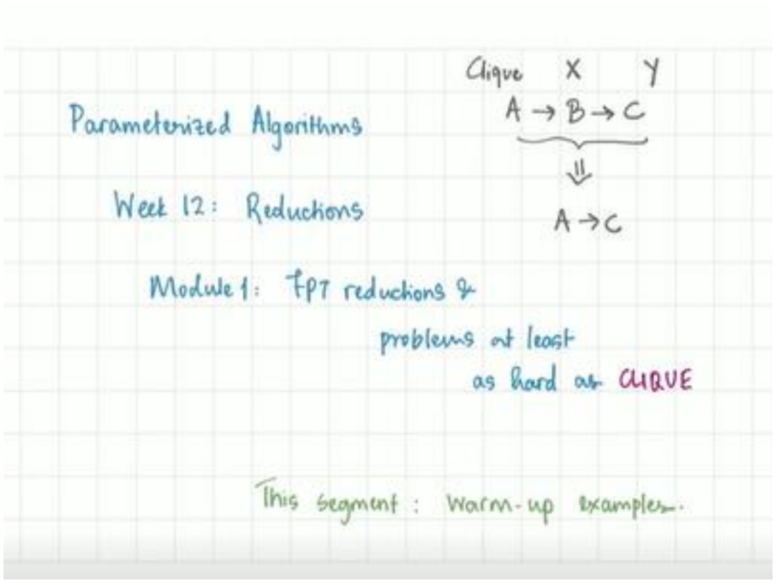
Parameterized Algorithms

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

Lecture – 48
Reduction – Problems as Hard as Clique I (Clique on Regular Graphs)

(Refer Slide Time: 00:11)



Parameterized Algorithms

Week 12: Reductions

Module 1: FPT reductions & problems at least as hard as CLIQUE

$$\begin{array}{ccc} \text{Clique} & X & Y \\ A & \rightarrow B \rightarrow & C \end{array}$$

$$\Downarrow$$

$$A \rightarrow C$$

This segment: warm-up examples.

Welcome back to the second segment of the first module in the 12th week of parameterized algorithms. As you know we are talking about reductions and let us just continue from where we left off last time. We concluded our previous discussion by saying that we are going to assume that clique parameterized by solution size is not an FPT. We have some very good reasons for making such an assumption.

This is something that you could consider as being reasonably widely believed but for now let us just assume that this is the case and let us explore the implications of this assumption. So, if clique parameterized by solution size is not FPT then what does this tell us are there other problems that clique reduces too, that also turn out to be not an FPT assuming that clique is not an FPT.

So, that is kind of the flavour of the discussion that we are going to have now and also into the next couple of segments here. So, we are going to start off with a couple of warm-up examples and then take a look at reductions that are a little more involved. By the way I will keep saying clique without mentioning the parameter explicitly. If it is an unusual parameter then I will try to emphasize it.

But if it is not mentioned then you can assume the parameterization to be the standard one which is the solution size. There is another preliminary remark in order before we get to the actual examples which is to say that although I have written here that we want to look at problems that are at least as hard as clique not every such problem that we discover has to have a direct reduction from clique.

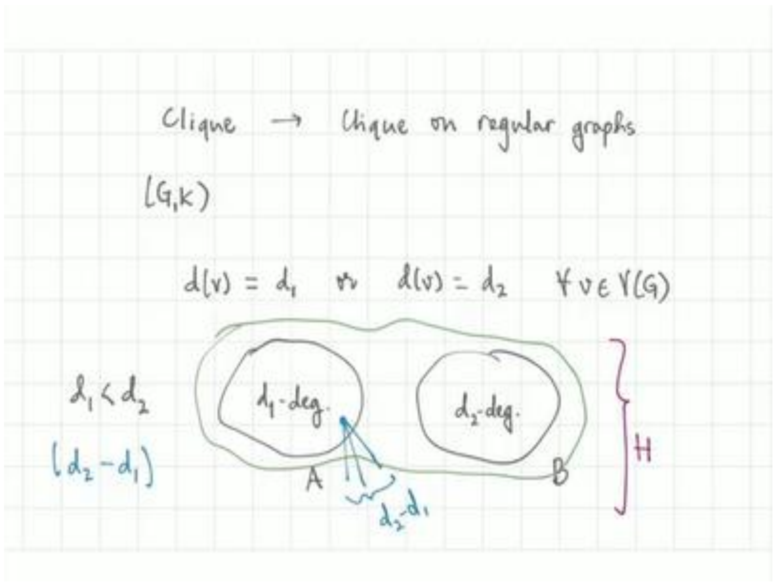
So, we may end up looking at reductions from problems different from clique but these will be problems for which we have already identified a reduction from clique. In other words what I want to say is that reductions are transitive. So, if you have parameterized problems A B and C and you have parameterized reductions that go like this from A to B and then from B to C then notice that this would also imply that you have a parameterized reduction from A to C.

So, in other words suppose you are able to show that clique reduces to some problem x and then x reduces to some problem y. Then y is also as hard as clique simply for the reason that you can chain these two reductions. So, suppose you have a FPT algorithm for y, you can use this FPT algorithm to get an FPTL algorithm for b, for x and then because of the reduction from clique to x you can use the FPT algorithm for x to get a FPT algorithm for clique as well.

It is worth writing down the running times to be sure that I am not pulling a fast one here but really it is a matter of just picking up some pen and paper and convincing yourself that everything works out nicely. So, just like polynomial time reductions parameterized reductions also have this nice chaining transitive property. So, we will be reducing initially from clique and in due course we basically expand our horizon of knowing which problems are at least as hard as clique.

Now if we encounter a new problem that we are trying to work with to show that it is at least as hard as clique it is enough to reduce from any of the problems that we currently know to be as hard as clique. So, I hope that that makes sense with that said let us go on and look at our first example.

(Refer Slide Time: 03:41)


Clique → Clique on regular graphs
(G,k)
$d(v) = d_1$ vs $d(v) = d_2$ $\forall v \in V(G)$
$d_1 < d_2$
$(d_2 - d_1)$
d_1-deg.
d_2-deg.
A
B
$d_2 - d_1$
H

So, our first example involves showing that clique remains hard even if you restrict your input to the class of regular graphs. So, what this means is that even on a restricted class of inputs clique remains as hard as it would have been when there is no such restriction at all. So, notice that this is a strengthening of the hardness that we have assumed for clique and this strengthening is something that we will actually obtain by an explicit reduction.

So, let me just write this out here we want to show that clique reduces to clique on regular graphs and notice that I am not really making any specific commitment to what the regularity is. For instance, I am not trying to claim that clique is hard on say three regular graphs which it would not be hard for any kind of constant regularity like that and you want to think about why for a minute.

The reason for that is because the moment you constrain the degree to be bounded by a constant then the sizes of the cliques that you can hope to find are also bounded by a constant. So, you could simply work things out by guessing a vertex in the solution which is a polynomially expensive guess and after that you could just explore its neighbourhood maybe even in a brute force fashion to find the largest possible clique.

So, we need to be clear about our expectations this is not about showing that clique is hard for graphs that are C regular for some fixed C. But as you will be able to observe once we are done with the reduction the regularity of the instance that we produce is something that will end up depending on the size of the graph that we are starting with and this is perfectly fine.

All that we want to say is that if there was a FPT algorithm that could solve a clique on regular graphs in general.

Then well it can also solve clique in full generality, thanks to this reduction because what this reduction will be able to do is take a general instance of clique and somehow modify it in such a way that you end up with an equivalent instance that happens to have the property of regularity, which is to say that every vertex has the same degree. So, let us think about how we could approach this.

Let us say that we have an instance of clique to begin with which we are going to denote by G, k and if G was a regular graph to begin with that is your lucky day. You do not have to do anything you could simply return the same instance because that is already an instance of clique on a regular graph. The more interesting situation is when G is not a regular graph and just as a warm-up sort of a step.

Let us think about what if every vertex in G had 1 of 2 degrees. So, let us say the degree of v, v being the vertex in G was either d 1 or d 2 and let us say this is true for every vertex in the graph G. So, that means you can essentially partition your graph into two paths, let me call this A and B and these are all d 2 degree vertices here in B and here are all d 1 degree vertices here in A and let me see without loss of generality this is just a notational thing.

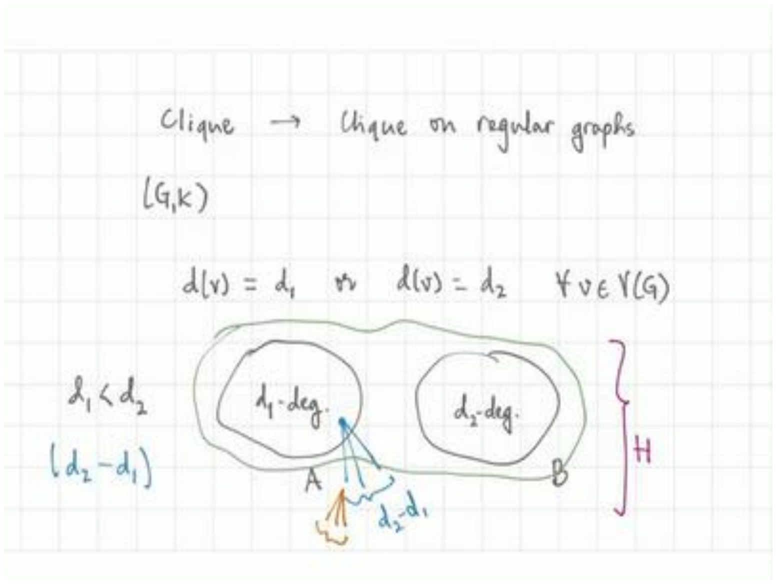
Let us say that d 1 is strictly less than d 2. Given a graph like this; which looks like it is almost regular except that there are two degrees that we have to contend with instead of one. How would you fix this in some sense? And turn it into a regular graph without disturbing the size of the clique maybe or perhaps you could turn it into a graph which has a clique of a different size but you are able to still relate it meaningfully to a clique of size k in the original graph.

So, one thing that seems promising about g is that it already has a lot of d 2 degree vertices. So, if only we could ramp up the degrees of the d 1 degree vertices somehow then and bring it up to d 2 somehow then perhaps, we would be in business. So, one quantity that feels like it might be relevant is this deficit d 2 – d 1. So, what if we took every vertex in A and gave it d 2 - d 1 new neighbours for free.

What this would do is it would ensure that every vertex that came from G now has degree d 2 because well the vertices in B anyway I have degree d 2 and we have not done anything to them and the vertices in A had degree d 1 to begin with and now they have d 2 - d 1 new neighbours. So, if you add up the degrees the number of old neighbours and number of new ones you see that they all have degree d 2 as well.

So, if I call this graph H the one that I just constructed in this way you should be able to show that G has a clique of size k if and only if H has a clique of size k as well. Notice that the newly added vertices have degree one. So, they do not really disturb the scenario in terms of the presence of a clique of size k. So, in fact since this is the first reduction that we are looking at let me just make this a little bit explicit.

(Refer Slide Time: 09:34)


Clique → Clique on regular graphs
(G,k)
d(v) = d₁ or d(v) = d₂ ∀ v ∈ V(G)
d₁ < d₂
(d₂ − d₁)
d₁-deg.
d₂-deg.
A
B
H
d₂-d₁

So, here is the claim. Remember that H is the graph that we obtained from G by adding these pendant vertices. So, the claim was that G has a clique of size k if and only if H has a clique of size k. So, you want to show for all your reductions some sort of a statement like this which establishes the equivalence of the instances that you are working with. The instance that you start with and the instance that your reduction ends up constructing.

So, I will use the standard terminology of forward and reverse directions to show both sides of this implication. So, first in the forward direction we want to say that if G has a clique of size k, then so does H but that is straightforward because H is a super graph of G. So, any clique that sits in G will also be present in edge in the exact same form. Now the other way is to say that well if H has a clique of size k, then so does G.

Now the thing is that H is a super graph of G. So, in general this is not something that will follow blindly, we want to say that the newly added vertices in H cannot huddle together and form a clique which was completely absent in G. But now notice that the vertices that we just added were pendant vertices they were degree one vertices. So, they are not going to participate in any clique which has more than two vertices.

So, if H has a clique of size k and k is 1 or 2 then notice that G anyway has such a clique because we said that d 2 is strictly greater than d 1 which means that you know even if d 1 is 0, d 2 is at least 1, so there is at least an H in G. So, if k is 1, if k is 2 then there is really nothing to prove. On the other hand, if k is 3 or more then the newly added blue vertices in the picture from the previous slide.

These will not be able to participate in any clique of H that has size at least 3. Therefore, any such clique must involve only the old vertices which are also vertices that are present in G. Notice that we did not disturb the internal structure of the vertices in G we did not add any edges or do any monkey business there. So, whatever is there an edge coming from G will be completely something that really came from G and you can relocate it with no trouble.

So, based on this you see that the equivalence goes through which is awesome but what is not so awesome is the fact that we are not done yet even for this specialized kind of a situation where you have vertices only of two distinct degrees. Why are we not done yet? You might have noticed by now that the graph H is not actually a regular graph remember our target is to generate an equivalent instance of clique which happens to be a regular graph.

But H is a lot like G in the sense that it has vertices whose degrees fall into two categories all vertices coming from G have degree d 2 and all the newly introduced vertices actually have degree 1. So, we are not yet at a position where we have generated a regular graph, so we are not quite done yet. So, somehow it seems like these degree one vertices need a degree boost as well.

And somehow, we could think about maybe giving them d - 1, d 2 - 1 new neighbours to fix their degree problem with this just pushes the problem further out it is passing the buck to a whole set of newly added pendant vertices and the problem does not really go away you could just keep doing this and you would go on forever. So, we need a bit of a trick here to see how we can manage amping up the degrees of these degree one vertices that we have just introduced.

But without getting into an endless rabbit hole of having to keep introducing new vertices. So, are you thinking this is a good place to pause and reflect on how you might want to fix this problem yourself? Feel free to use concrete numbers instead of d 1 and d 2 if that helps. One hint that I could give you is to think about maybe possibly working with multiple copies of the original graph G instead of just 1 like we are doing here.

That might help you break out of this endlessly generating new vertices to fix the degree of previously introduced new vertices. So, take a few moments think about this and come back when you are ready. So, welcome back hopefully you had a chance to think through this a little bit, let us continue our discussion here. Remember what we are trying to do is essentially fix the degrees of the blue vertices.

So, we introduced these blue vertices to fix the degrees of the vertices from G but now these blue vertices themselves end up being low degree vertices and they need their degrees also increased to d 2 in at least this example here. And we said that if we just introduce new vertices just like the blue vertices then we might just get stuck in an endless loop of having to fix the degrees of the newly introduced vertices and so on.

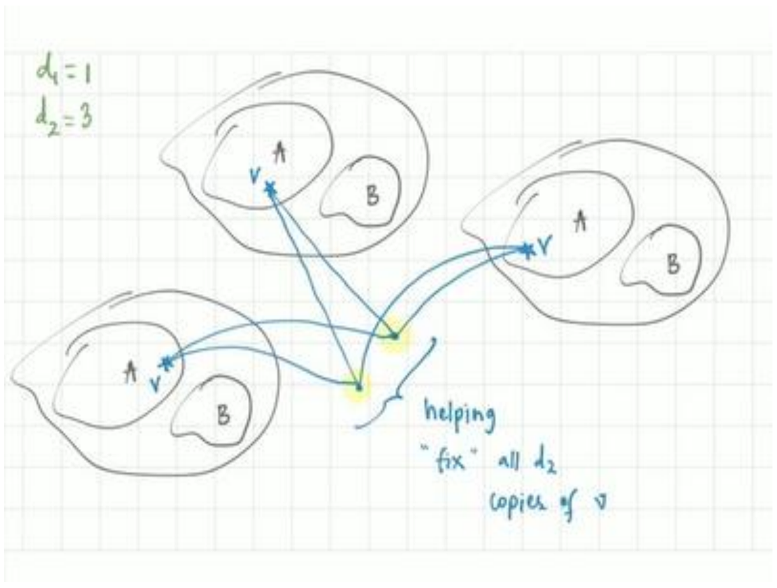
But at the same time, it is clear that we somehow do need more vertices in this picture because if we try to supply the additional degree that the blue vertices need just from among the vertices in this picture and you can try to do that there are various things that you could consider doing. But you will find that first of all it is hard to do this in a systematic fashion. The second thing is that you might end up by adding more edges accidentally creating larger cliques that you did not intend to.

And that may become hard to keep track of when it comes to equivalence. So, it not saying it is impossible but it definitely seems like it would be a little messy or tricky to get it just right. So, it seems like the simplest thing would be to introduce some more artificial vertices to handle the degrees of the blue vertices. And one hint that we shared before taking a break was to see if we could consider making copies of the graph G.

Now why would we want to do that. Well, think of the blue vertices as fixer vertices these vertices are helping you fix the degrees of the needy vertices in the graph G. So, what if we created more needy vertices? Then we could use hopefully the same blue vertices in the same way to fix the requirements of these newly created needy vertices as well. So, you will essentially kill two birds with one stone.

Because you are creating these new needy vertices but you are having their degree needs met by these existing blue vertices. But at the same time these blue vertices are also experiencing an increase in their degree as they go through the process of helping out these newly introduced needy vertices. So, let us see how that would actually play out in the context of an example.

(Refer Slide Time: 17:14)



So, let us make this a little more concrete by saying that d 1 = 1 and d 2 = 3. So, notice that if you just focus on one of the copies of G here then this is a familiar picture, we have a vertex v which is in the A group remember the a group is all d 1 degree vertices. In this case the A group contains all the degree 1 vertices in G and a vertex in A is going to have 2 blue neighbours. So, that its overall degree becomes 3.

And now what we have done is we have created 3 copies of the graph G and when you look at the vertex v the vertex that is been highlighted from the first copy. You look at the same vertex in the other copies. We are going to have these copies of we use the same helper vertices to get their degrees fixed as well. In the meantime, notice what is happening to the helper vertices?

The helper vertices are not only helping the newly created needy vertices but they are also helping themselves in the process. So, now notice that the 2 blue vertices actually also have degree 3 because they are helping out each of them is helping out one vertex in each of the three copies of the graph G. So, from the point of view of any one copy these are pendant neighbours degree 1, neighbours is before.

But when you take all the copies considered together the blue vertices end up having exactly the degree that you need. So, just to recap here is the construction. At least for the situation when the original graph has two types of vertices, vertices of degree d 1 and vertices of degree d 2 and once again d 1 is strictly less than d 2. So, what we are going to do is make d 2 copies of the graph G.

And now let us fix our attention on any vertex which has degree d 1 in the graph G and let us look at all the d 2 copies of this vertex. What we are going to do is introduce d 2 – d 1 new vertices. And have all of them become adjacent to all the copies of this vertex of degree 1. What happens is that every vertex every copy of the vertex that had degree 1 in the graph G now gets d 2 – d 1 new neighbours.

So, all of these copies have their degrees fixed their overall degree is now d 2 but now all the newly introduced vertices have one neighbour in each of the d 2 copies of the graph G, so their overall degree is also d 2. Now remember that although this picture shows it to you for a fixed vertex v for clarity. You are actually going to repeat this process for every single vertex in the collection A.

So, every d 1 degree vertex in the graph G is going to get the same treatment they are going to find d 2 - d 1 new neighbours waiting for them in the reduced graph H. So, this completes the description of the construction. Notice that this edge now for real is a regular graph its regularity is d 2 every vertex has d 2 neighbours in this graph. And now what we need to do is revisit the argument for equivalence.

I should pause here for a moment to say that this is a fairly natural thing that happens when you try to come up with your own reductions it is rarely a one-shot process. So, you try to do something you play around and it may or may not work. So, you go back and modify what you were trying and every time you modify things it is important to make sure that you check that what you are working with is indeed a valid parameterized reduction.

And if it is not then that is going to be a reason to go back and modify things further. So, what happened for us here was that although our first attempt was okay in terms of equivalence. It did not quite get us where we wanted in terms of the structure of the reduced graph. So, we had to come back and tweak things a bit and now we want to make sure that our argument for equivalence still goes through.

And notice that it is actually going to be a very similar argument. The spirit of the argument is still that the blue vertices do not interfere in any non-trivial cliques. They cannot really be a part of any clique of size three or more. And the reason for that is that they do not really have more than one neighbour into any copy of the graph G and you really cannot have a clique that involves vertices from more than one copy of G because there are no edges that go across copies of G.

The copies of G are completely disjoined in the reduced graph H. So, just like we did before we will not have to worry about the cases when k = 0, 1 or 2 because they can be handled trivially. And if k is at least 3 then any clique in H which has at least 3 vertices must be contained completely in one of the copies of G and that is the crucial observation. Based on this you can pull back a copy of this clique in your original instance.

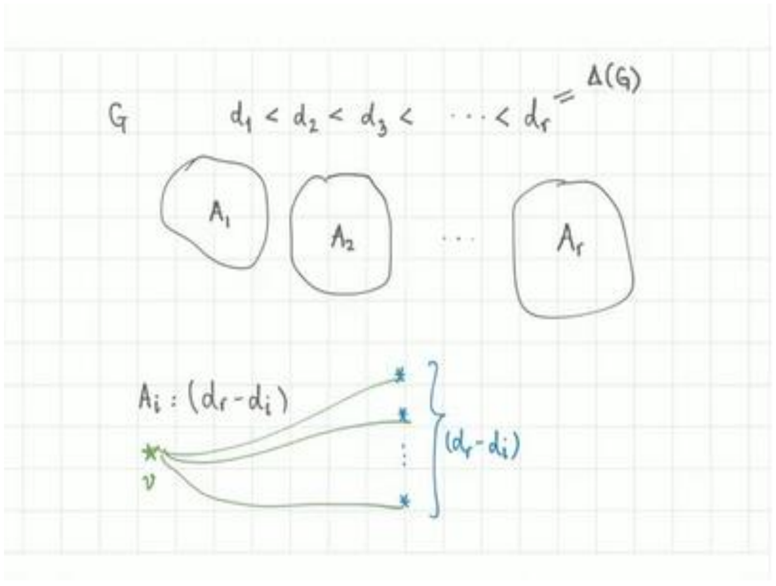
Because the way the copies of G are constructed in edge is that it is simply exact copies of G. So, whatever you find an edge you will find when you go back to G as well. So, just to summarize in the forward direction we want to say that if G has a clique on k vertices so does H and this is true because H is simply a super graph of G. And in the reverse direction we want to say that if H has a clique on at least k vertices.

Then so does G and that follows from the argument that we just made. So, I will be finally done. Well, in the context of this restricted special case I would say yes. Notice that we have just proved the equivalence and the other two properties that we need from a parameterized reduction which are that the parameter is preserved and the running time is FPT are easy to verify.

In fact, the parameter does not even change in this particular reduction k goes to k and the running time here is in fact even polynomial. So, we have nothing to worry about this is a perfectly valid parameterized reduction. The only catch is that we did this whole discussion in the setting of graphs which have vertices of degree either d 1 or d 2. But that is not what general graphs look like our original intention was to work with general graphs.

And say that clique on general graphs reduces to clique on regular graphs. And this was something of a special case that we did as a warm up.

(Refer Slide Time: 24:13)



So, let us go ahead and upgrade the reduction so that it works for a general graph. And it turns out that what you have to do is really not all that different. So, let us say that we are looking at a graph G and now we are not going to make any additional assumptions about this graph and let us say that we just make a list of all the degrees that we encounter. So, just go through the vertex set of the graph and make a list of all the degrees that you see.

And let us say that the degrees you see are d 1 d 2 d 3 and so on up to the r. And now r need not be anything special certainly. We are not going to assume that it is a constant or anything like this. I just make a list of all the degrees that you see and let us group the vertices of G by their degree. So, I am going to say A 1 consists of all the vertices of G that have degree d 1, A 2 consists of all the vertices that have degree d 2 and so on, all the way up to A r.

Since these are all the degrees that we saw in the graph G this is a partition of the entire vertex set of G into r many paths. Now without loss of generality let me say that these degrees were identified well the degrees are distinct. So, let us say that they were identified in increasing order which means that in particular d r is actually the maximum degree of the graph G. So, what we are going to try and do just like we did before is that we are going to try and convert this into a d r regular graph.

So, the vertices in A r we do not have to worry about them because they already have the degree that we want them to have but everybody else has a bit of a deficit. So, in particular if you look at all the vertices in A i they are falling short by d r - d i. So, what we want to do is for any vertex in A i we want to introduce a group of deficits many vertices. So, this is d r - d i many vertices and we are going to introduce of course a separate group for each vertex in A i.

So, for particular vertex and A i is going to get these many new neighbours and that is going to fix the degree of this particular vertex and A i. As I said we repeat this process for every vertex in A i and we are going to do this for all i between 1 and r - 1. So, notice that if r was 2 this is exactly what we started to do in this special case that we were discussing. Now like before we have the situation where we have generated a large number of degree 1 blue vertices.

And we need their degrees to be fixed and we do the exact same thing that we did before which is to say that we create d r many copies of the graph G and we have the same blue vertices help out the same copies of the original vertices. So, as before if these blue vertices that you see on your screen here were helping out a particular vertex v in A i then they will continue to help all copies of v across, all the d r mini copies of G that we generate.

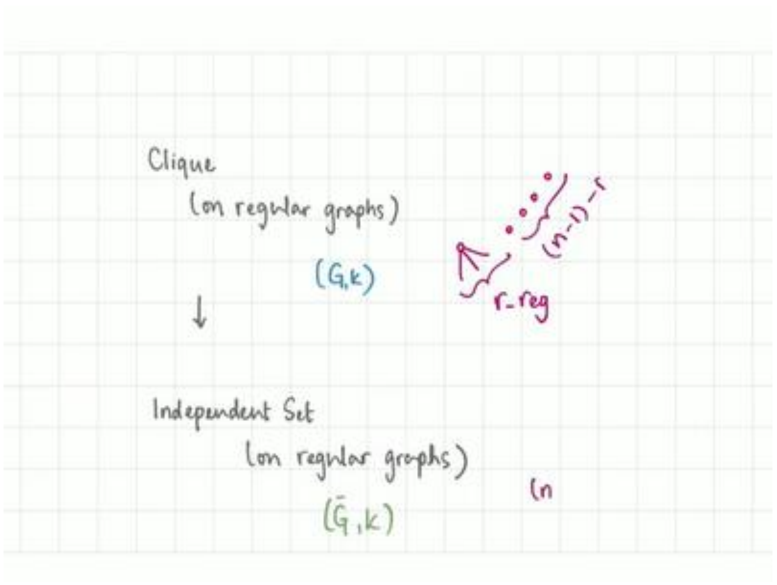
This way all the blue vertices will also end up with a degree of d r and since every copy of every vertex got the help that they needed from these extra blue vertices. They also end up with a total degree of d r. In particular any vertex in A i will have a total degree of d r - d i + d i, which is its original degree which really adds up to d r. So, that is it, that is the entire construction and the argument for correctness is pretty much the same as before.

Once again, the main thing to note is that the newly introduced blue vertices really have no stakes in any large cliques in the reduced graph H and you can argue the equivalence the same as we did before. The other aspects are quite straightforward to see k goes to k once again. And you should be able to easily convince yourself that the whole construction only requires polynomial time.

So, that is our first example of a parameterized reduction which demonstrates that clique is as hard as it originally was even if you restricted your input to regular graphs. So, one of the reasons this is a good thing to know is because sometimes for certain problems it is easier if you were to reduce from clique on regular graphs as opposed to clique on general graphs. We will sort of see an example of that playing out although not directly with clique.

So, what we are going to be able to use in the next reduction that we see is the fact that independent set is as hard as clique even when restricted to regular graphs.

(Refer Slide Time: 29:26)



The reason for that is because clique on regular graphs reduces to independent set on regular graphs. And what is the construction here? Well, the construction here is the same construction that you might be familiar with that connects clique and independent set in the classical world. So, notice that a graph G has a clique of size k if and only if the graph something has an independent set of size k.

What is this graph that is not written on the screen yet? Let us take a moment and maybe you can confirm with me that this graph is actually just the complement of the original graph G.

So, if you replaced every edge in G with a non-edge and every non-engine G with an edge then if you just focus your attention on the clique in G just shine a spotlight on the vertices of any clique and then go through the complementation process.

You will just see all of the edges of the click disappear into the void and what you will be left with is an independent set in G complement. Conversely if you have an independent set in G complement and you go from G prime to G, then you will see that the independent set actually ends up manifesting into a clique in the original graph G. So, observe that this is in fact a valid parameterized reduction in contrast with the other one that we saw between vertex cover and independent set.

There the graph remained the same but the parameter shifted from k to n - k and that was a deal breaker. But here it is the graph that is getting complemented the parameter stays the same. And of course, this whole thing is a polynomial time construction. So, this is in fact a valid parameterized reduction and also notices that this construction preserves the regularity although the regularity may not be the same.

So, for example if you started with a graph that was r regular what would you end up with. Well, if every vertex has degree r it means that it has r neighbours and n - 1 - r non-neighbours. I am going to assume that the graph is simple without loss of generality. And when you complement the graph therefore you are going to end up with something like n - 1 - r regular graph because this applies to every vertex.

So, the regularity is preserved and you end up having exactly what we claimed which is that independent set on regular graphs is as hard as clique on general graphs because now you can change the reductions as we said before. Next, we will use the hardness of independent set on regular graphs to show the hardness of partial vertex cover, so stay tuned for that. It is coming up in the next segment.