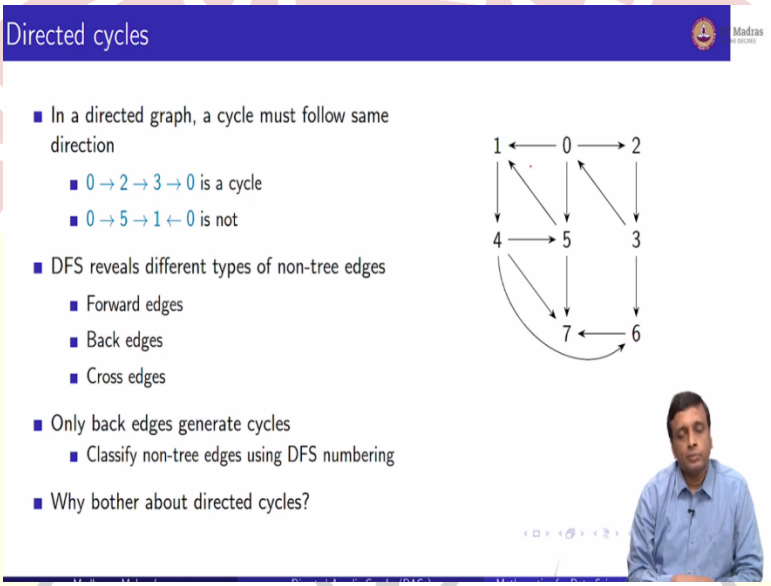
IIT Madras

ONLINE DEGREE

Mathematics for Data Science 1
Professor. Madhavan Mukund
Department of Computer Science
Indian Institute of Technology, Madras
Lecture No. 67
Directed Acyclic Graphs

(Refer Slide Time: 00:14)



So, last week we looked at graphs with cycles. So, we saw that we can use our depth first search and breadth first search to find cycles and graphs. And in particular, we looked at directed cycles. So, we said that if in a directed graph, you have a cycle, then the cycle must follow a uniform direction. So for example, in this graph here, we see that this is a cycle because you can go from 0 to 2, 2 to 3 and back to 0.

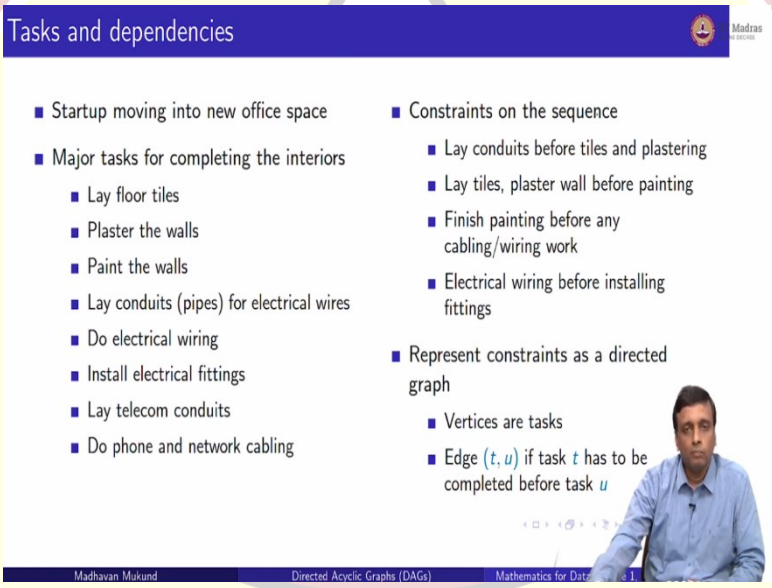
But although this other one here on the left looks like a cycle, it is not because if you go from 0 to 5, and then 5 to 0, then this edge from 0 to 1 is in the opposite direction, so we cannot follow that direction. So, that is not a directed cycle. Whereas for instance, this is a direct cycle. And so this is what we looked at last time. And what we said is that we can use DFS, Depth First Search to find these directed cycles. Because when we do DFS, we will construct a tree to begin with.

So, we have all the edges which are passed are part of the tree which have been used during DFS. And then among the non tree edges, we have 3 different types, so we have these forward edges, which go forward in the tree, we have back edges, which go up the tree from a later node to an

ancestor. And then we have cross edges which go across branches. And we said that only the back edges actually generate cycles.

And by using this DFS numbering by recording the time at which we enter each vertex to process it, and we exit after finishing processing the vertex by looking at these DFS pre and post numbers, we said that we could analyze the tree and look at all the non tree edges and decide which category they belong to. So in this way, we can find out all the non tree edges. And the back edges in particular, which are the cycle forming edges. So, now the question is, why were we so worried about cycles in directed graphs to begin with?

(Refer Slide Time: 02:00)



So, let us look at a general problem, where we have some things to do some tasks, and there are some dependencies between the tasks. So, as an example, suppose there is a startup, which is trying to move into some new office space. So, there is some brand new office space, and the startup needs to set up this office before it can move in. So, this office space is completely unfinished. It is a new building just constructed, just the bricks are there.

And so what we need to do is a number of things, we need to lay the floor tiles, we need to plaster and paint the walls, we also need to lay pipes, these conduits as they are called. In order to take wires from here to there. So, there are wires of two types there are electrical wires, but also there are networking cables for computers, there are also telephone cables and so on, and these cannot go in the same conduit because they interfere.

So, we will have separate conduits for electrical wires, separate conduits for telecom equipment, then of course, you have to put in the wiring. And you have to also after you finish the wiring of the electrical things, you have to put in the fittings, you have to put in the lights, the fans, the switches, and so on. So, now, these are all activities which need to be done, but clearly they cannot be done in arbitrary order. So for instance, we have these constraints.

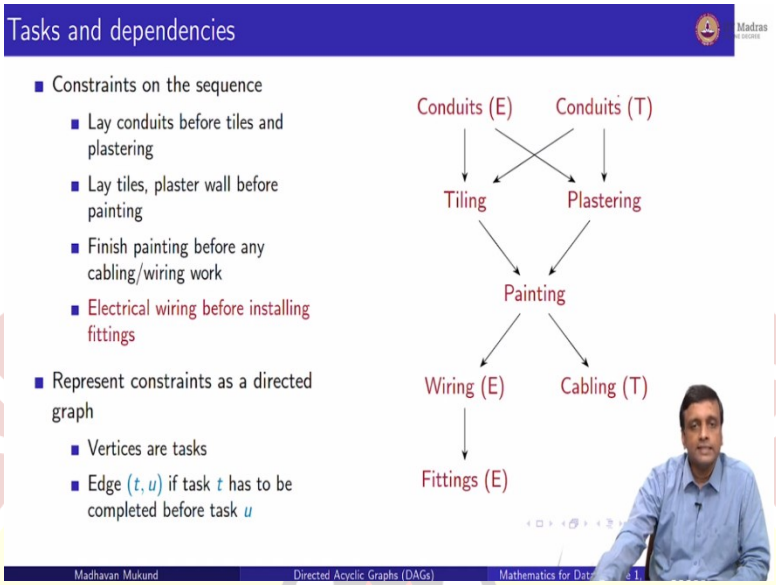
So, we need to put the conduits into the wall and the floor. So typically, these conduits run along the wall to the plug points and to the ceiling. And then across the room, they will flow travel underneath the floor. So, before you put the tiles and before you plaster the walls, you need to put the conduits otherwise, obviously you have to break the tiles or break the walls, which is not a good idea.

Then before you paint the walls, you must of course plaster through the wall. But typically you also like to paint before you lay the tiles because you expect that the person who is laying the tiles might mess up the walls by putting cement when they are laying the tiles. Whereas of course, tiles are usually washable. So, if you are painting it and some paint falls on the tiles, It is not a problem. And clearly you would like to finish the painting before you start putting in the wires into these conduits.

The reason is that if you have the wires hanging out loose when they are painting then the paint will go and gum up the wires and then you will have also paint going into these cracks. So, normally you seal up these conduits with something and then you paint it and then you open it up and put push the wires through these conduits. So wiring cabling happens after the paint and clearly you cannot put your fittings in or you cannot put plug points and you cannot put lights and fans unless the wires are there. So, you can finish the electrical wiring only after installing the fittings.

So, we are going to model this as a directed graph. So, the vertices are going to be the tasks that we have to perform. So, all these tasks on the left laying the tiles plastering the walls and so on. And an edge is going to denote a dependency. So, an edge from t to u says that t has to be finished before you can be started. So, in our case, for instance, you have to lay the tiles before you paint the walls.

(Refer Slide Time: 04:53)

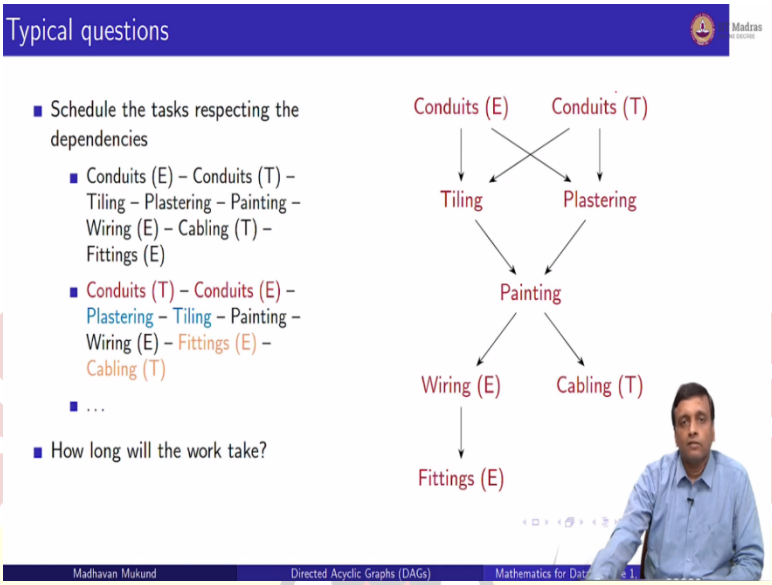


So, if we take this particular thing and draw the graph, so first we have the vertices so we have these vertices corresponding to the activities we have two different types of conduits, electrical and, and Telecom. And we have tiling and plastering. We have painting. We have two kinds of wiring, we have electrical wiring and telecom cabling. And finally we have the electrical fittings. So, these are all the activities. These are the nodes in my graph. And now I have these constraints.

So, the first constraint says that I must lay all the conduits before I do tiling and plastering. So, from each of the conduit nodes, vertices, I have an edge to each of the other two vertices one to tiling and one to plastic. Then it says you must finish plastering and prep and doing the tiling before you paint the walls. So, from both the tile and the plaster edges, we have nodes we have an edge to painting.

Similarly, from painting, we have an edge wiring and cabling because these happen only after painting and finally we can do the fittings only after we do the wiring. So, this is a directed graph, which we have constructed from the task constraints given to us.

(Refer Slide Time: 05:56)



So, what is our problem now our problem is we need to complete these tasks in a way that respects the dependency, so we must make sure that the painter comes only after the tiling and the plastering are done. So, here is a possible way of doing it. So, we start with things which do not have any dependencies, the conduits can be laid right at the beginning. Once they are both done, we can do tiling and then plastering.

Then once both of these are done, we can do painting and then we can do the wiring first the electrical wiring then the telecom cabling, and finally we can put the electrical fittings. So, this is a sequence in which we can complete these tasks so that whenever we come up to take up a task, all the tasks which needed to be done before that are already done. But this is not the only such sequence of course.

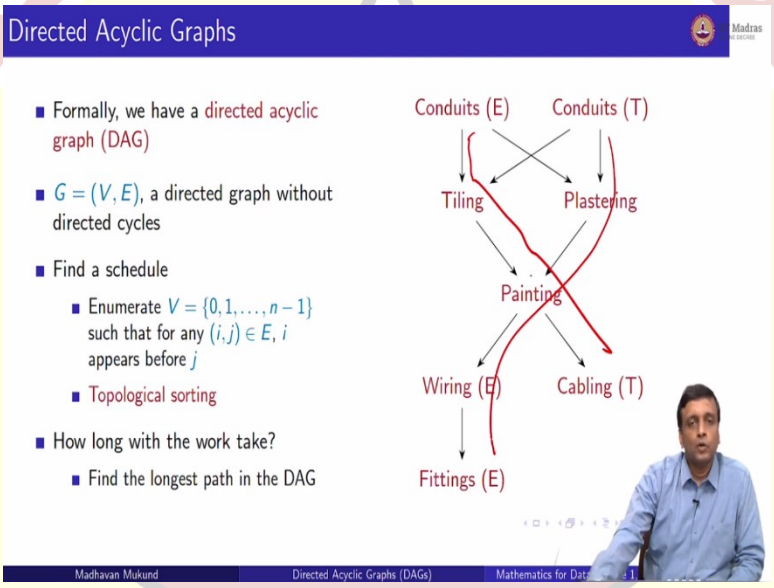
For instance, we could have done the waiting in the opposite order. It does not matter whether we do the telecom cabling (bef) conduiting before or after the electrical conduiting. Similarly, we could take up the tiling after the plastering, because plastering and tiling do not depend on each other. And similarly, we can do the electrical fittings even before we do the telecom cabling, because they go through different conduits and they do not interfere with each other.

Another question we might ask so, this first question is how do we sequence these in a way that does not violate these constraints? The second question is, what is the best way to do this? Supposing we could do things which are independent of each other at the same time. For instance,

we could ask the person who's putting the tiles to work alongside the guy who is plastering because we said that tiling and plastering can happen together.

Similarly, we can have the person doing the electrical conduiting, working alongside the person who is doing the telecom conduiting. Similarly, we can have the wiring and the cabling done at the same time. So, if we can do all this, if we can optimize this so that things which are not dependent are done in parallel, then how soon can we finish this? How many days will it take to complete all these tasks, following these dependencies?

(Refer Slide Time: 07:47)



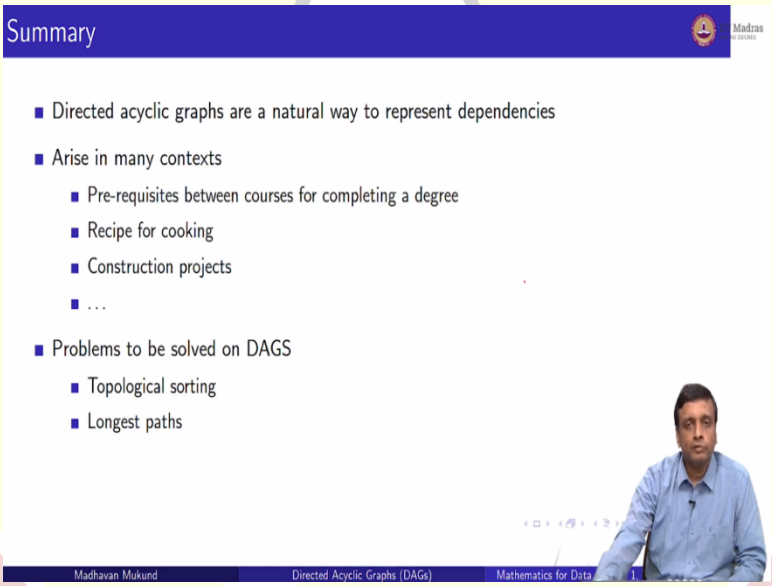
So, if we look at this graph, formally, it is a directed graph. But more importantly, it is acyclic, we do not have any cycles in this because cycles represent dependencies, if a depends on b and b depends on a then which do you do first. So, what we are trying to do is to find a scheduled, which enumerates these vertices in an order, such that in that sequence in the list in which we enumerate these vertices.

If a task i must be done before a task j according to the dependencies, then it must appear before j in the sequence. So, every time we have a dependency, an edge in the graph, that edge, the starting point of the edge has been listed before the ending point indicating with the starting task finished before the second task began. So, this problem formally in a directed graph is called a Topological Sort. So, what we want to do is topologically sort this.

And the second thing is to discover how long we need to move. And then this way, I do not find essentially the longest path. So for instance, we could say that if we start from here, then it is going to take us four steps from starting the conduiting to finishing the cabling. But if we go along this path, for instance, this actually says that we need to do 5 things in a sequence. We cannot do these any faster.

Because plastering can be done only after the conduiting, painting can be done only of the plastering, then wiring and then the fittings. So, we are trying to find the length of this longest path. So, these are the two formal problems that we have with DAGS, topological sorting, and longest path.

(Refer Slide Time: 09:19)



So, to summarize, directed a cyclic graphs are a natural way to represent dependencies. The direction of the edge indicates the direction of the dependency, what must come before what. The fact that it is acyclic follows from the fact that if you have a cycle of dependencies, if i depend on u, and u depend on somebody else, and that person depends on me, then we all depend on each other, so we cannot get started.

So, if I am waiting for you to finish and you are waiting for somebody else to finish and that person is waiting for me to finish, who goes first. So, these cycles cannot be these dependencies cannot form a cycle therefore, it must be a directed and acyclic graph. And these arise in many contexts.

So, we saw this context where we had to finish a room. It could also represent for instance the sequence in which you take courses to complete a degree.

So, courses usually come with prerequisites. So, you cannot do Maths 2 before Maths 1, maybe you cannot do the ML for computation or Computing for ML course, unless you have finished Python programming and both the math courses and both the stats courses and so on. So, now, if you have prerequisites like this, then find a sequence in which you can take the courses to complete the degree.

Cooking is another constraint, a place with a lot of constraints, you need to first of course, make sure you have the ingredients. So, there will be typically a list of ingredients, then there is some processing to be done before you have to chop some things you have to make some things you have to grind some things and so on. And then after that there is a specific sequence in which things go into the pot. So, you put some oil and then you do something else and so on.

So, there are certain things that can be done in parallel 1 person can be chopping the vegetables while somebody else is grinding up something, but there are some things which have to follow a sequence. So, cooking recipes also impose a natural dependency on the tasks in order to prepare a dish. And finally, the kind of problems that we looked at is like a typical project a construction project or any other large project which has many phases, and these phases, some of them can be done in parallel some have to be done in sequence.

And once we have modeled these things as DAGS, we can solve a similar problem that arises across all these different applications by a uniform problem on DAGS, namely topological sorting and longest paths. So, this is what we will be looking at.