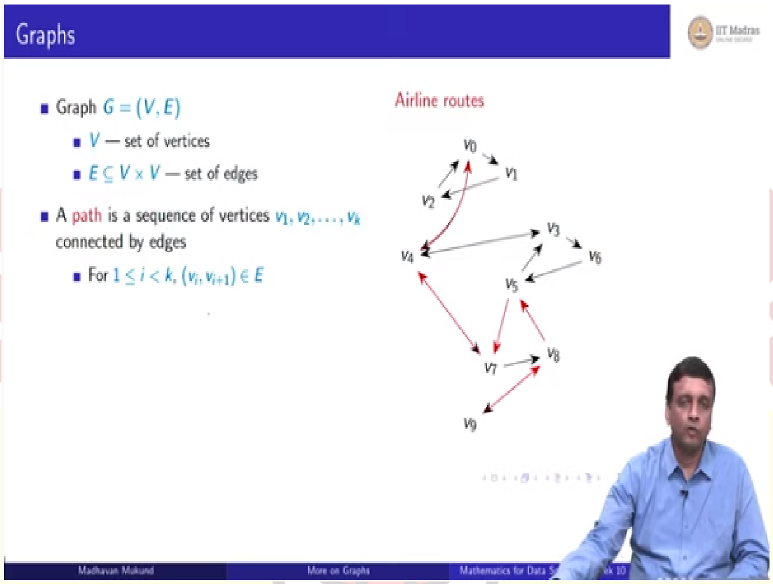
IIT Madras

ONLINE DEGREE

Mathematics for Data Science
Professor Madhavan Mukund
Indian Institute of Technology, Madras
Lecture 60
Some General Graph Problems

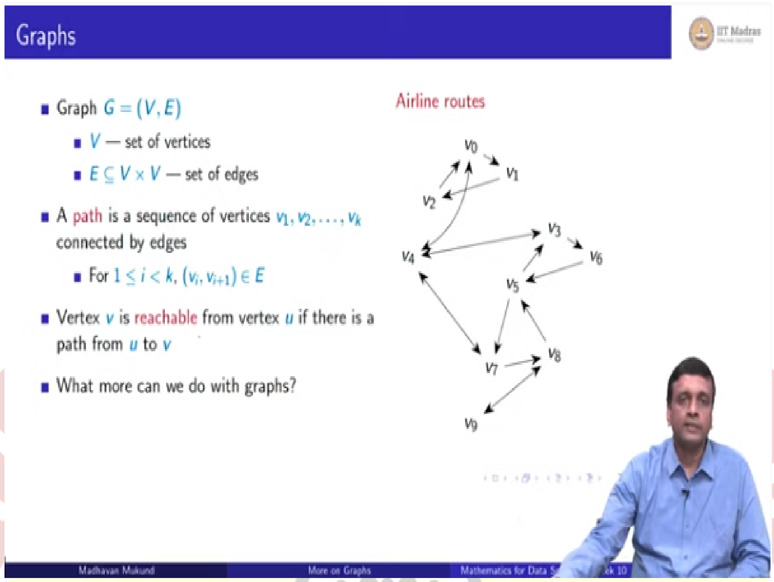
(Refer Slide Time: 0:16)



So, in our first lecture we introduced the concept of a graph, so we said that a graph consists of a set of vertices and a set of edges, so the edges are just pairs of vertices, so edge relation is a subset of v×v. So, for example, we had this directed graph on the right which represents airline roots.

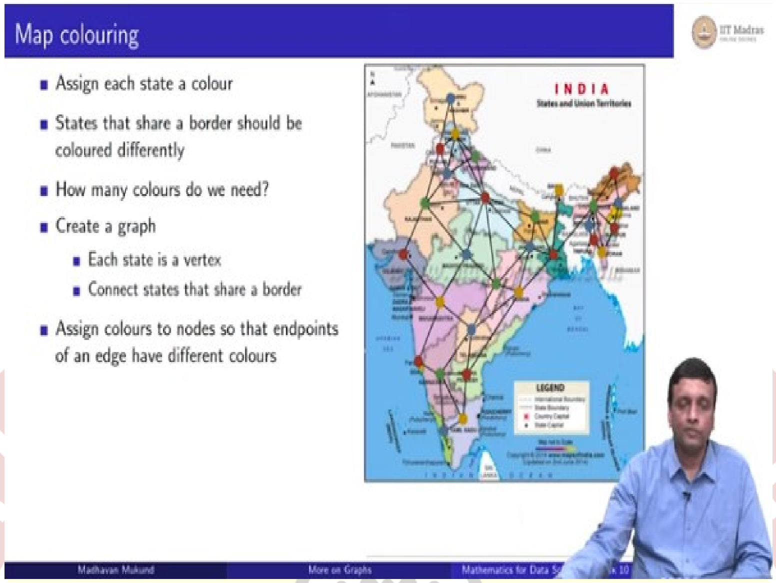
And then we said that a path in a graph is a sequence of edges leading from one vertex to another vertex without any vertex being repeated in between. So, here we see a path from v_9 to v_0, so each edge must be an extension of the previous edge, so we go from v_9 to v_8, then we go from v_8 to v_5 and so on, right.

(Refer Slide Time: 0:55)



And then we talked about reachability saying that we might want to ask whether a vertex u, starting from a vertex u we can reach a vertex v by finding a path. So, at this point the only problem that we have really looked at in graphs is reachability and this is really one problem which we will spend some time on but before we get into more details about how to calculate reachability in a graph. I would like to show you that graphs have much more interesting problems than just reachability associated to them. So, having a graph representation of a problem allows you to deal with very many different scenarios.

(Refer Slide Time: 1:35)



So, let us start with a problem which does not appear to be connected to graphs at all, that is how to color a map. So, typically when you see a political map of the world or of a country like India you will see that each political unit has a different color, but not all have the same have different colors for instance you might find a color that is repeated like you can see in this map for instance that some colors like light blue and green are repeated in different places.

So, the rule is that normally when you color some state in the map or a country in a map it must have a different color from all the countries or states which share a border, so there is no confusion at the border because one side is colored one way and the other side is colored the other way.

So, this is the rule for map coloring, so one question that you might ask and at the moment it seems like an ideal mathematical question is how many colors do we need? Now, clearly if I have a certain number of states, I can use a different color for every state, so I have an upper bound, right, if I have say 27 states then I need 27 colors, if I have 27 colors each of them will get a different color there is no question of two states sharing a boundary having the same color because no two states are the same color.

But maybe I do not need 27 colors, can I do better than that? So, here is where a graph comes in. So, how do we create a graph to represent this problem and what is the problem that we are trying to solve on the graph. So, to create a graph what we do is create these vertices and what are our vertices in this case? The vertices are the states, okay or if it is a map of the world or the countries.

Now, what is the edge relation? The edge relation is going to represent when two states share a border, when they are neighbors and must be colored differently. So, we connect all states which share a border and then we get these black lines connecting these black dots. So, for every state there is a black dot and every state is connected to its neighboring state black dots, this is our underlying graph.

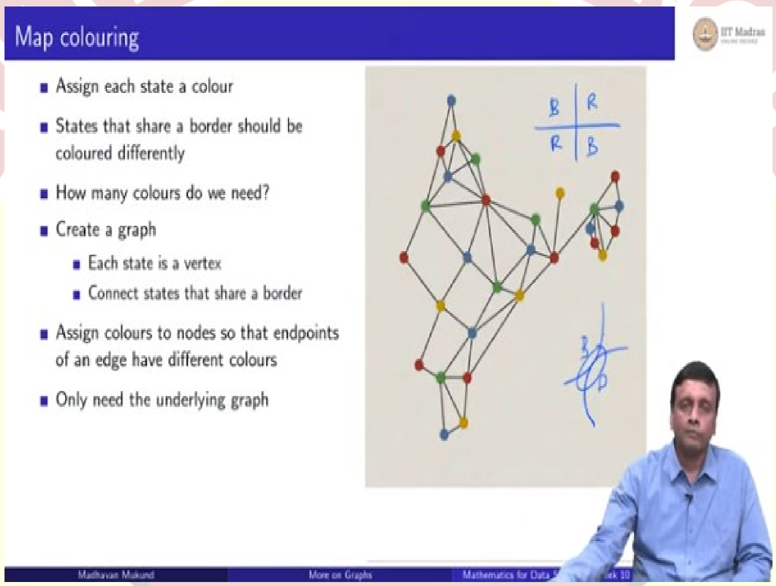
Now, our goal is to associate a color with every state on the map which is the same as associating a color with every black dot in this graph. So, we start maybe by assigning a color in this case red to Uttar Pradesh, now the rule for map coloring tells us that if Uttar Pradesh is red then all the neighboring states must be a different color other than red. So, anything which is connected to Uttar Pradesh in the graph, any node that is connected to Uttar Pradesh must have a different color.

So, we start with this color for Uttar Pradesh and then we can start coloring its neighbors, so for instance we might choose a different color green for Uttarakhand and we might choose blue in this case for Haryana. So, proceeding in this way we go to the neighbors of these and we use a different color but notice now that once we go from Haryana to Rajasthan, we can reuse the color green because green has been used for Uttarakhand and Uttarakhand and Haryana and Rajasthan do not share a border, so there is no confusion, so we could reuse a color if it is not being used for any of the neighboring states.

So, we proceed in this way so we could again now reuse red for Punjab because Punjab is not connected to Uttar Pradesh, remember Uttar Pradesh was originally red, so we could not take any neighbor of Uttar Pradesh but since Uttar Pradesh was not connected to Punjab we do not have to worry, we can reuse it but finally now when we come to Himachal, we have a problem because Himachal is now surrounded by three neighbors which have already been assigned three different colors, Punjab is red, Haryana is blue and Uttarakhand is green, so we have to choose a fourth color, say yellow for this.

So, we keep proceeding in this way and we can now whenever we need a new color, we use it, whenever we can reuse a color, we reuse it. So, we can come up with a more say you know less expensive coloring in terms of number of colors for this, by coloring these nodes in this way.

(Refer Slide Time: 5:51)



Now, notice that we do not really need that map anymore once we have constructed this graph which describes the connectivity of the states in terms of which ones share a border we could always as well start coloring the graph using this rule, that if I color a node with one color, I cannot color any of the neighboring nodes, any edges connected to it must lead to different colors.

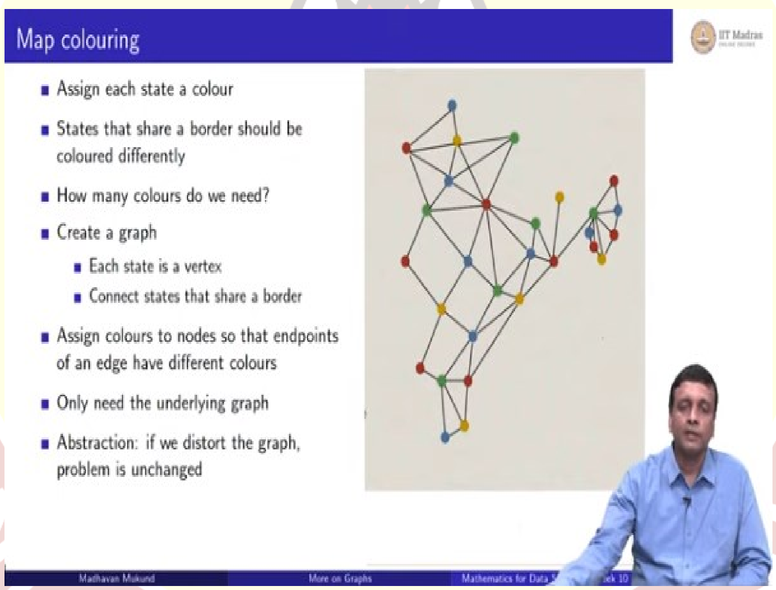
Now, here you can see one advantage which is when we are working with the physical map we have to stare at the borders and it depends on how well the map has been drawn to be able to distinguish because sometimes we have borders which meet at a corner and technically across a corner this coloring rule does not apply.

For example, if four states which happens actually in the United States, if four states meet like this. Then I can actually use the same color, I can use say red, red, blue, blue this is legal as far as map coloring goes. So, if two states or two countries touch only at one point then they are not considered to be sharing a body.

So, this might depend on how the border is drawn maybe that is the picture that you see but actually if you blow it up it actually looks like this, and then I have a problem, because now I cannot use blue, blue because there is a segment of border which is common to these two states.

But once I have transferred this information to the graph then I do not have this confusion anymore and in fact I can take this graph and I can even distort it, right. So, this is the original graph somewhat faithfully representing the geometry of the underlying country.

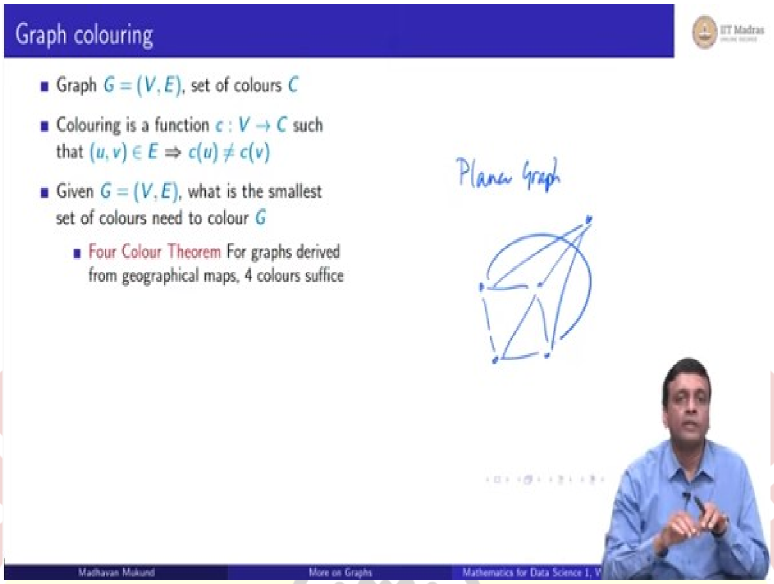
(Refer Slide Time: 7:16)



And now I can take some of these nodes which are bunched up and move them far apart so that I can draw my coloring better. So, this is one advantage of moving to the graph which is that the graph abstractly represents the relevant information, so we can now work with whatever format of that information is convenient in the graph without worrying about the original format in which the information came.

So, here in this case the geography or the geometry of the actual state boundaries is no longer important, we just need the connectivity saying which states are neighbors of which state.

(Refer Slide Time: 7:50)



So, abstractly we have transformed our map coloring problem to what is called a graph coloring problem, so in a graph coloring problem we have a graph which consists of some vertices and edges as before and separately we have a set of colors and we want to do a coloring, so what is the coloring?

A coloring is a function which assigns to every vertex a color from the set C and the rule is that if I have a pair of vertices connected by an edge their end points should have different colors. So, u, v is an edge, the color of u should be different from the color of v, this is what graph coloring demands.

And the question that we were asking is given a set a particular graph, if I do not fix the set of colors in advance, like we were doing there when we did our example, we started with one color red and then we were forced to we chose another color green and then we were forced to we choose another color blue, then when we were forced to we chose a fourth color yellow and so on.

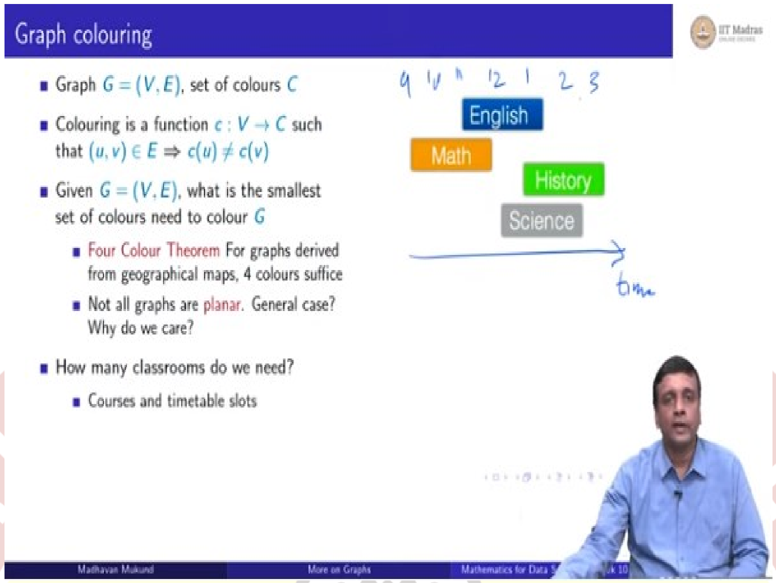
So, if I add colors only as I need them how many colors will I need, right, what is the minimum number of colors I need for this specific graph? So, it turns out that this problem has actually been well studied for these graphs which come from maps. So, there is a very well known theorem which is very hard to prove called the Four Color Theorem which says that for graphs which are derived in the way we showed from geographical maps, four colors suffice. So, technically these are what are called planar graphs. So, planar graph is something where if I draw the edges, right, they will not cross, so this is a planar graph.

If I draw this edge for instance, then these two edges are crossing but this is not necessarily a problem because I can actually take this edge is crossing, right and I can actually draw it around. So, this is still a planar graph but now if I try to connect for instance some a third thing which is outside here and I try to connect it across these then I will have a problem, right. So, some graphs cannot be drawn on a sheet of paper without edges crossing and these are called non-planar graphs.

Now, it turns out when you have a map laid out and you start connecting them obviously the map cannot, you cannot have share a border with something which is far away, so therefore, a map will always be a planar graph, right. So, not all graphs are planar, so the question that graph coloring asks is the general case, right.

Now, from our perspective the question is why do we care, I mean map coloring itself seems to be maybe a particularly specialized application where we want to look at this coloring problem and translate it to graphs but why should we care in general, how many colors we need for a graph? So, it turns out that graph coloring actually is very useful in a number of different cases.

(Refer Slide Time: 10:39)

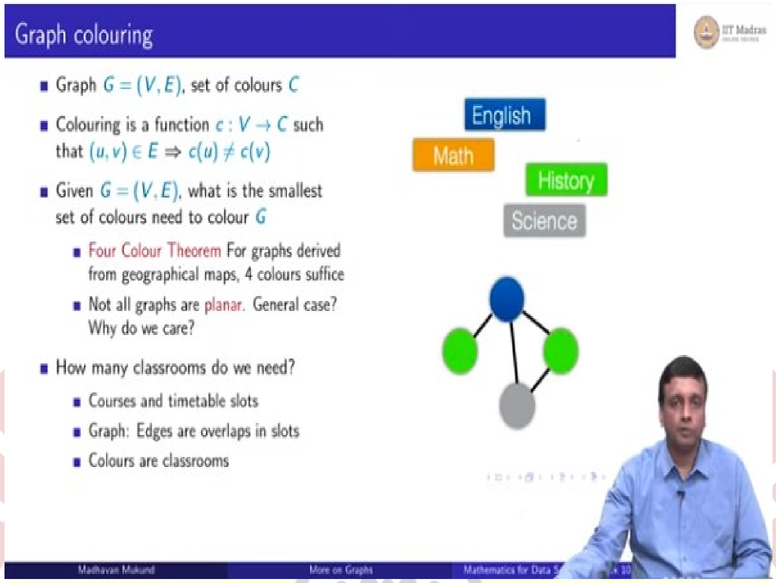


So, one typical case is in classroom scheduling, so supposing we are running a school and we need to determine how many classrooms we need in order to run our classes. Now, this depends on the time table, right. So, we have some time tables and we have some courses and let us assume that this is a graph, not a chart representing, so this is the time of the day, right. So, this might be like 9 o'clock, 10 o'clock, 11 o'clock, 12 o'clock, 1 pm, 2 pm, 3 pm. So, across the day we have these different lecture slots and we have different courses which occupy different slots in our time table.

Now, clearly if maths is running from 9 to 12 and English starts at 11, then English must be in a different classroom from maths. Similarly, if history is running from 1 to 3 and science already started at 12 and goes on till 2, history and science cannot be in the same classroom. So, if we have overlapping slots then the corresponding classes need different classrooms.

So, the question is what is the minimum number of classrooms I need in order to run all these classes without having any scheduling conflicts? Now, like before we could assign a separate classroom, we could have a separate English classroom, we can have a separate math classroom, we can have a separate history classroom, a separate science classroom and the problem is solved but we want to optimize, we do not need to want to necessarily allocate a separate class, for every classroom for every course.

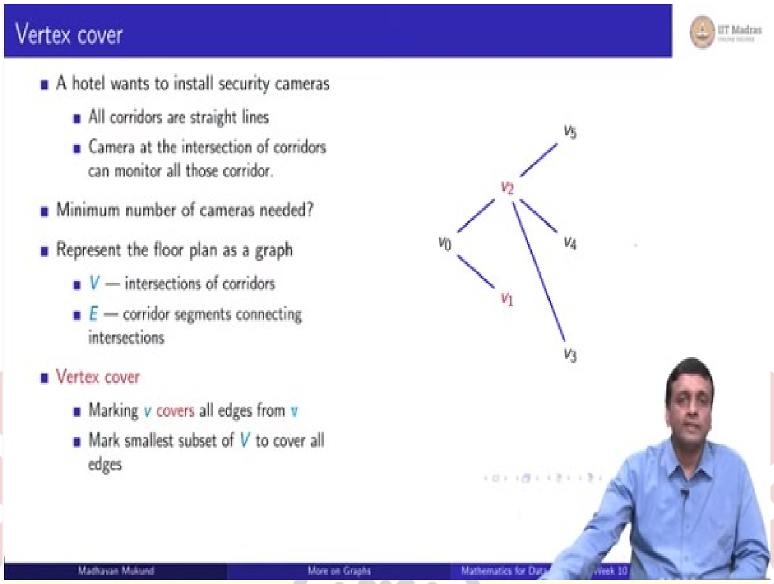
(Refer Slide Time: 12:01)



So, as before now we can draw a graph, so in this graph we have nodes which are our courses and now the edge relation represents overlaps. So, an overlap says that these two courses share a time slot and therefore, they cannot be both scheduled in the same classroom. So, for us now colors are classrooms, so here is a situation that we have four different colors assigned to these four different nodes saying that we are going to have every class running in a different classroom.

But now we observe that maths and history do not overlap. So, since maths and history do not overlap, using graph coloring we can see that the same color can be assigned to both the maths node and the history node. So, in this way many scheduling problems can be actually converted to graph coloring problems.

(Refer Slide Time: 12:50)



Now, here is another problem. Supposing a hotel wants to install security cameras, so they want to put off cameras in the corridors of a hotel and as you know in many hotels corridors are very neatly aligned, they are all straight lines but there might be a maze of corridors which meets at different corners. So, let us assume that if you put a security camera at a particular corner, it can monitor every corridor that meets at that intersection.

So, now the question is what is the minimum number of cameras that you need to monitor all the corridors on the floor? So, once again we can go to graph theory, so we can represent the floor plan of the hotel for that floor as a graph, so here the points of interest are these intersections because clearly it is to our advantage to put a camera at an intersection because if I put it at an intersection that camera can monitor multiple corridors and I want to cover all the corridors.

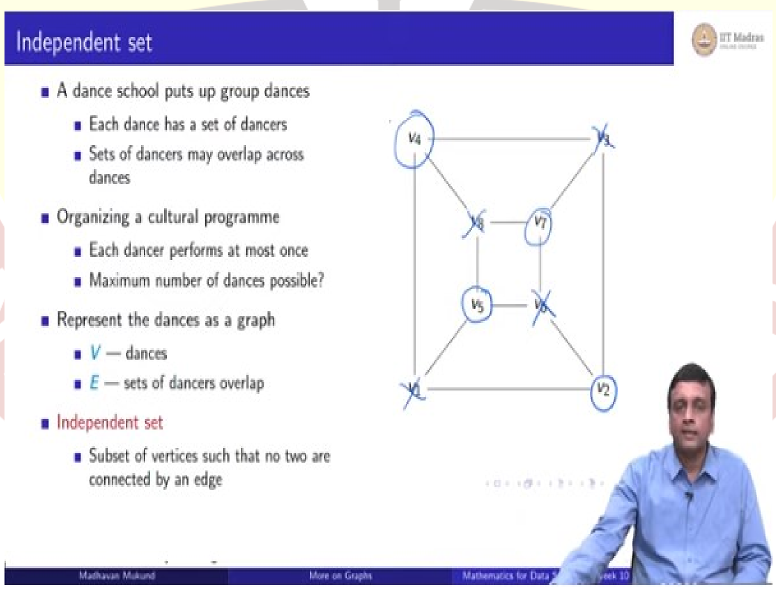
So, I put vertices and intersections and my edges now connect these intersections, there are segments of the corridor which run from one intersection to another intersection. And now my question is one of called, something called a vertex cover which is that I want to choose a subset of these intersections, such that if I put a camera at this subset then every corridor segment is covered. So, in graph theory this is called a vertex cover question, so vertex cover basically says that I want to choose a subset of vertices such that if I choose that subset then those vertices cover all the edges in my graph.

So, let us look at this graph on the right, so it has 6 vertices named v_0 to v_5, so maybe this represents the corridors in our hotel, so maybe I choose to put a camera at v_2. So, if I choose to put a camera at v_2, then this covers these 4 corridor segments but it does not cover the segment from v_0 to v_1, so I have to put one more camera, I can choose to put it at v_0 or at v_1, so let me say I put it at v_1 and now I have a vertex cover.

So, my vertex cover is v_1 and v_2, if I choose v_1 and v_2 as my locations for my cameras it covers all the corridors, so this is a problem in graph theory which you can solve independent of the source. So, this is one motivation but you can come up with other situations where the solution that you require is a vertex cover.

A similar situation could be for instance if you want to locate ambulances at intersections, so that they can reach all localities fast, so if you want to cover all localities with ambulances, where should you place the ambulances in your city map so that every locality is served within a reasonable amount of time.

(Refer Slide Time: 15:33)



Here is yet another problem. So, supposing there is a school, a famous school of dance and they are going to put up a show which consists of a number of group dances, so in a group dance obviously there are a number of dancers who participate but the school as a whole has over a period of time rehearsed many such dances and some dancers take part in more than one dance. So, if I look at all the dances that the school could possibly put up there are overlaps between the dances in terms of which dancers are required.

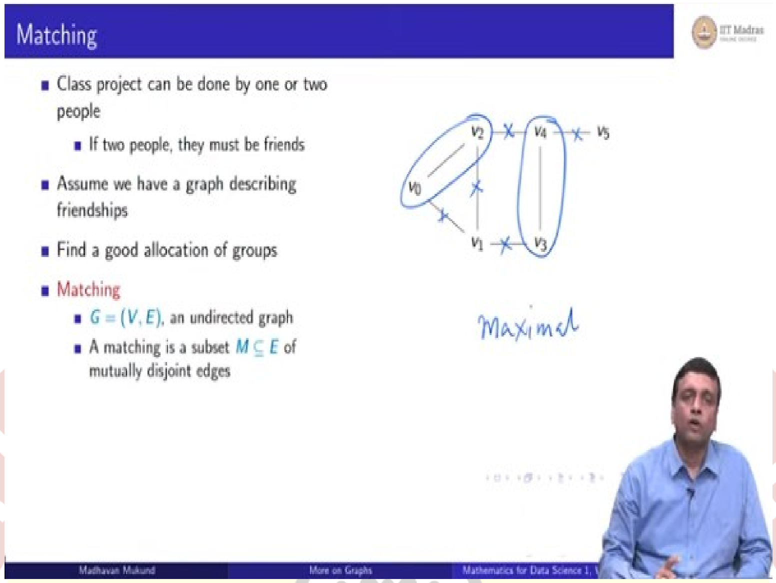
So, now the problem is to organize a cultural program and in this cultural program because of costume changes and other constraints we would like each dancer to participate in at most one dance, so we do not want a dancer to take part in a dance and then to go back have to change and come back and take part in another dance with a different costume. So, given that we have some information about the dances and which dancers are required for each dance, can we come up with a large set of dancer which we can put in this cultural program, so that no dancer has to dance twice during that program.

So, in this particular case the graph will consist of vertices which represent the dances and now an edge will represent an overlap in terms of the dance group between two dances, so if two dances share a common dancer, then we cannot put both dances in the program because one dancer would have to dance in both dances and that is not allowed by the rules that we have just stated. So, this is our graph now and now we want to find what is called an independent set, so we want to find a set of vertices such that there is no edge between any two vertices in that set.

So, we want to pick a set of dances so that no two dances in the set that we have chosen for the program share a dancer and therefore, require a dancer to dance twice. So, here is an example with 8 nodes, so now supposing we pick v_2 to be in our independent set, right, then because I cannot use anything which is connected to it, it means that v_6 cannot be part of my independent set, I can no longer use dance in v_6, I cannot use the dance in v_3, I cannot use the dance in v_1. So, what can I do? Maybe I can choose v_5, now v_5 already rules out v_6 and v_1 which I have already gone but it also rules out v8 but I can now do v7 which has no further constraints and finally I can do v_4.

So, in this particular scenario I could choose actually four of these vertices such that there is no edge between any of them, so this is what is called an independent set, right. So, the independent set here, one independent set is v_2, v_5, v_7, v_4 of course, there is a symmetric independent set I can treat the black edges also, black vertices also as an independent set and so on.

(Refer Slide Time: 18:31)



So, final example of this kinds of problems that we can do with graphs, let us look at a problem which is called matching. So, supposing we are assigning class projects and the teacher allows the project to be done by either one person, student individually or by two people but there is a constraint that if two people participate in a project then they must be friends because if they do not get along with each other then the project would not get done.

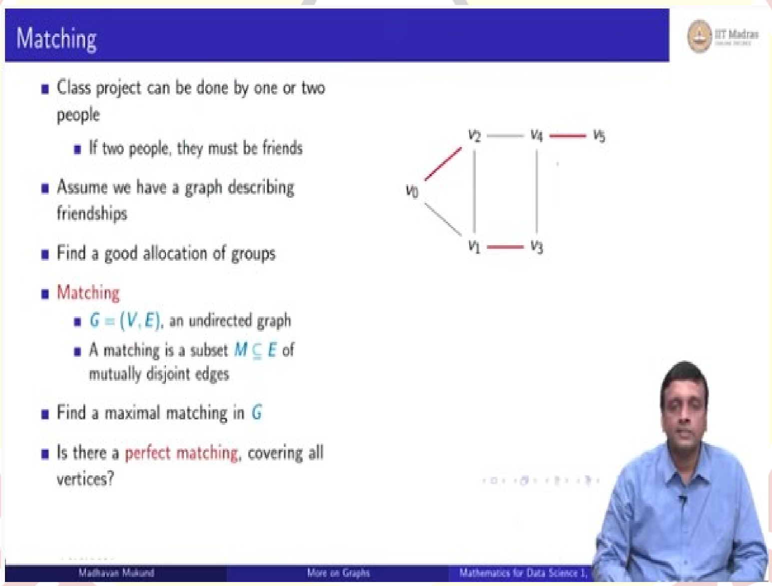
So, let us assume that like we had before we have a graph which describes friendships, what we want to do is given this graph of friends, we know who's friends with whom, we want to find a good allocation of groups, right, we want to find pairs but these must be pairs. So, if I have three people who are all friends of each other a, b and c are all friends of each other, if I make a, b a pair then b cannot be a partner of c, c has to find a different partner, c cannot partner with a, c cannot partner with b, right. So, this is what is called a matching.

So, a matching is a subset of edges which is mutually disjoint, that is if I pick one edge and I pick another edge they do not share any vertex, right. So, here for instance if I pick this edge, right then I cannot pick the edges here, here or here because all of them either touch v_0 or they touch v_2, so I can touch, I can pick any of these three, okay. So, this is the problem that we have and this is called a matching.

So, for instance you might ask for what is called a maximal matching? So, maximal matching like we started this one for instance, right. So, at this point when I have done, when I have chosen this one, right I rule out some vertices but there are some edges but there are still some edges which are permitted, so I can pick one of them and create one more pair. So, for instance I might pick this one but now this rules out this vertex, this edge and this rules out this edge also because now those edges have common vertices v3 and v4.

So, now at this point all the edges have been ruled out or included and I cannot proceed, so this is what I call a maximal matching, right. A maximal matching is one which cannot be extended by adding any more pair without violating some condition.

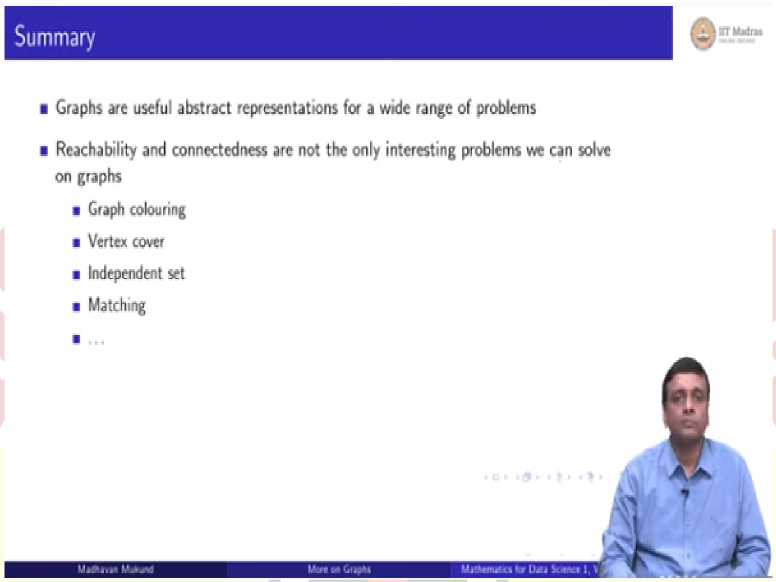
(Refer Slide Time: 20:48)



So, here for instance I what I drew as a maximal matching but here is another maximal matching, right. If I take v_0, v_1 then this has knocked off these edges, right and now if I take v_2, v_4 it has knocked off these edges because v4 is connected to both of them. So, I cannot add any more edges and I am stuck making only two pairs among these six students. Now, ideally if there were 6 students, I should hopefully be able to make three pairs and that is what we call a perfect matching.

So, perfect matching is one which is a matching but also connects every vertex in the graph, so every vertex is part of some pair. So, here is a perfect matching on this graph there are six edges, I mean six vertices and there are three edges and these three edges are mutually disjoint. So, this is the third type of, fourth type of problem that we can see.

(Refer Slide Time: 21:35)



So, what we want to emphasize is that graphs are not just about connectivity and reachability. So, reachability and connectivity are actually very important problems in graphs but that is not the beginning and end of graphs, there are very many interesting problems that you can frame once you put your problem in a graph theoretic sense.

So, we saw graph coloring which we saw an example with scheduling, we saw vertex cover where we saw an example with allocation of say security cameras, we saw this independent set problem which in our case was about having a maximum number of dances where only one dancer can only dance in one dance during the program and then we saw this matching problem of allocating groups within a class.

So, although we will not necessarily look at all these problems in detail in this course it is important to understand why there is so much emphasis on graphs and procedures and algorithms involving graphs because the underlying representation of a graph actually is a very rich representation and by solving problems in this abstract world of graphs you can actually solve a number of concrete problems in one shot without having to solve them individually.