do this such a thing I can build a recurrence. And how will it build a recurrence first of all notice that g of empty is 0, now look at some S and fix some V in S. So, now independent sets in graph induced on S are of 2 types.

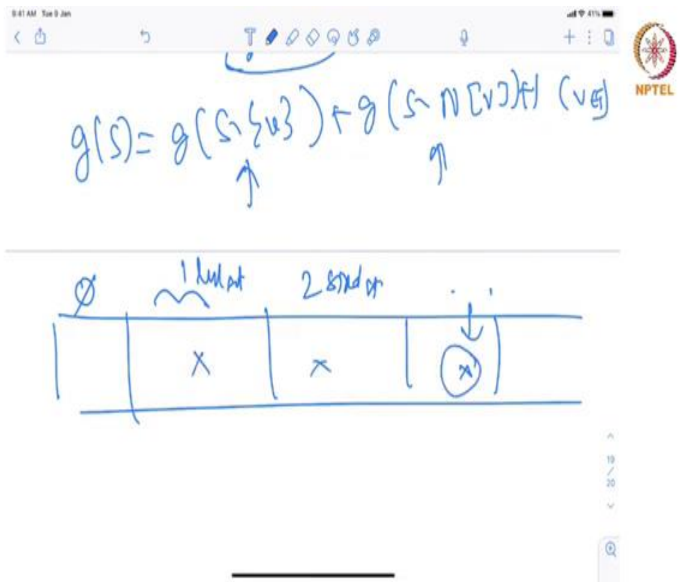
(Refer Slide Time: 59:42)



One those who does not contain V and those who contain V. Those who do not contain V are same as that those who are independent set in graph S - V which you have stored would you have this is same as g of S you stored here. Now if you contain V then there are so, first of all V itself could act as an independent set or you have to find an independent set if you contain V then none of the close neighborhood of V can be contained.

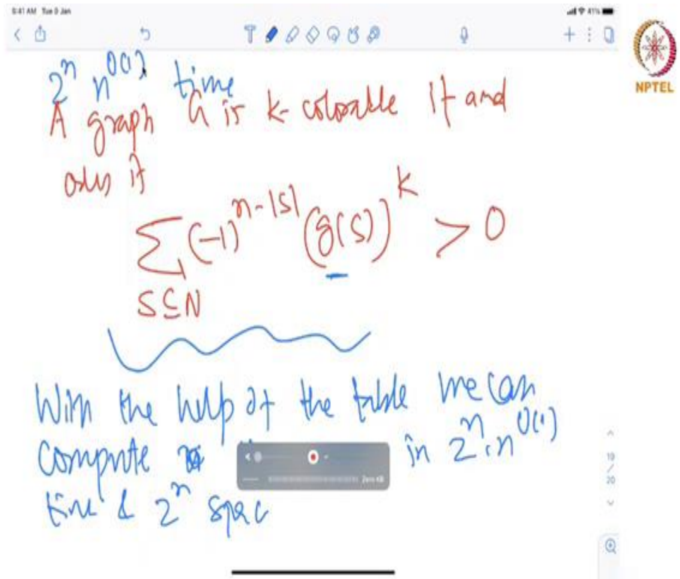
Because V is there none of its close neighbor set which implies that in this case you are looking for you compute the independent set in graph g of S N - 3 add V 2 that so, that is 1 family of independent set plus you have to also add 1 for V itself. So, this is nothing but g of S - N V + 1.

(Refer Slide Time: 1:00:56)



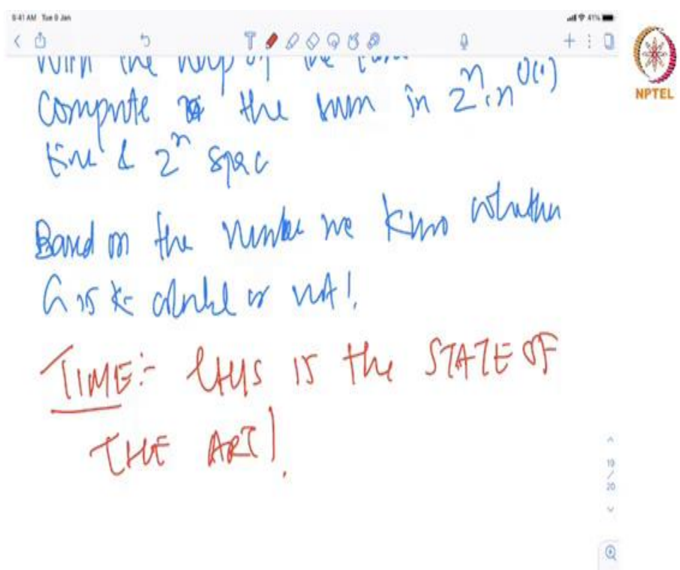
So, that implies that immediately gives us a recurrence g of S is nothing but g of S - V + g of S - N of V + 1. So, this belongs to V in S that is it so, once I have this recurrence then I can again fill this table. In so first increasing order like 5 then all 1 length sets, 2 sized sets. And at any point of time you are looking you just need 2 entries from your array which you have already pre computed. So, each index can be computed. So you like so, these sets are indexed by a sets of your vertex set and each to compute each index set you only need to look at 2 previous entries.

(Refer Slide Time: 1:02:02)



So, you can fill this table in 2 power n n to the power of O of 1 time. And now once you have computed this, now that you have computed this table g of S, now you go 1 by 1 and like you valuate these numbers and just check this. So now, with the help of the table, we can compute with the help of this table, we can compute the sum in 2 power n n to the power O of 1 time and 2 power n space. 2 power n time and 2 power n space.

(Refer Slide Time: 1:03:04)



And based on the number we know whether G is K colorable or not. So, in terms of time this is the state of the art. So, in the next lecture, we will continue our further inclusion exclusion but this time I will give you the classical that intersection way or union way of doing the inclusion exclusion and then we will take it from there and give another example of inclusion exclusion and then well move to the different techniques in designing algebraic algorithms.