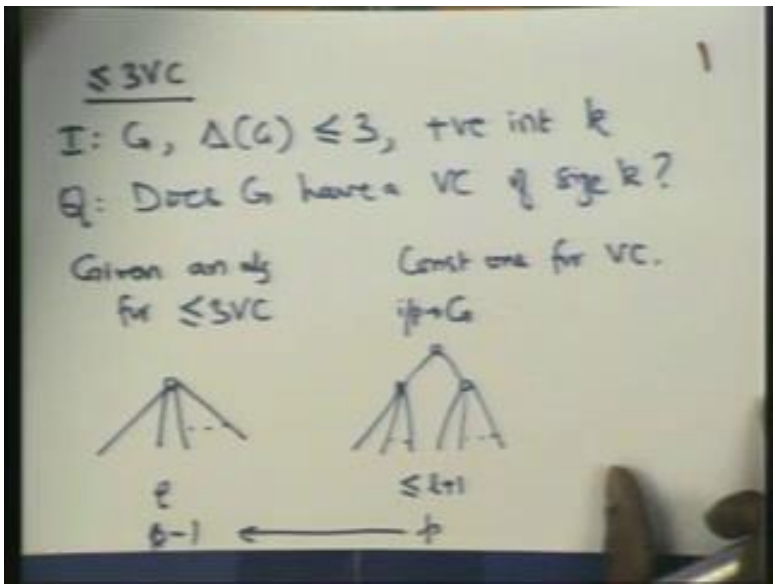
Design and Analysis of Algorithms
Prof. Sunder Vishwanathan
Computer Science Engineering Department
Indian Institute of Technology, Bombay

Lecture - 30
NP-Completeness - 5

(Refer Slide Time: 01:00)

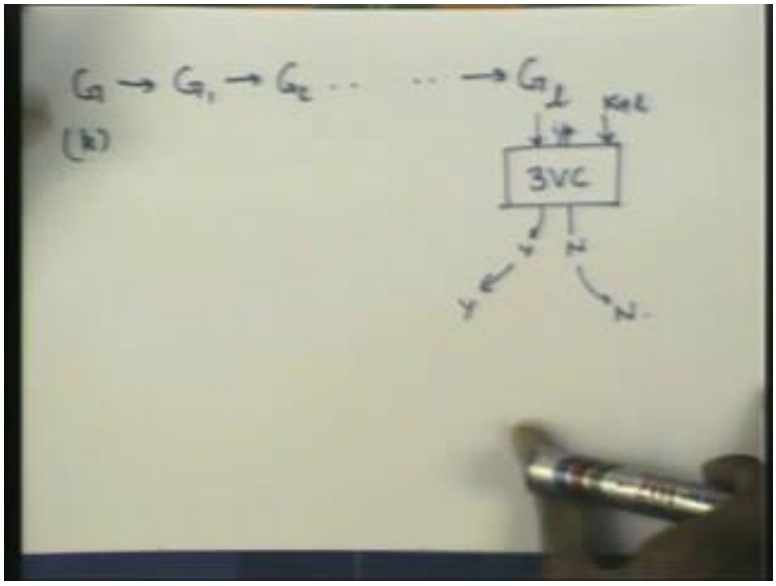


We were looking at the problem less than equal to 3 V C. So, this problem the input is a graph G such that delta G, which has the maximum vertex degree is at most 3 and the positive integer k. The question we ask is does G have a vertex cover of size k? Our objective was to prove that this is n p complete; it is easy to see that this problem is an n c; the proof is very similar as for the vertex cover. The fact that delta G is less than or equal to 3 does not change anything. So, now we need to prove that if there is a polynomial time algorithm for less than or equal to 3 V C. There is one for everything in it in n p; we have seen that it suffices to prove that if there is a polynomial time algorithm for less than equal to 3 V C. There is one for b c that is what we were doing. The trick was this, so we are given a algorithm for 3 V C. So, we are given an algorithm for less than equal to 3 V C. So we want to construct one for v c, so we take the input graph let us say G.

Now, this could have vertices of varying degrees especially vertices with degree greater than 3. If the vertex degrees are less than equal to 3 then we have no problem. Now, what

do we do for large vertices with large degree, we use the split operation. So, supposing I have a vertex with large degree, what you do is you sort of split this vertex into 3 parts. Now, some of the neighbors are attached here and some of the neighbors are attached there. Essentially this is broken up into 2 parts some of them are attached here some of them are attached here. Now, we saw that if this has a vertex cover of size let us say l this has a vertex cover of size less than equal to l plus 1. And if this has a vertex cover of size p, this has a vertex cover of size less than or equal to sorry p minus 1. So, this is the effect of splitting a vertex into these 2 parts. So, the vertex cover goes up by only 1 while we have reduced the degree of this vertex.

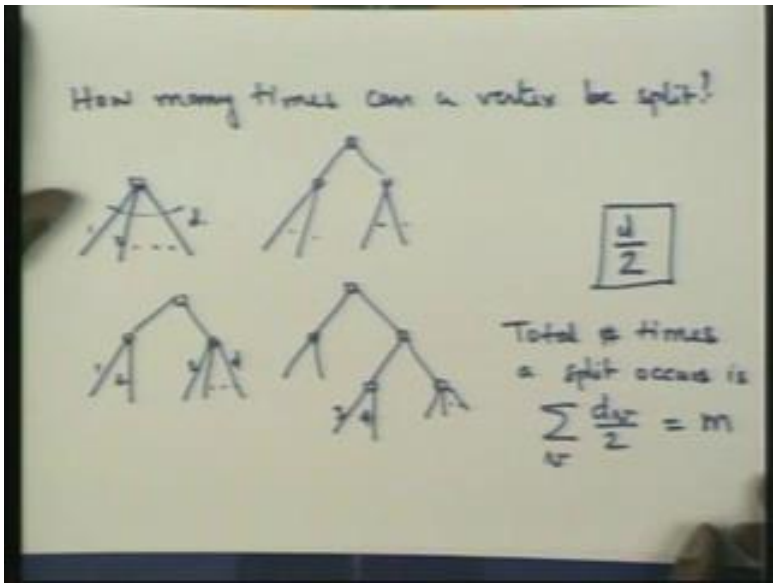
(Refer Slide Time: 04:01)



So, the algorithm is as follows, so you take G and then construct keep splitting vertices as long as there are vertices with you know degree at least 4 G 1 G 2 so on. So, maybe I will do this l times so this is done l times. So, and in G l I know that there is no vertex with degree greater than 3. So, G l is applied on this algorithm for 3 V C. So, this is now, input to an algorithm for 3 V C and the k, so initially k had a k here. So, now I feed in k plus l and if it says yes then I output yes, if it says no then I output no, so this is the entire algorithm. So, I take G I want to find out whether it has a vertex cover of size k. I do this sort of transformation I get final graph G l I feed this into 3 V C with my new k as k plus l. If it says yes then I say yes if it says no I say no. Now, if this runs in polynomial time I want to show that this entire thing runs in polynomial time. The first thing is that there should not be too many of these steps right if there are large number of these steps the

overall algorithm may not be polynomial may not run in polynomial time. So, let us first bound l, so how many times can a vertex get split into 2 parts. So, supposing I have a vertex with let us see this.

(Refer Slide Time: 05:55)

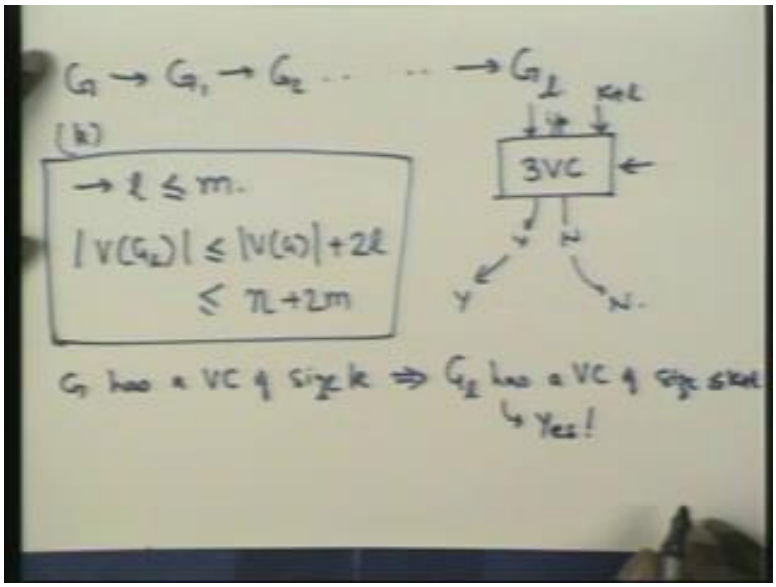


So, how many times can a vertex be split into this is the question we want to answer. So, supposing I have a vertex with degree d. So, I could split this once twice, I mean how many times we do this. Well, note that each time the maximum degree will keep decreasing by at least in fact, by 2. So, the maximum number of times I am going to do this is d by 2. You can do it in fact, better by splitting it in half etcetera. So, the maximum times you will do this is actually d by 2 I hope this argument is clear let me repeat this. So, each time I split right the maximum, so if I just look at these graphs the degree of this vertex now, is at least at most d minus 2 rights and the next time I split one of those vertices. Again I will decrease by 2 and so the maximum number of times I do this is d by 2.

The other way to see it is this. So, let us say I have 1 2 up to d first time I split it like this I have 1 2 and the rest 3 on up to d. And now, for this vertex I again split it into 2 parts. So, this portion remains as it is. So, this portion now I split as let us say this is 3 4 and then and so on. And you can see that this is done at most d by 2 times. So, the number of times the vertex of degrees d is split is d by 2. So, the total number of times, so the total number of times a split occurs is summation over all vertices d v by 2 and that we know

that is nothing but number of edges in the graph right. So, this half comes out and sum of the degrees of the vertices is twice the number of edges right. So, the total number of times the split occurs is at most if the vertex has degree 3, of course it does not split.

(Refer Slide Time: 09:07)

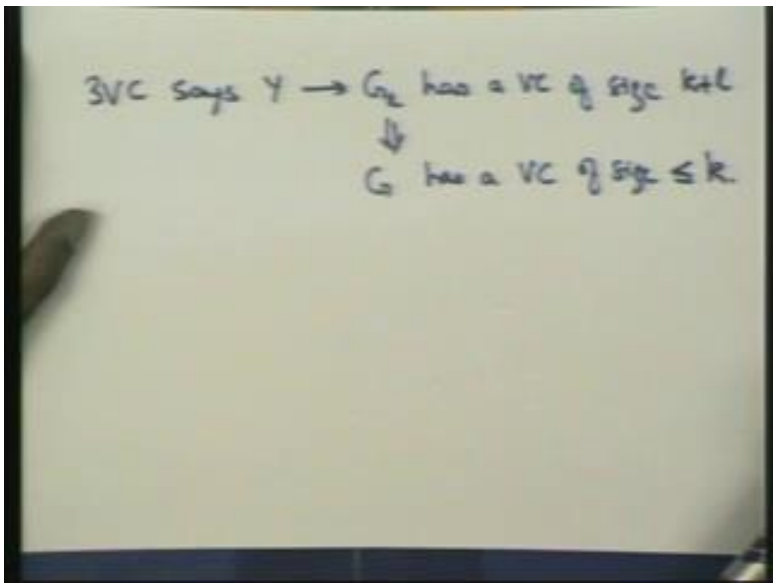


So, here let us go back to this, I know that l is at most m and each time you split you add 2 new vertices to the graph. So, the size of G l v of G l is less than equal to size of v of G and each time you split you add 2 new vertices plus twice l. So, it is at most n plus twice m, the degree of each vertex in G l is 3. So, the number of edges is also some constant times n plus 2 m. So, the size of G l, so the number of edges in G l is polynomial in the size of G. it is not too big you do not do this too many times and final graph that results size of that graph is not too big. So, any running time which is polynomial in that size is also running time is polynomial in the input size. So, this thing runs in time, so we look at 3 V C this runs in time, which is polynomial in this input size of G l, but that is also polynomial in size of G.

So, the whole thing runs in time, which is polynomial in size of G. And now so we need to know show that this works. So, I hope that you are all convinced that this whole thing runs on polynomial time if 3 V C runs on polynomial time. Now, we need to convince ourselves that if this says yes then answer there is yes it says no then the answer is no. Now supposing G has a vertex cover of size k, so let us prove both ways supposing G has a vertex cover of size k. If G has a v c of size k, so this implies that G l has a v c of

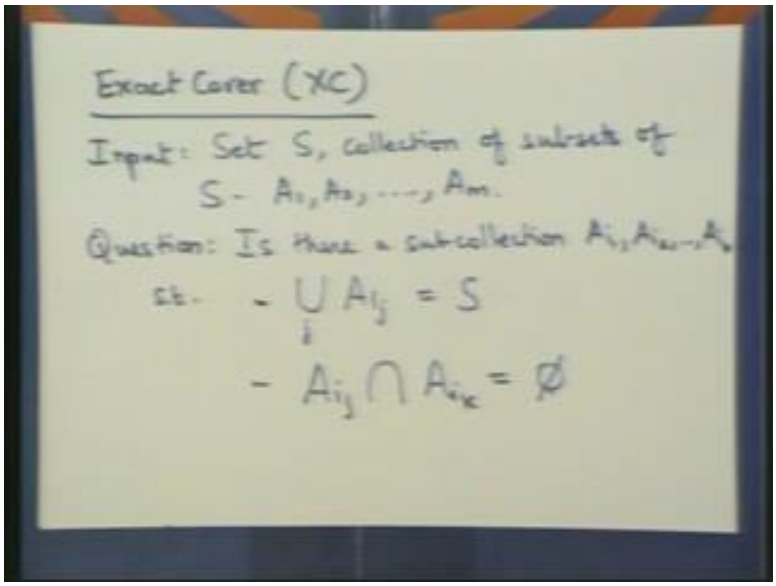
size less than equal to k plus l right. So, this has a vertex cover of size at most k plus l. So, it will say yes, so this answer will be yes, so this is fine. What we need to do is also the other way round that it does not have a vertex of size k it will answer no, but it is better to do the other way round. So, if this says yes, if 3 V C says yes, this means that G l.

(Refer Slide Time: 12:26)



So, 3 V C says yes, which means G 2 has vertex cover of size k plus l. And we know that this implies that G has a vertex cover of size at most k right. So, G has a size of at most k, so this is also same which means G has a vertex cover of size k. So, both ways we have proved that the algorithm worked out. If it has a vertex cover of size k well, here its k plus l G has a vertex cover of size k this implies that G l has a vertex cover of size k plus l. We have done, which means G has a vertex cover of size k then it answers yes and if it answers yes then G must have a vertex cover of size k. So, this proves that 3 V C is.

(Refer Slide Time: 13:55)



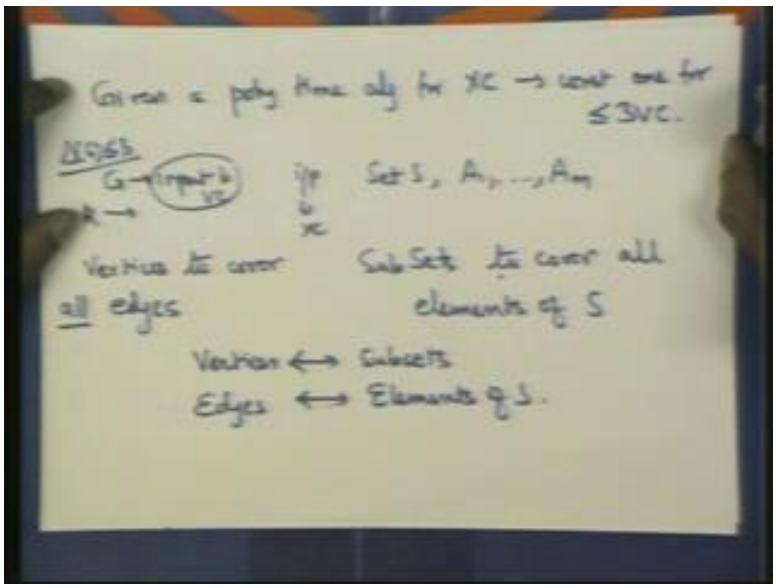
So, our next problem is exact cover. So, called exact cover XC exact cover here, the input is set S and the collection of subsets S A 1, A 2 up to A m there are m subsets of s. The question we ask is the following is there a sub collection of these sets. So, that the union is S and the sets in the sub collection are disjoined is there a sub collection. So, let us say A I A I 1 A I 2 up to A I p is that 2 things the union must be yes. The union for all j A I j is yes second thing is that A I j intersect A I x i s null set.

So, you need to pick some sub sets of some of these sub sets in these collections. So, that each element is present in one of them and exactly one of them. So, each element is covered and covered exactly once. So, that is where you get the name exact cover is there a sub collection, which exactly covers the set S, which covers set S in the sense that union is S union of I j is S. And it is covered exactly in the sense that each element is covered exactly once. So, these must be disjoint. So, is this problem n p complete the answer is yes as you must have guessed, because those are the problems we have discussing.

So, let us prove that this problem exact cover is indeed n p complete. So, the first thing to note first thing is to prove that exact cover is in n p to prove that this is in n p. So, what is A S input A S input is just collection of I need a collection. So, that there is some sub collection. So, that a union is S and the sets in the sub collection are disjoined. So, I guess it must be clear what prove the provers must supply the verifier. So, that the

verifier can verify that there exists such a sub collection. In fact, the prover just gives such a sub collection the prover tells the verifier what are the sets in the sub collection the verifier takes these sets. He verifies that they are disjoint and he also verifies that the union is S, which means every element of S does appear in one of the sets in the sub collection. So, this proves that this problem is in n p. The blame will prove it is n p hard is to show that if there is a polynomial time algorithm for exact cover. There should be a polynomial time algorithm for less than equal to 3 V C.

(Refer Slide Time: 18:26)

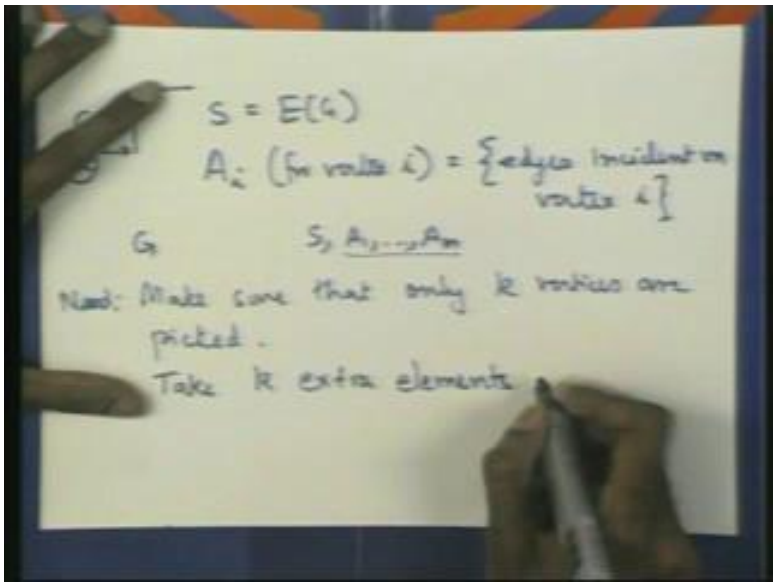


So, let us do this. So, given a polynomial time for x c I want to construct one for less than equal to 3 V C. This is our goal, so how do we do this. So, we want to construct for one less than equal to 3 V C. So, I guess the algorithm will take a graph as a input, there is also this k, we know that delta G is at most 3 and we want to know. We want to somehow figure out whether this graph has a vertex cover of size k or not. That is the problem we really want to solve and what we can use is the sub routine is this problem. So, somehow we need to use sub routine to X C to solve this problem on graphs. So, let us write down what a solution to x c and solution to vertex cover look like.

The input to X C is set s and A 1 through A m. This is the input this is input to V C now; here what I want to know is vertices to cover all edges. Here I want sets to cover all elements by sets I mean from this collection A 1 to A m right. So, somehow vertices there must be some relationship between the vertices and the sub sets, so let me say sub

sets. There must be some correspondence here and the edges must correspond to elements of S. If there is such a correspondence then it looks like, you know that there is some similarity between these 2 problems. This is what we would like to we would like to exploit, so let us take to attempt at this problem.

(Refer Slide Time: 21:43)

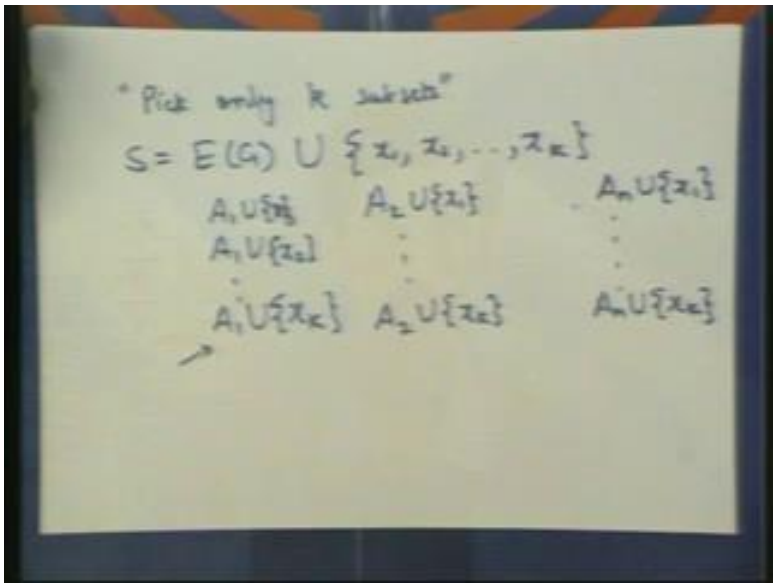


So, we know that the elements of s are edges and blind thing and then we take G is the input. So, now, I look at G and I need to convert this input into A input for exact cover. So, I say the set s is nothing but the edge set of G and now, the subset must correspond to the vertices. Each subset here must correspond to a vertex in G and what is the natural sort of correspondence for vertex I, I add a set A I for vertex I. This is nothing but the edges incident on vertex I, if I take A I to be the edges incident on vertex I. Now, I have this correspondence right. So, I have the graph G here and I have s A 1 through A m picking vertices to cover edges corresponds to picking subsets, which cover elements of S.

Elements of edges are covering edges and each subset is like picking a vertex in G. Picking a sub set here is like picking a vertex in G. Now, there is a small problem first problem is I need to pick, you know there is this k floating here. So, I should somehow make sure that make sure that only k vertices are picked. We need this if I just do this there is no sort you know I could pick as many of them I want right. So, a vertex could be I could pick as many of these subsets. So, this fact that in the original problem, I

should only be allowed to pick k vertices is the does not reflect in this transformation. So, how do we fix this problem? This problem is actually fixed easily. So, what we do is this, so we take k extra elements.

(Refer Slide Time: 24:26)

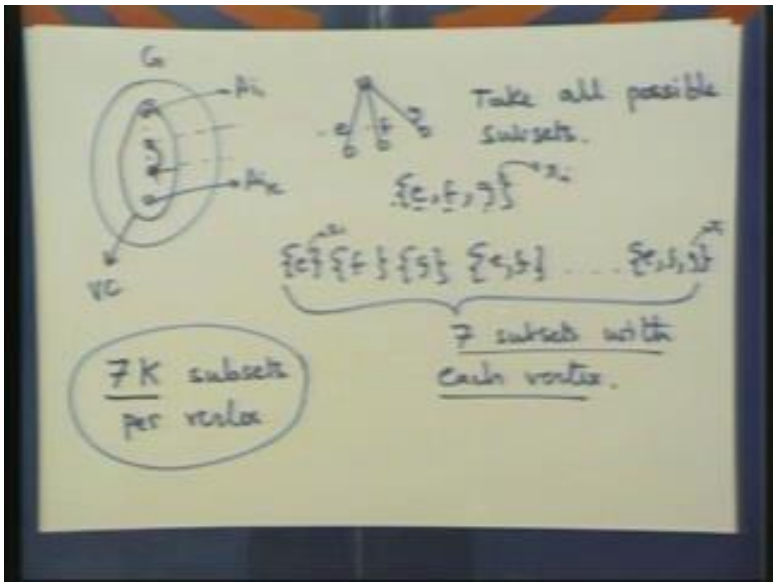


So, take k extra elements we need to what we are shooting for is pick only k subsets. This is what we want to do? So, take s now apart from G there will be some k extra elements x 1 x 2 so on till x k and supposing the sets k 1 through what I do is I take the, I change the collection of sets. I take A 1 union x 1 A 1 union x 2 so on A 1 union x k similarly A 2 union x 1 so on up to A 2 union x k and finally, A n union x 1 up to A n union x k. Well, n is the number of vertices in the graph that is why it is n here and not m, m is the number of vertices. Well, the number of collections has now, grown to a factor of k initially, we had n sets and now, we have k times n. But now, let us see supposing I pick up an exact cover here supposing I pick up an exact cover from this collection from these I can I can pick any one of them right.

But if I look across row, I can pick only one of them. From this I can pick only one from this, I can pick only one and so on. And from this I can pick only one, so totally I can pick only k sets. So, let me say this argument again now, when I look at this collection then in each set one of x 1 x 2 x 3 x k occurs. Each set one of these k elements must occur and if x 1 occurs in many states, I can only pick one of them, because x 1 has to be covered exactly once right. In fact, exactly one of these sets, so I have to pick a set,

which contains S 1 as to pick a set, which contains s 2 I have to pick a set, which contains x 3 and so on up to x k. These are k sets and I cannot pick anymore, because any other set has to contain one of these elements x 1 x 2 up to x n. So, this trick force, you to pick when you pick an exact cover this forces, you to pick exactly k of these sets. So, that is taken care, so is there any other problem well unfortunately there is the problem is that in vertex cover.

(Refer Slide Time: 27:59)



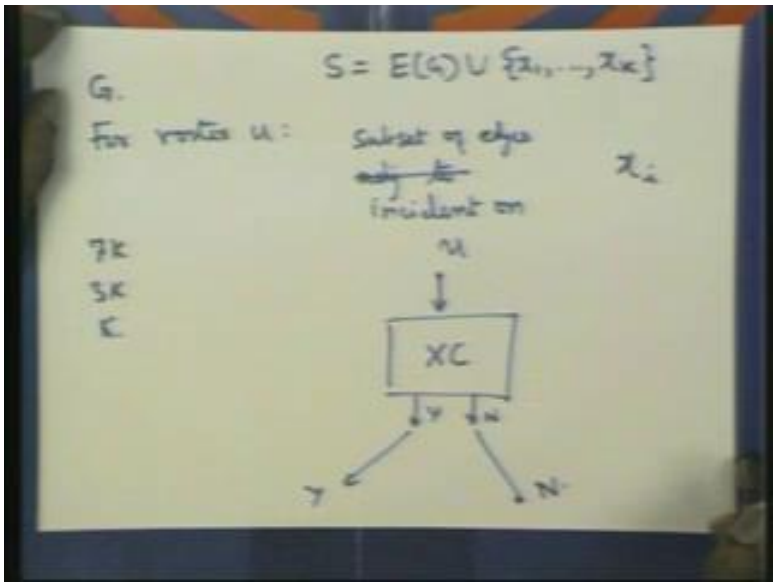
So, let me sort of tell you what the problem is, so your problem is this, when you look at a graph I look at vertex cover. You know it may, so happen that there may be a edge between 2 vertices in the vertex cover. So, this is the vertex cover, it may be, it may happen that there is an edge between 2 things in a vertex cover. So, when I pick the corresponding sets A I 1 and so on. This is A I k, when I pick the corresponding sets, then you know this edge gets covered twice. It gets covered when I pick this vertex at this set; it also gets picked when I pick this set. So, this is the problem. So, what we would like to do is when you get to this vertex not pick this edge, but may be pick the other edges, which are not yet been covered. So, I would like to pick these sets 1 by 1 when I get to a vertex I would like to pick those sets those edges, which are not covered and the solution is actually simple.

So, supposing I had recalled that the degree of each vertex is 3 supposing I had a degree of vertex 3 right. So, there are 3 edges, which are adjacent let us say e f and g now, it

could, so happen by the time I get to this vertex. I would like to pick this vertex, these edges sum of these edges are already been covered by other vertices. Then I would like to leap at G let us say if e and f are already covered I would like to leap at G for this vertex. If only e is covered I would like to pick only f and g and similarly if none of them are covered then I would like to you know have a set, which contains all of them. So, the trick is to just take all possible subsets. So, take all possible subsets. So, initially I had with the vertex, I had the set e f g. Now, I will have many of them instead of this, I will have now e f g and then all pairs e f so on and finally, e f g.

So, there are 7 of these with each vertex well actually this would have x 1 x 2, so on up to x k also. So, that also has to be put in. So, this set would have e f g in one of the exercise remember our previous construction that x i would also go into each of them. So, it us not 7 subsets, it is 7 k times it is 7 k with each vertex I had k subsets 1 for x 1 1 for x 2 and so on. Now, I have 7 k subsets per vertex I have 7 k subsets per vertex, if the vertex has degree 2. Then it would be less then it would be 3 k, because there are if I take a size 2 the number of distinct subsets, which are not empty is 3. If the vertex has degree one then I would just have one subset, which would translate k other subsets. So, when the vertex has degree k degree 3, I have 7 k subsets per vertex.

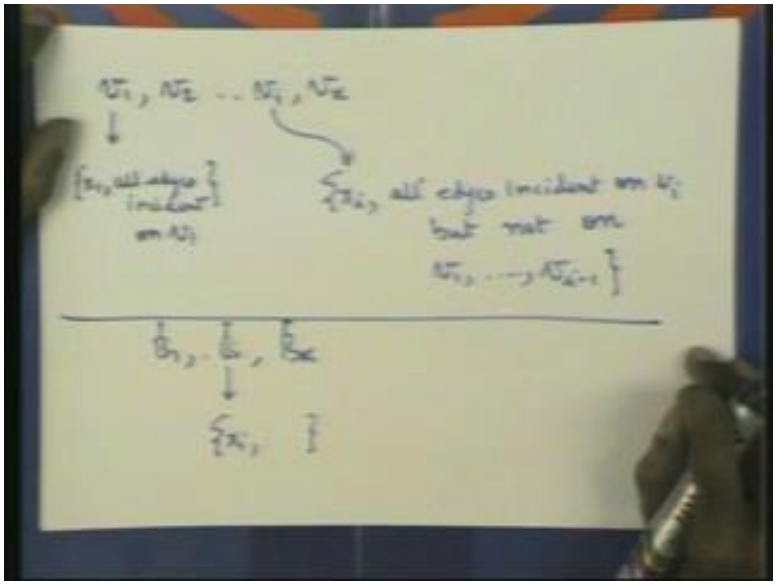
(Refer Slide Time: 32:23)



So, this is translation I take this graph G and for every vertex for vertex u. I must say that the set is nothing but E of G x 1 through x k for a vertex u. I would have either 7 k or 3 k

or k say depending upon the degree of the vertex. So, what is the typical, what does the typical set look like? So, it is a subset of edges adjacent incident on u and one of the exercises. So, I take a subset and x 1 subset x 2 subset and x 3 and so on up to x k. So, this is how each element corresponding to a vertex look like. There could be either 7 k or 3 k depending on the degree of the vertex. And this is now, I take this instance of S and this collection and I feed this into sub routine for x c. If this says yes then I output yes if this says no then I output no. So, this is this is my transformation. Now, to see that this works clearly, if this has a vertex cover of size k I have a exact cover of I have exact cover of this, why is that true?

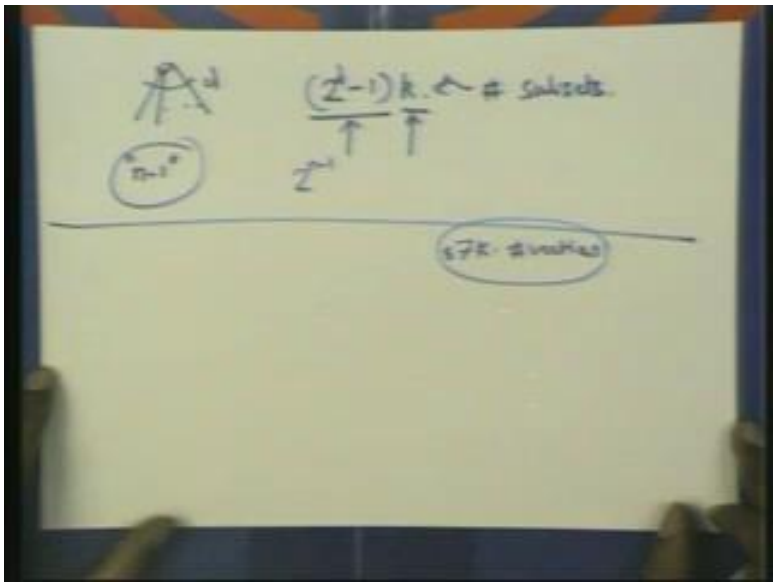
(Refer Slide Time: 34:35)



So, let the vertex cover consist of vertices v 1 v 2 so on up to v k now, for v 1. I put the set x 1 comma all edges incident on v 1 i pick the set with v i. I pick the set x i comma all edges incident on v i, but not on v i so on up to v i minus 1. This is my set which I picked and you can check that this is this is an exact cover x 1 is covered by this set this set corresponding to v 1. x i is covered by v i and so on. x k x 1 through x k is covered and every edge is also covered. Since this is a vertex cover if I take any edge, it must occur between there must be an end point in this there must be at least one end point. Choose the smaller end point supposing it is v I then this edge must have the other end point is either one of v i plus 1 through v k or something else completely different in either case that edge will be part of the set.

So, every edge is covered by by this collection edge is covered here. So, every element of S is covered by this collection. Similarly, if I have a collection which covers let us say b 1 through B k is the sub collection, which covers every element of S. Let us say that B i contains x i x 1 through x k must be covered. So, and they are covered exactly once, so they occur there must be k sets and each of them must contain one of x 1 through x k let us say b 1 contains x 1 and so on. And B I contains x i B k contains x k now, every edge must also be covered. And must occur in one of these sets, which means if it occurs in set B i, it must be adjacent to the vertex I right. So, what I do is the vertex cover just consists of the vertices, which corresponds to each of this each of these sets, which corresponds to B 1. The vertex, which corresponds to B i and so on, so those vertices will form a vertex cover, so this shoes that exact cover is n p complete, why did we pick 3 V C, why not I mean less than equal to 3 V C, why not vertex cover? We have done this with vertex cover well the answer is no at least this reduction does not work.

(Refer Slide Time: 38:06)



The reason is that supposing I have a vertex of large degree some degree d the number of subsets, I have corresponding to this is 2 to the d minus 1 times k. This is the number of subsets, remember how we took? How we got the subsets for the vertex? We took all possible subsets here non empty subsets. And we added for each subset I added x 1 x 2 x 3 so on up to x k. So, these many non empty subsets times k is the total number of subsets we get and this can be just very large right. If I have a graph with degree for instance the complete graph as degree n minus 1. This size can be 2 to the n minus 1,

which is just too much, which is too much and by the time you write this down, it is much more than polynomial type.

So, that will it will not work in polynomial time if you take just about extra cover. So, we want this degree to be bounded by a constant, if we use 3 V C and then this is 7 times k which is not 2 bit. So, the total number of subsets we have is actually 7 k times the number of vertices, this is the total number of subsets is at most this much if its instance if the degree is bounded by 3. That is the reason we needed less than equal to 3 V C we have shown that exact cover is n p complete. The main trick was to first show that 3 V C is n p complete less than equal to 3 V C is n p complete, which is a restriction of vertex cover. Then show that exact cover is n p complete one can actually do it in other ways without going through 3 V C and I will let you try this on your own. The reductions become slightly more involved, but you can still do them. After this we will look at problems where sizes come into play. So, that is the next installment.