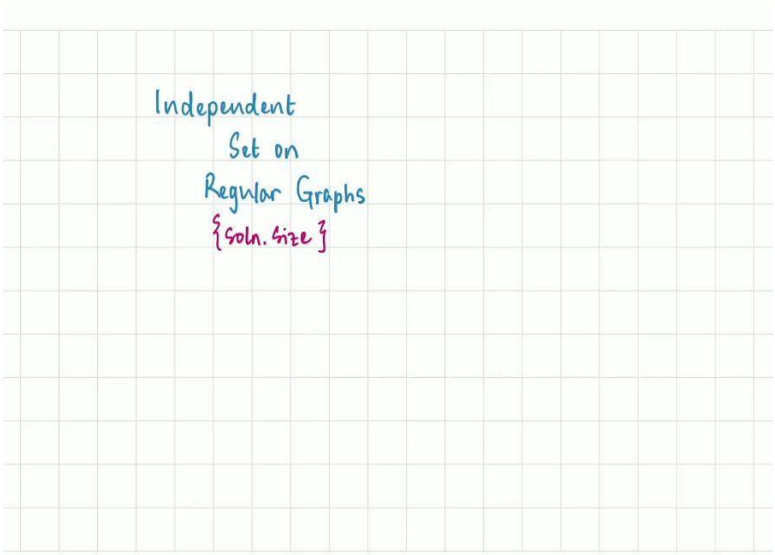
Parameterized Algorithms
Prof. Neeldhara Misra
Prof. Saket Saurabh
The Institute of Mathematical Science
Indian Institute of Technology, Gandhinagar

Lecture – 49
Reductions – Problems as Hard as Clique (PVC, MCC, MIS)

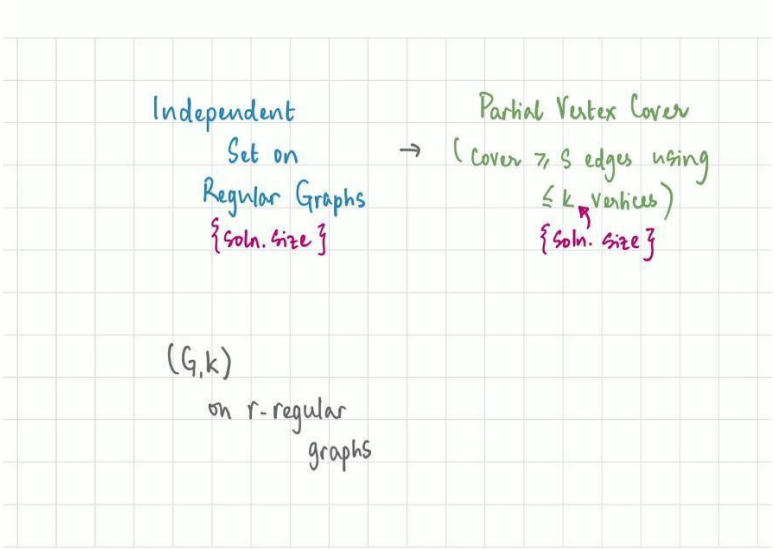
(Refer Slide Time: 00:11)



Welcome back to the third segment of the first module in the 12th week of parameterized algorithms. We are looking at reductions and at the end of the previous module we talked about how solving independent set, even on the restricted class of regular graphs is as hard as solving clique on general graphs. And we are working with this running assumption that clique on general graphs parameterized by solution size does not admit a FPT algorithm.

And the theme of our current discussion is to really examine the consequences of this assumption. So, in this segment I want to use the hardness of independent set on regular graphs in particular to show you the hardness of partial vertex cover on general graphs.

(Refer Slide Time: 00:54)



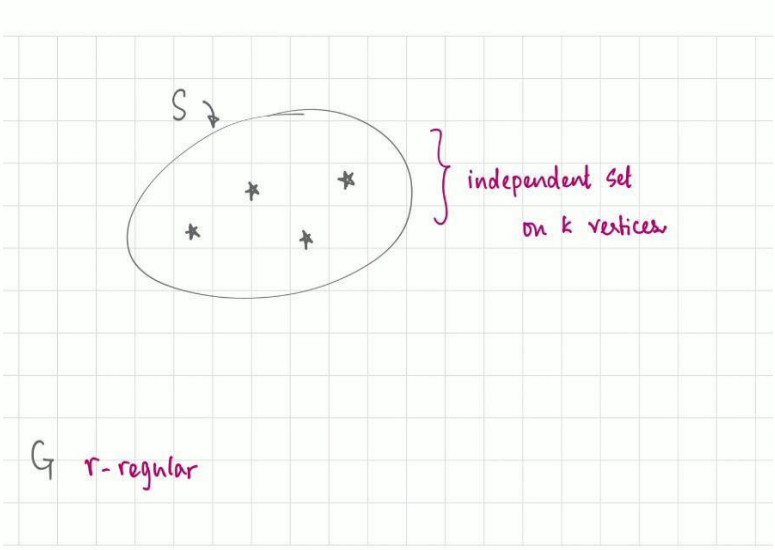
So, what is the partial vertex cover problem it is a very natural generalization of our officially favourite problem of vertex cover. So, let us imagine that you are working with a budget of k and somehow this is a hard constraint you cannot afford to work with more than k vertices and let us say you have a graph where there is no vertex cover of size k. What is the next best thing that you can do?

Well, you now want to think about I could not cover all the edges with k vertices. What is the largest number of edges that I can cover with k vertices? If you choose your vertices carefully you might be able to cover a substantial fraction of the edges in the graph. So, the partial vertex cover is a question in this spirit it says suppose you have a fixed budget k and you have a target s. Is it possible to cover at least s edges using at most k vertices?

Notice that if s = m then this is the original vertex cover problem. So, you could think of partial vertex cover as being a generalization of the vertex cover problem. So, we want to consider partial vertex cover parameterized by k again, where k is the size of the solution and we want to think about well you know what is the complexity of this problem what I want to do in this segment?

At least in the first part of it is to talk about a reduction from independent set on regular graphs to partial vertex cover on general graphs with the parameter k. So, let us just think about what does an independent set on a regular graph look like. So, suppose we have an instance G, k let us look at G.

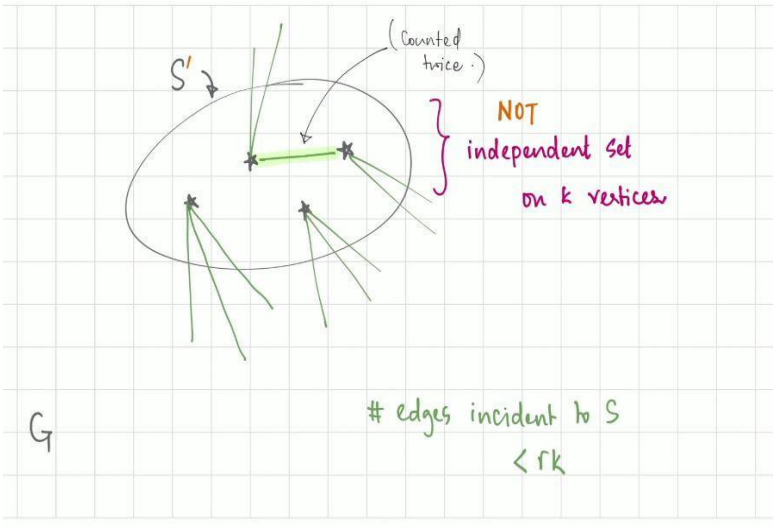
(Refer Slide Time: 02:47)



And let us look at an independent set on k vertices in this graph. What possible connection could this have with partial vertex cover? Well, we already know that the complement of an independent set is a vertex cover. But I do not really want to go down that path because as we know this will involve going from k to n - k and then that is going to get messy. So, let us not think about the complement of the independent set.

But let us instead think about the independent set itself. How many edges does this independent set actually cover? Remember that G is a r regular graph. So, based on this if you have an independent set of size k how many edges do you think this independent set covers take a moment to think about it and then come back and tally your answers with me. Well, this independent set is going to actually cover r times k many edges.

(Refer Slide Time: 03:47)



Because every vertex in the independent set has degree r and in some sense all of this degree is going outside the independent set. And other words every vertex in the independent set has r neighbours and all of these are neighbours are sitting outside the independent set just by the very definition of an independent set. So, no vertex in the independent set can have a neighbour inside the independent set.

So, all of this degree is getting thrown out of the set which means that the total number of edges covered by this set is in fact r times k. Just to really make this more explicit notice that if you were working with a subset of k vertices which is not an independent set. So, let us say that we have S prime, again S prime is a subset of k vertices but this time it is not an independent set there is at least one edge sitting completely inside the set.

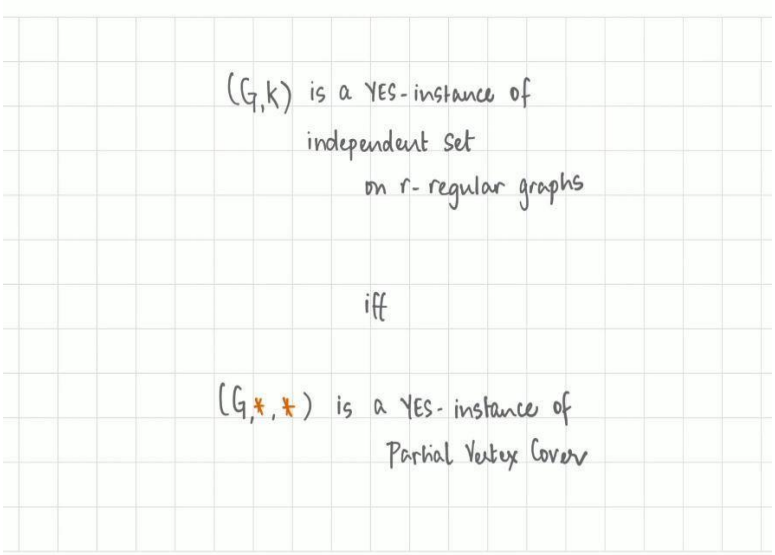
Then notice that such a set will not cover r time scale many edges, because this edge that is sitting inside will essentially get counted twice. So, the number of edges that you will cover will be strictly less than r k, take a moment here if you needed to convince yourself that this is indeed the case. So, once again when I am trying to count the number of edges incident on the set one way that I can do that is basically add up the degrees of all of the vertices.

And then adjust for my over counting so, if there was some edge that got counted twice then I will have to subtract one for every such edge. When all of the neighbours were sitting outside of this set, I do not need to do any adjustment because there was no edge that got counted twice. On the other hand, if there is an edge that sits completely inside the set then I do have to adjust for it.

And therefore, in this sort of a setting the number of edges covered by this set is strictly less than r k. Why am I telling you all this? Because all of this leads up to a hint for how you can transform an instance of independent set on r regular graphs to partial vertex cover parameterized by the solution size. Take a moment here and think about what the transformation could be.

In this example we do not really have to add any artificial content into the graph your graph can be just the same as it is for the source instance but you just have to carefully determine what your s and k have to be.

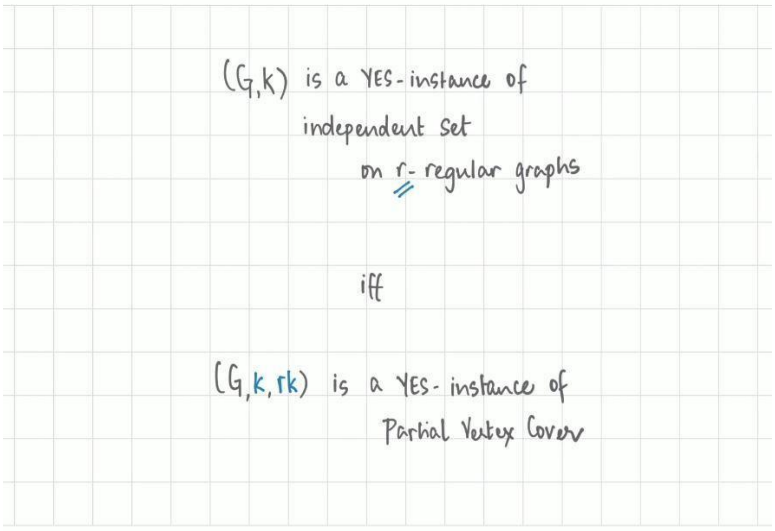
(Refer Slide Time: 06:21)



So, just to make this explicit here is the kind of statement you want to prove. You want to say that r regular graph G has an independent set on k vertices if and only if it also has a partial vertex cover on something many vertices that covers something many edges. So, these 2 blanks that you see on your screen right now should be quite fillable based on the discussion that we have had so far.

So, feel free to pause and take a moment here and commit your answers to how you would complete the statement and we can exchange notes once you are done.

(Refer Slide Time: 06:59)



So, what we want to say is that G has a partial vertex cover on k vertices covering at least r times k many edges where this r is the same as the regularity of the graph that we started with. So, notice that the forward implication which says that if G has an independent set on k

vertices it has a partial vertex cover on k vertices covering at least r k edges follows from the first part of our discussion.

Where we observed that if you have an independent set of size k and the graph is irregular all of these are neighbours for every vertex in the independent set must lie outside. And therefore, when you add up the degrees there is no adjustment to be made and the number of edges covered by this set is in fact r times k. So, that proves the forward direction. In the reverse direction suppose you do have a partial vertex cover on k vertices that covers r k many edges.

Based on this you want to discover an independent set on k vertices. Well, let us just propose the partial vertex cover as our candidate independent set it. Certainly, has k vertices and let us assume for the sake of contradiction that this is not an independent set. That means that there must be an edge sitting inside the claimed partial vertex cover which is supposed to be covering remember at least r k many edges.

We know that G is r regular so, suppose that the partial vertex cover is not independent it does have an edge sitting inside it then based on the second part of our earlier discussion. We see that this partial vertex cover will actually fail to cover r k many edges contradicting our starting assumption. So, at this point we are actually done this concludes the argument for equivalence and notice how the regularity of the graph really came into play and was quite crucial in helping us easily establish this connection.

The other two aspects that we need to confirm for this reduction to be a parameterized reduction include the fact that the parameter is preserved and the fact that the reduction runs in polynomial time. The fact that the reduction is efficient is clear because we did not really have to do anything except specify the values of k and S, it is really the same graph. So, that is straightforward.

In terms of the parameter notice that k goes to k, that is why it is important that we are looking at partial vertex cover parameterized by the solution size. Notice that S which is the target the number of edges that we want to cover is r times k. So, if you were working with S as your parameter instead of k then this would not have been a valid parameterized reduction,

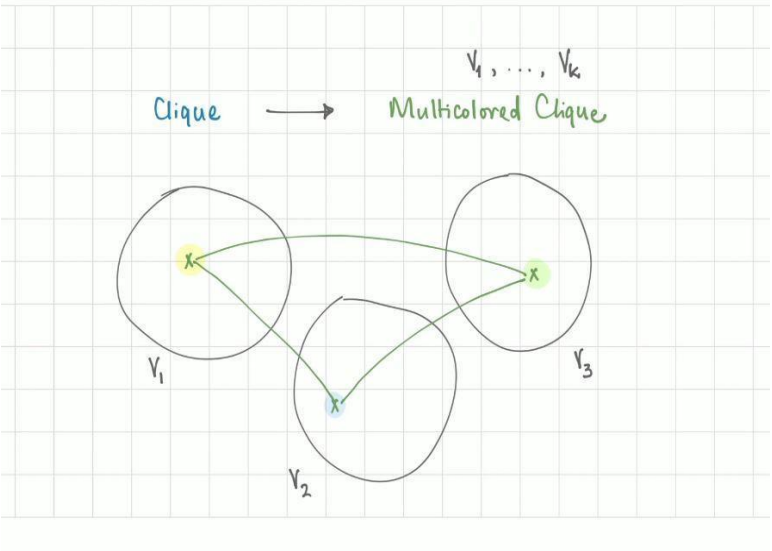
because you get a dependence on r and remember that r is something that is quite independent of k.

It is certainly not guaranteed to be a constant or anything like we discussed last time in fact the hardness that we have for clique. Actually, the hard instances that we generated had a regularity which could potentially be a function of n depending on the graph that you started with. So, as far as the solution size is concerned, we are safe the parameter transforms nicely from k to k.

So, this is a valid parameterized reduction for partial vertex cover parameterized by solution size. And it would not be a valid reduction for partial vertex cover parameterized by the number of edges that you want to cover. So, it turns out that there is a good reason why you would not expect this reduction to work. In fact, partial vertex cover parameterized by S or the target number of edges that you want to cover is actually FPT.

And this is a fun exercise in colour coding if it is something that you have not done before. I would definitely encourage you to think about it as a fun exercise. So, moving on the next thing I want to tell you about is a problem called multi-coloured clique.

(Refer Slide Time: 11:02)



And it is sibling multi-coloured independent set. These problems are very similar to their counterpart's clique and independent set. But somehow, they are apparently superficially reformulated, it turns out that this reformulation just is very convenient especially in the

parameterized context making them extremely popular starting points for a variety of reductions.

So, it is worth knowing about what makes multi-coloured clique as hard as clique and we will see one example of how the multi-coloured variant is a useful starting point for a different reduction. So, for now let us focus on just showing the hardness of multi-coloured clique based on the hardness of clique. So, let me begin by defining the multi-coloured clique problem. Here the input is again a graph whose vertex set has been partitioned into k parts.

And the question is if G has a clique that picks exactly one vertex from each of these k paths. So, the reason this is called multi-colour clique is because sometimes it is convenient to think of the partition as a colouring of the graph and the parts as colour classes. If you do that then a multi-coloured click is essentially what you might think of as a colourful clique borrowing terminology from the colour-coding days.

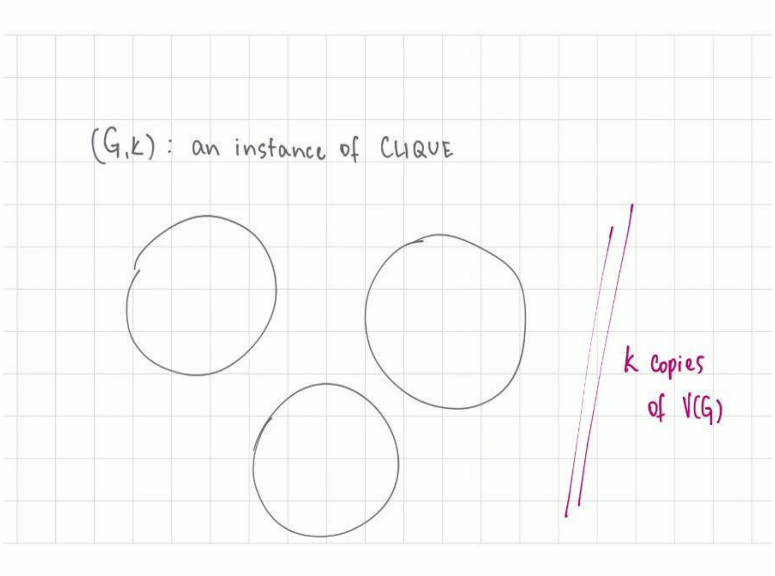
A colourful clique would be a clique that does not have any repeated colours. And what we are claiming here is that finding a colourful clique is as hard as finding a colourless clique in the normal setting. So, notice that this is not necessarily obvious especially given that in the past we have been in situations where we have said that looking for a colourful variant of an object can actually be easier than finding the analogous colourless version of it.

So, it turns out that for clique it is just that the colouring does not make a difference and you are just back to square 1. The way we establish this formally is by reducing clique to the multi-coloured variant. And once again this would be a good place to pause and think about coming up with such a reduction yourself. One hint to work with is to see if you want to make copies of G just like we did when we were working with the reduction from clique to clique on regular graphs.

Here you want to think about how the copies would be useful. Loosely speaking you could think of the copies as corresponding to colour classes and that should lead you to a thought process involving well how should we connect vertices between two copies. The last time that we worked with copies they were all nice and disjoint and they connected to a common pool of dummy vertices.

But here the spirit of the construction is going to be slightly different. So, I hope this pointer helps. Take a moment play with this a little bit and come back once you have done that. So, welcome back hopefully you have had a chance to take a break and think through the reduction yourself. Following up from the hint that we were talking about. Let us actually begin this construction by making multiple copies of our source instance the graph G.

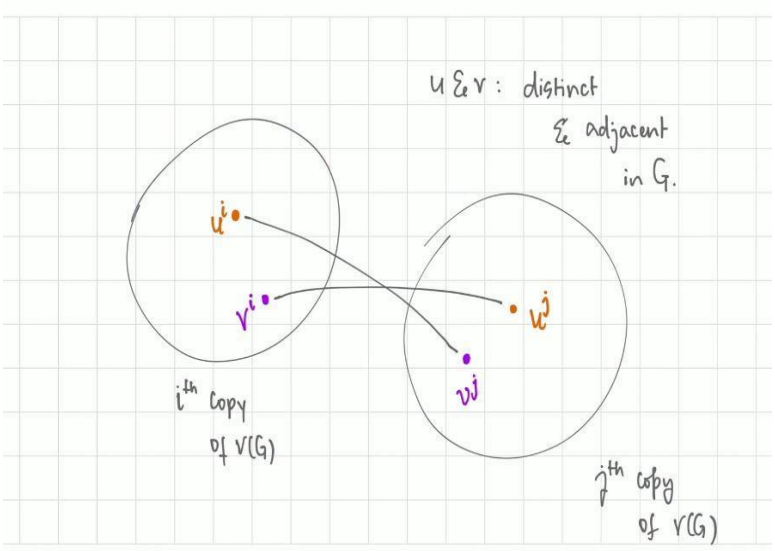
(Refer Slide Time: 14:23)



So, let us say in particular we are working with an instance of clique which is given by G, k where k is the size of the clique that we are looking for. What we are going to do is begin by making k copies of the vertex set of G and we really want to think about what is going on within these copies and what is going on across these copies. Intuitively I want to think of these copies as ultimately the parts of my partition or the colour classes of multi-coloured clique.

So, we do know that we are never going to be indulging in more than one vertex from within any copy. So, I can actually keep the copies empty I do not have to really add any edges inside these copies. You could actually do whatever you like within the copies and it turns out that it would not really matter, but what is really critical is what is going on across the copies.

(Refer Slide Time: 15:14)



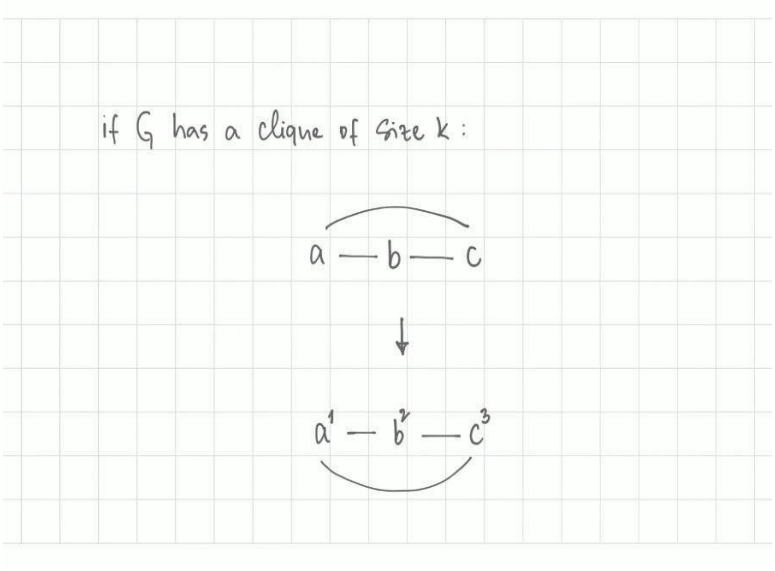
So, let us fix any pair of copies of the graph G we are going to actually do this for every pair of copies. So, we are going to consider all k choose two pairs of the k copies that we have made and here is what we want to do across the copies. If we have an edge between the vertices u and v in the graph G then we are going to look for the ith copy of you and the ith copy of v and the jth copy of here and the jth copy of v.

Here to make sure that they are connected in this way in particular the ith copy of u is connected to the jth copy of wave by an edge and similarly the ith copy of v is connected to the jth copy of u by an edge. Remember that we are working with simple and undirected graphs so this is all that we have to worry about. In particular it is very important that we do not add edges between copies of the same vertex.

Notice that if you did that and you did that for all the k choose two copies then the graph that you generate will always be a yes instance of multi-coloured click irrespective of the status of the instance that you started with. So, that is definitely not going to be a valid reduction. So, just keep that in mind and this actually basically completes the description of our construction.

So, whenever you have an edge make sure that you sort of spread it across all of the copies. Every copy of you gets connected to every copy of v whenever u and v have an edge between them in the original graph G. So, that is what is going on here.

(Refer Slide Time: 16:50)



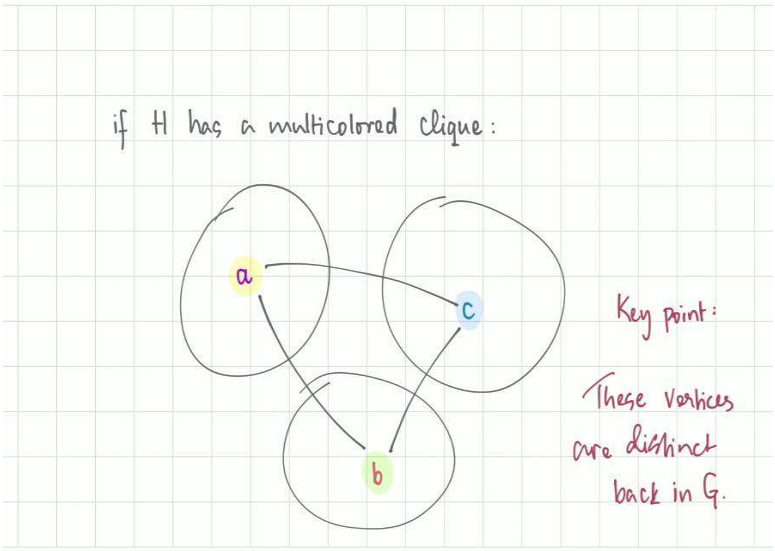
And now let us try and argue the equivalence of the instances that we are working with. So, suppose the instance that we started with which is G has a clique of size k, then to find a multi-coloured clique in the graph that we have constructed all you do is look for the copies of the vertices involved in the clique. So, you could line up the vertices of the clique in whatever order you like.

So, let us say that you have the vertices a b c with k = 3 in this example. Then look for the first copy of a, the second copy of b and the third copy of c. You would have made three copies of the vertex set. So, you can find these copies and because of the construction because of the way we added edges between copies notice that these three vertices will actually form a clique.

In general, if you have a clique on vertices say v 1 v 2 through v k then the multi-coloured clique will essentially consist of the vertices the first copy of v 1 the second copy of v 2 the third copy of v 3. And so, on up to the kth copy of v k and now consider any pair of vertices among these chosen vertices let us say the ith copy of v i and the jth copy of v j. Notice that they are going to have an edge between them because of the construction.

Because of the fact that we know that v i and v j have an edge between them, because they were coming in from a clique in G. So, that covers the forward direction.

(Refer Slide Time: 18:18)



In the reverse direction suppose the graph that we constructed let me call it H actually has a multi-coloured clique, we want to be able to somehow pull this back into a clique of G. We want to say that if the instance that we constructed is a yes instance then the instance based on which this instance was constructed is also a yes instance. So, the natural thing to do of course is to ask ourselves for all the chosen vertices in the multi-coloured clique which vertex were there a copy of back in G.

And once you have the answers to these k questions you have found k vertices back in G and these k vertices must form a clique because again if there was any pair of them that did not have an edge between them, then the corresponding pair for whatever copies you chose would not have an edge between them and H either. So, certainly the chosen vertices will form a clique but one thing to be careful about is to be sure that this is actually a clique on at least k vertices.

So, we have k distinct vertices in edge, but do they lead us back to k distinct vertices in G. What could happen is that you have two vertices which are distinct in G but then they were copies of the same vertex in G. So, when you go back to G then they sort of collapse to the same vertex and lead you to a clique on fewer than k vertices total which would not be nice. So, this is something to watch out for.

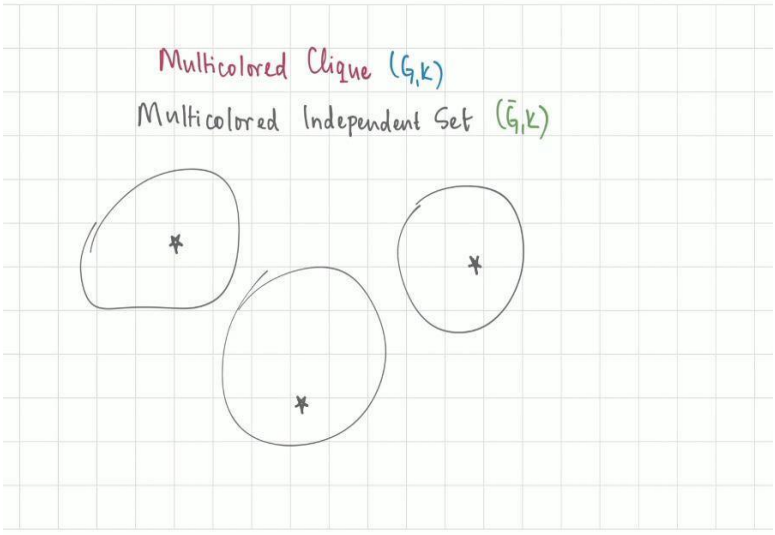
And to make sure that this does not happen we actually ensured in the construction that we never add an edge between two copies of the same vertex. And this is why this aspect of the construction is really crucial, it ensures that whatever vertices you picked in edge actually

correspond to or come from distinct vertices in G. So, that when you pull these vertices back into G you are really looking at k distinct vertices corresponding to a clique therefore of size k.

This completes the argument for the equivalence of the two instances that we are working with. And it is straightforward to verify that this reduction has the other two properties that we desire from it which is that the parameter is preserved. Once again, we are in a situation where k goes to k and that the running time is efficient. And once again, we have a reduction which runs in polynomial time because what we did was to make k copies of the graph G.

And we had to construct the H set between of course every pair of copies that we introduced and all of these are basically polynomial time operations. So, we conclude that multi-coloured clique is in fact as hard as clique itself. And this in fact also shows the hardness of a related problem which is multi-coloured independent set.

(Refer Slide Time: 21:09)



So, multi-coloured independent set is defined pretty much the same way as multi-coloured clique except that we are now looking for an independent set rather than a clique. So, your graph once again is given as a vertex set that has been partitioned into k colour classes and what you are looking for is a colourful independent set which is to say an independent set of size k that picks exactly one vertex from each of these parts or each of these colour classes.

Notice that the hardness of multi-coloured independent set follows quite naturally from the hardness of multi-coloured clique which we have just seen and the way this would work is

similar to how we deduce the hardness of independent set on regular graphs from the hardness of clique on regular graphs. So, in particular if G k is an instance of multi-coloured clique, then G complement k is an instance of multi-coloured independent set with exactly the same partition as G.

And it is straightforward to verify that this works exactly as you would expect the clique in G becomes a clique in G complement and if the clique was multi-coloured to begin with since it is the same set of vertices in the same partition. It is also a multi-coloured independent set in G complement. So, we will see how multi-coloured independent set can be used to show the hardness of dominating set and that is something that is coming up in the next segment and I will see you there.