Data Structures and Algorithms
Dr. Naveen Garg
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture – 29
DFS in Directed Graphs

Today we are going to be talking about depth first search in directed graphs. In the last two classes we have looked at depth first search in undirected graphs and seen applications for the same. How does depth first search in directed graphs differ from depth first search in undirected graphs? Let's take an example of a directed graph. Let me make sure this has something interesting happening. Let's put this edge back here. Suppose this is the directed graph that I started and this is my start vertex. The process is going to be exactly the same. As in the depth first search, so the code for depth first search would remain the same. Let me write it down once again for your benefit. So depth first search from a vertex v, what did we do. We set visited v to 1 and then for all w adjacent to v. What do you do? If not visited w then DFS (w) and this was the quote for depth first search in undirected graphs.

Now this is the same quote for depth first search in directed graphs. I just need to redefine what the adjacent means now. What do you thing adjacent means? So a vertex w will be called adjacent to vertex v, if I can go from v to w. I am at vertex v, I want to look at all the vertices to which I can reach from v. So these three vertices would be adjacent to v. So sometimes the term out adjacent is used because you might also want to call this vertex adjacent. So x will be called in adjacent but here we talk of adjacent we mean out adjacent, vertices to which you can go from vertex v and the rest are the same. Let's see what will happen now. Let's say I start from this vertex S and I took, this has the first edge to go out. I came to this vertex, now I took this as the first edge to go out. So I came to this vertex. Now I will consider the edges going out of this vertex.

How many edges are going out of this vertex? Only one which is going to a vertex which is already visited. Yes. So there is nothing more to be done at this vertex, I will backtrack from this vertex, go back to where I came from. I come back to there. Now from here I am going to look at the other edges which are going out of this vertex, there is one more edge so that will bring me to this vertex. From here I am going to look at the edges going out of this vertex. There is only one edge which is going out of this vertex that's going to bring me to this vertex.

From here I am going to look at the edges going out of this vertex. There is only one edge going out of this vertex which is going to a vertex which is already visited. So I am done here, I backtrack come back here. Yes, backtrack. Why do I backtrack from here? Because there is no other edge going out of this vertex. So I backtrack, I come back here. There is no other edge going out of this vertex, so I backtrack, I come here. From here now what will I do? I will look at the other edge going out. This is going to this vertex, so I come to this vertex along this edge and from this vertex, I look at the other edge going out, any edges going out, so this edge is going out but it is going to a vertex which is

already visited, yes. So I backtrack from this vertex and I come back to here. There is no other edge going out of this vertex so I backtrack from S which effectively means I have finished the procedure. [Hindi] This is what DFS here would look like.

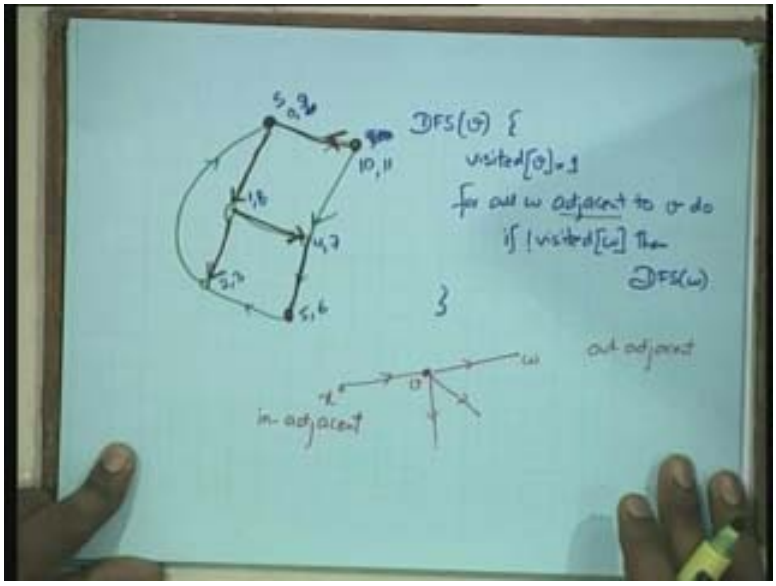
Let me give the arrival and departure numbers to these vertices so that it's completely clear what we are doing. So this zero, can some quickly tell me what these numbers would be? 1 here, 2, 3, 4, 5, 6, 7 that is very easy. We are just saying 1, 2, 3, 4, you are not putting it down on the paper, it is harder for me; 7, 8, 9 here, 10 also here, 11 here. [Hindi] It's clear? This is how our depth first search happens. Now suppose let me modify this graph a bit. Let me change the direction of this edge. Let me make it this way. What will happen now? [Student: vertex will not] This vertex will not get visited.

What will be the departure time of this vertex now? I will come back to this vertex, so I departed from here at time 8, I came back here. Come to here and now there is no other edge going out so I will finish at time 9. Now what happens? What about this vertex, when will this get visited? [Student: this will be only be visit start from] Now we are going to… so depth first search when it starts from a vertex, it might not visit all the vertices. This is true even in the case of an undirected graphs. It will not visit all vertices if the graph is not connected for instance as was the case of breadth first search. You started the breadth first search from a vertex, it would visit only the vertices and that connected components.

Similarly depth first search in an undirected graph would visit only the vertices in that connected component. In a directed graph, the depth first search would visit only such vertices which can be reached from this vertex such that there is a path from this vertex to that vertex. [Hindi] There is no path from this vertex to this vertex. Why, because this vertex is no edge coming into it now. There are only two edges going out of this vertex and no other edge coming in. There is no way of going from here to here. In fact there is no way from any vertex to this vertex. This vertex does not get visited when we do our first depth first search but just as in the case of breadth first search what we did was, if some vertices were not visited we took a vertex which is not visited and started out breadth first search from that vertex.

Similarly we are going to do that here. If some vertex is not visited after we finish our depth first search from this vertex then we are going to pick that vertex which is not visited and continue our depth first search from there so which means that I am going to now pick this vertex, give it an arrival time of 10 now, look at the edges which are going out of this vertex. They are all going to vertices which are already visited. So there is nothing to be done so which means that I also finish this depth first search. In this manner every vertex is going to get both an arrival and a departure time. I will keep this picture and I will again redraw the tree, the depth first search tree.

(Refer Slide Time: 09:31)



We did a similar thing in the last time. [Student: sir we are all vertices] we have to visit all the vertices, so both depth first breadth and depth first search require that you go visit every vertex. So while I did not elaborate on this when we were discussing depth first search in undirected graphs but there I was assuming that the graph was connected. If the graph is not connected, you did a depth first search from a vertex, you visited a bunch of vertices. If you have not visited all vertices then you will take another vertex which is not visited and start a depth first search from here. [Student: sir to see that whether this vertex is not been visited, we also have to go to once again to all the vertices]. We had looked at a procedure for doing this in the clever manner, when we were looking at connected components. All we have to do is traverse the visited array to find the first vertex which is still at a zero and we will never have to retrace, we will just have to make one scan of that array. We can adopt the same procedure here.

Let me now draw. These are the tree edges that you have drawn. Now I am going to draw the back edges. Now we will understand what kinds of edges can there be, just one second. I have drawn all the edges as you can see, so the tree edges are in blue. [Student: we can't] You can't see the colors. Wow, I didn't realize that. That's tragic, let me try to make them darker. Can you see the color now? [Student: yes] It looks a deeper black [student: it is easy bold and fine] fine [student: find in dash dash dotted] so good that you pointed out. Is that today that you can't… [Student: no noise every day you pen moves on that we assume that it is the same color] is it better? [Student: slightly] slightly [student: but because if you use a pen now we realizing its blue] so this blue edge, no it's not blue. Does it look blue now? [Student: its red light red] its red [student: sir sketch pen. Sir we have to make that out by using the sketch that you are using] [Hindi]. Hopefully the others were watching this program, we will be as smart as you are; (1, 8), (2, 3), (4, 7), (5, 6) and (10, 11).

Now we don't have the nice picture we had in the case of undirected graphs that all edges are in two categories either tree edges or a back edges. As you can see there are three different color already and I have not yet used a forth color. There is also a possibility of a forth color and let me show, you could also have an edge which goes from here to here. This is green clearly. This edge could have been there, why? Suppose this edge was there, I am continuing to follow this path. When I come back here to this vertex, I look at the other edges going out, this is another edge going out but its other end point is visited. Now we have to classify these edges, we have to give them names. Our green edge, this edge is a forward edge. Why forward? Because it is going forward in the tree, down. We will call that forward. The brown edge is a back edge, it is going back up in the tree. We don't use the term backward edge, we just use back edge. It is nothing backward about the edge. The red edge is called a cross edge. [Student: which has which one which one].

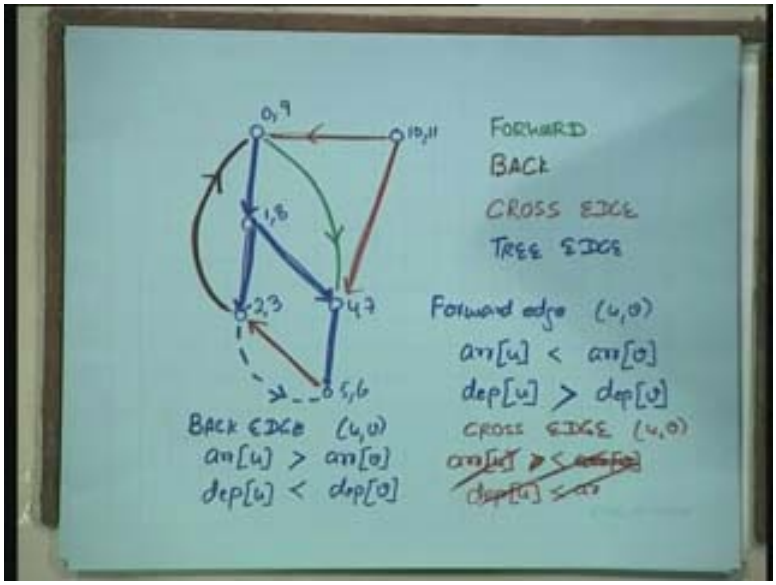
This is a red edge and this is also a red edge and this is also a red edge, these three are red edges. The brown also looks like black. Is it? The brown edge is that edge, it is a back edge. So it should be clear. Tree edge is completely clear, an edge along which we traverse is called a tree edge. Now something that is going forward in the tree is called the forward edge, something that is going back in the tree is called a back edge. Something that is not going either forward or back is called a cross edge. What is a property of a cross edge? [Student: not going to ancestor] not going to ancestors or to a descendant. [Student: descendant]. The two end points of the edge don't have an ancestor descendant relationship, not parent child, ancestor descendant relationship. The two end points of this edge or none of them is a ancestor of the other.

Similarly for this edge, none of them is an ancestor of the other. This is one tree, this is another tree so to say, just a single ten vertex. This is not an ancestor of any of these vertices. These three are cross edges. Could I have a cross edge which goes in this direction? It would have become a tree edge. So cross edge would go in one direction only. What is that direction? We will translate all of these into numbers very soon but essentially if you draw picture in which you are first visiting the left side of your tree and then going right. Then they are going from right to left. The cross edge only go from right to left so to say not higher to lower but more like right to left. This cross edge is also going from right to left, this is also going from right to left. We know this is bit subjective but I will tell you what the actual thing is.

What’s the property of a forward edge, who can tell me? So what is it that makes an edge of forward edge? So a forward edge how can I relate its arrival and departure times. If I have an edge u v which is a forward edge, what can I say about the arrival of u verses the arrival of v. [Student: arrival of u arrival of v] pardon, arrival of u. [student: is less than other] it’s an forward edge such an edge. This is u, this is v. I would have first reached u and then I would have reached v. What can I say about the departure of u verses the departure of v? [Student: departure of v] Clearly I will leave this before I leave that. So departure of v is less than the departure of u. That's for a forward edge. I can play the same game. Let me now say it for a back edge. Once again I have a u v back edge. So arrival of u verses of arrival of v which is smaller? This is a back edge, this is u, this is v.

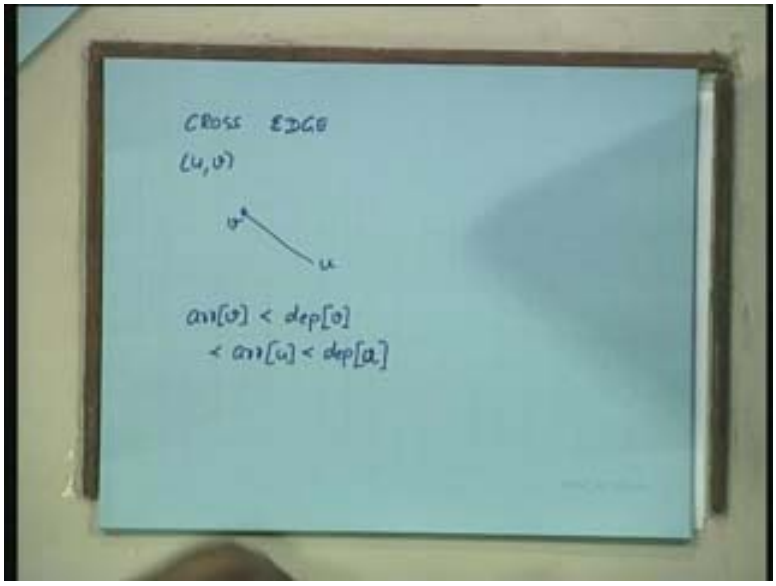
V will be smaller because when I am saying the edge is u v, in a directed edge I always specify the tail first. This is u, that's v. This is v then I reached this before I reached this. The arrival of v is less than arrival of u. What about departure? Departure of u verses departure of v. Departure of u is a u v edge, I will leave u before I leave v. Now let's come finally to our cross edges and since this is more important and you have not seen this before. Once again u v is a cross edge. If this is a cross edge, this is u, this is v. What about arrival of u verses departure of u? [Student: arrival of u] arrival of u verses arrival of v, this is u, this is v. [Student: every] arrival of u is greater. This is u, this is v. I will reach here, after I have reached here. Arrival of u is greater than the arrival of v. I will first arrive at u actually then I will depart form u. Yes, so I should actually write it in a more sophisticated way as departure of u. Sorry arrival of u is not greater, I would first less. I first arrive at u then I depart from u then I arrive oops. What am I saying, sorry.

(Refer Slide Time: 21:25)



I have managed to create a mess there. Let me do it again. So we are considering a cross edge, can you all see this picture? Let me consider a cross edge again. I am looking at an edge u v, so in my picture this is u, this is v. Now it should be clear. So which vertex will I reach first? [Student: v] v, so first I reach v then I leave v. That's important then I reach u, then I depart from u. Yes.

(Refer Slide Time: 22:49)



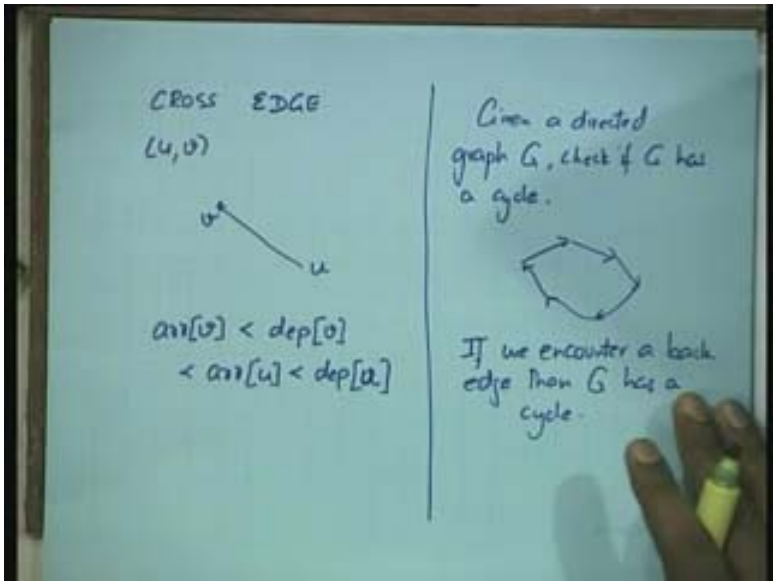
The arrival and departures are arranged in this way for a cross edge. Great. So in a depth first search the edges get classified into four categories and based on the arrival and departures, you can figure out what those categories are or based on the fact that whether they are a tree edge or not. So first you will create the tree edges, first you will identify what the tree edges are. Using the tree edges you can identify the parent child relationships also, ancestors descendant relationships and using that you can figure out whether it is a forward edge or a back edge or a cross edge. If the two end points have an ancestor descendant relationship then it's either a forward edge or a cross edge or a tree edge.

If they don't have then it has to be a cross edge. Sorry in the first case it's either a forward or a back or a tree and in the second case it is a cross edge. When you have to distinguish whether it is a forward or a tree or a cross, then again you know it's based on whether the tail is an ancestor of the head or whether the tail is an descendant of the head. You can use that to figure out these things. So as an application we are going to look at how to check if a given graph has a cycle or not. So that's the application we are going to look at. I am giving you a graph G, so given a directed graph G check if G has a cycle. What is a cycle in a directed graph? It's basically a path which closes on itself, the starting and the ending of the path are the same.

How will you check if a given directed graph has a cycle? How will you check if a given undirected graph has a cycle? [Student: starting node if we have any back edge if we access the starting node] [Hindi] so then there is a cycle. If there is a back edge there is a cycle. If there is no back edge, is there no cycle in the graph? [Student: yes] Because then the remaining edges are just tree edges, they form a tree. A tree does not have a cycle in it and a back edge, why does it form a cycle? Because the two end points of the back edge are connected in the tree. There is a path between those two end points.

So for an undirected graphs, it is very simple. For a directed graph it is not so simple. When does a directed graph have a cycle? Once again you want to do a depth first search. In a depth first search in what's [student: the back edges are definitely] If there is a back edge then the graph has a cycle. If we encounter a back edge then G has a cycle. Great, why? Well, let's just look at an example. This was the tree, this was the back edge, what is a cycle? Well, the cycle is exactly this.

(Refer Slide Time: 26:32)



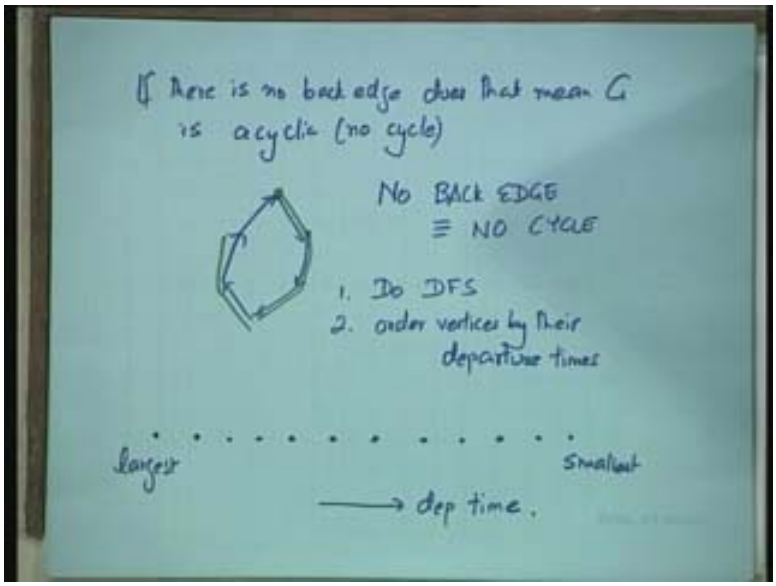
Since it is a tree, this node is an ancestor. This is a back edge so this is an ancestor of this. If this is an ancestor of this, there is a path in the tree from here to here. I will go down to one of its child and so and on and reach here and this together with this forms the cycle for us. If there is a back edge, there is a cycle. If there is a cross edge, does that mean that there is a cycle? So question is if there is no back edge, does that mean G is acyclic? I am using a term here, what does acyclic mean? No cycle. [Student: there is a cycle consider a graph it is only a cycle when definitely you will not have a back edge] He is saying consider a graph which is only a cycle. We will not have a back edge. It will have a back edge. Is it clear? So he is saying consider a graph which is only a cycle [Hindi]. When we do a depth first search of this graph, what do we get? No matter of what vertex I start from.

I will let's say go down here then I go down here then I go down here, here, here and at this point this is will become a back edge because I will just retrace my path here and reach here. So this will become a back edge. So back edge, if you find a back edge there is a cycle. If there is a cycle, it seems that you will get a back edge. Can you prove this? If there is no back edge does that mean that there is no cycle in the graph. No. [Student: all forward] If there is no back edge, does that mean G is acyclic? [Student: due to a cross edge also because it can happen that in a particular cycle the child is traversed by some other path like the origin of cross edges] [Hindi].

So let me do a very interesting proof of this statement or what is a statement? This is not a statement, this is a question. So statement is no back means no cycle. No back edge, no cycle. [Student: for a cycle does it mean a arrow is going in one direction] Of course, this is a directed graph. A cycle means a path. I was very clearly specified in a directed graph, a path means you can go from one vertex to the next. Think of them as one way streets.
A cycle would mean only if you can traverse in such a manner such that you can reach the starting point. Great.

So no back edge means no cycle and how am I going to prove this? Let me do the following. I will do a depth first search and now I will order the vertices according to their departure times. So do DFS, order vertices by their departure times. What does that mean? Let's say I will first put down the vertex which has the largest departure times say [Hindi]. This has the largest and this has the smallest and the departure times are decreasing like this. This is the largest side [Hindi]. So this is the vertex which has the largest departure time, no two vertices at the same departure time. Yes because every time we depart from the vertex, we change the time. So this is the vertex with the next smallest, the next smallest, the next smallest and so on and on.

(Refer Slide Time: 31.40)

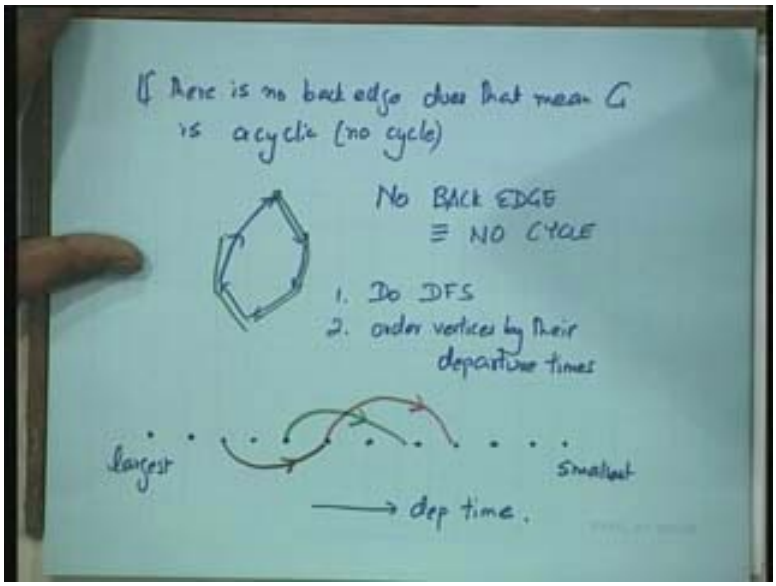


Now let's go back to all that we have done so far. Let's look at this picture. For a forward edge, if u v is a forward edge then the departure time of u is more than the departure time of v, [Hindi] yes. So the departure time if u v is a forward edge then the departure time of the u, u is the tail, v is the head of the edge. The edge is going from u to v. The departure time of u will be more than the departure time of v. So if I have a forward edge then will it go from left to right or will it go from right to left? If I have a forward edge its tail would have a higher departure time, so it could go from left to right. A forward edge would like this.

Let's look at a cross edge now. A cross edge, u v cross edge the departure time of u is again more than the departure time of v. So the edge will also go, a cross edge would also go from left to right. [Hindi] there is a cycle in the graph. So what edge remains? [Student: tree edge] tree edge. What is the property of the departure time if u v is a tree edge then departure of u v [Hindi], so first I will leave from v and then I will leave from u. So departure of u would be more than the departure of v. great. So a tree edge also goes like this. All edges are going from left to right. How can they be a cycle? Can you create a cycle by just going from left to right? No. If you have to come back to the starting vertex [Hindi] and you have to come back to this vertex, [Hindi] you can go forward but at some point you have to come back but there is no edge which is coming from right to left. So there is no cycle in this graph. [Hindi] yeah, everyone follows this proof.

Now you know why we were worrying about departure times. So you have to do this with departure times. You can't do this with arrival times, unfortunately. So try that as an exercise. If I were to order them by arrival time this is not going to work. [Hindi] So this is actually a very simple proof theorem. If there is no back edge in the graph then there is no cycle. If there is a back edge then there is a cycle. So all you have to do is do a DFS, if at any point you encounter a back edge, you declare that the graph has a cycle. If you are able to finish your depth first search without encountering a single back edge then you declare the graph is acyclic. [Hindi] What is an acyclic graph? A acyclic graph is a graph which does not have a cycle.

(Refer Slide Time: 35:45)

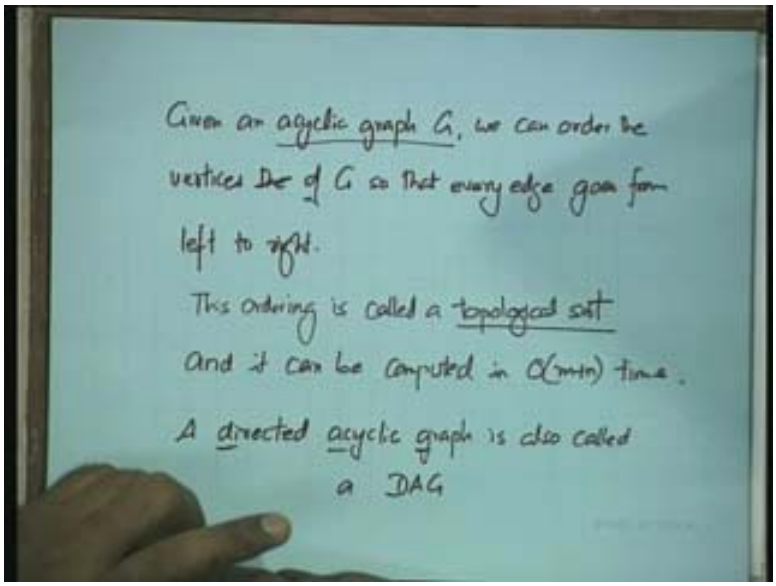


This ordering so… what we have… Another thing we have shown and that is the very nice thing is that given an acyclic graph, let's say G is an acyclic graph. We can order the vertices of G so that every edge, so that every edge goes from left to right, yes. That's what we did here. We said we started with an acyclic graph. We started with a graph, we did a depth first search in a graph.

We did not encounter any back edge. So we said the graph does not have a cycle. It is an acyclic graph and then we said lets order the vertices of the graph according to decreasing departure times. Then what we see? If we order the vertices in the manner then every edge goes from left to right. This ordering is also called a topological sort [Hindi]. So in a acyclic graph you can order the vertices, linearly order the vertices so that the edges are going only from left to right. This ordering is called the topological sort. So how much time do you take to find a topological sort? You just have to do a DFS and then, then sorting. [Student: this is] check what? [Student: is there exists some] [Hindi] but you have to produce an ordering of the vertices right. [Student: Hindi] Right, so we don't need to sort because we know what the departure times are. What is a maximum departure time we can have? [Student: 2 n] 2 n, yes 2 n actually 2 n minus 1 because each vertex is getting two numbers, right.

So the total set of numbers that we are going to be assigning arrival and departure times will be in the range 0 through 2 n minus 1. So the departure times is at most 2 n minus 1 that is at least to 1. So I can just have an array with 2 n entries in it and as I depart from a vertex, as I assign it with departure time at the position in the array I put down the vertices. I just need to make one scan of this array to get the vertices in the right order. [Hindi] So you don't need to sort the n departure time because if you had to sort, you would take n log n time. We have not seen a sorting algorithm which performs better than n log n. So this ordering is called the topological sort and it can be computed in order n time, order n plus m time. So I have used, so I should also tell you about another term that is used for acyclic graph. So we are talking of directed graphs here. A directed acyclic graph, directed graph which is acyclic is also called a DAG, D A G [Hindi].

(Refer Slide Time: 40:29)

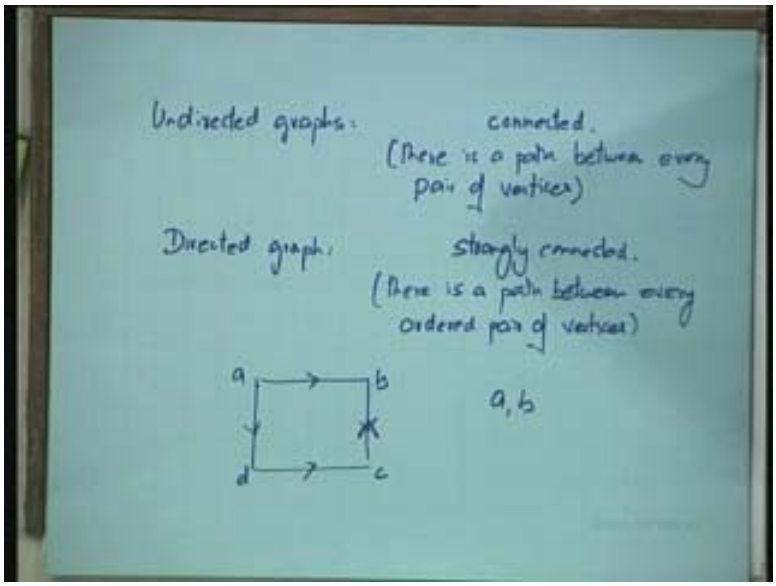


These graphs arise quite a bit in circuits, in combinational circuits where you know your pulses are essentially traveling from one side to the other. There is no notion of a cycle, there is no loops and so these graphs model that and we can do a lot of things on such

kind of graphs which we cannot do on a regular directed graph. The graph which has cycles in it and we will see some of that in this course. Questions so far? [Hindi] So let me introduce, I will like to take one more application of DFS in directed graphs and since that is a longer application, we will not be able to finish it in this class but I will develop the terminology for it. So let's go back to our notion of… so for undirected graphs, so if I gave you an undirected graphs there was a notion of whether the graph is connected. Yeah, when is an undirected graph connected? When it all vertices [student: there is a path between every two vertices] there is a path between very two vertices then we say that the graph is connected. So connected means there is a path between every pair of vertices. In a directed graph the corresponding term is what is called strongly connected.
A directed graph is called strongly connected, if there is a path between every ordered pair of vertices.

Why am I using this term ordered pair of vertices? Between which, so I said pair of vertices but you know it is a directed graph. May be there is a path from u to v but there is no path from v to u. Let's take an example. Is this graph strongly connected? Let me label. There is a path between a and b but there is no path from b to a. There is a path from b to c but there is no path from c to b. there is a path from d to c but there is no path from c to d. [Hindi] It is still not strongly connected. There is a path from a to b [student: but not from] but from b to a there is no path still. Pardon yeah, sorry. [Student: what do you mean by the] no, no when I say there is a path from a to b it basically means that I can go from a to b like this or like this right but I cannot go b to a there is no path from b to a. Is there a pair pair of vertices such that there is a path between these two vertices in both directions. [Student: no] no cannot happen right because not in this graph.

(Refer Slide Time: 44:55)



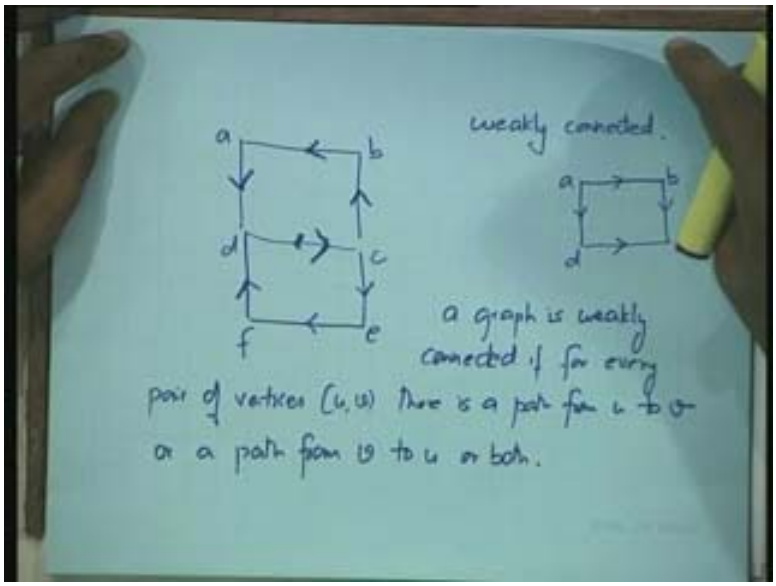
Let's see more examples which will perhaps be better. So a b c d, is this strongly connected? No, yeah between any two vertices. So from d to b there is a path, from b to d also there is a path because everything is on a cycle. [Student: cycle] yes, so this graph is

strongly connected. Is this graph strongly connected? [Student: yes sir] [Hindi] is there a path sorry, is there a path from a to f? [student: yes yes] yes [student: a to b] yes a and f there is also path from f to a going like this. So this graph is also strongly connected. No that is not sufficient right, I can't just take two pairs and check them and say it is strongly connected. We have to look at every pair right but you can check. [hindi] all of these form one strongly connected component. For all these 4 vertices any pair I choose is connected. Similarly for these four vertices any pair I choose is connected yeah and that should somehow tell you how to do things. [Student: both of them must be part of cycle] right something like that.

If there is a notion of strongly connected there should be a notion of weakly connected, not daily weakly connected, sorry. It's weakly connected. What do you think is a weakly connected graph? [Student: between u and v all v to u] yes [student: as undirected graph] no no, not as undirected they are connected. The following is [student: there is a path from u to v or] there is a path from [student: u to v or v to u] or v to u and or v to u. The following graph is weakly connected. [Student: yes] yes [Hindi] Is there a path from b to d? No. Is there a path from d to b? No. Is it weakly connected? [Student: no] no, this is not a weakly connected graph.

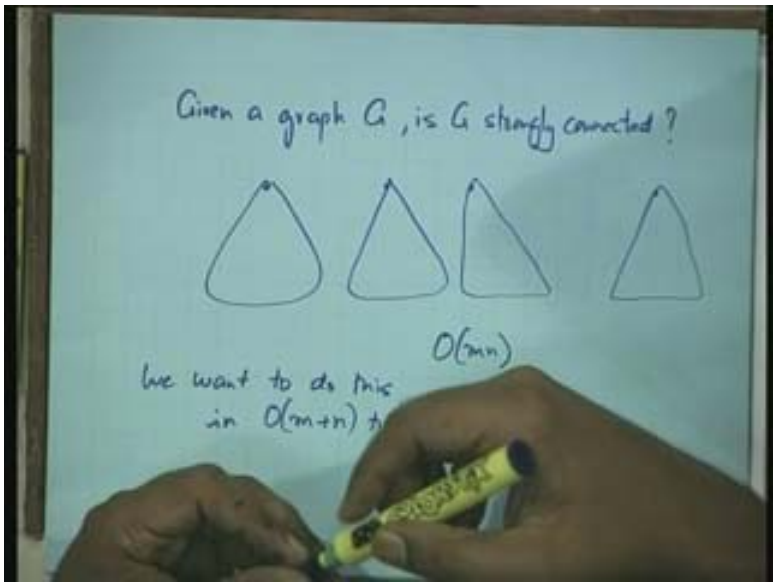
So why am I showing you this example, it's not sufficient to say... [Hindi] So this is a graph such that if I were to ignore the directions, it is a connected graph but it is not a weakly connected graph. So what is the right definition of weakly connected? A graph is weakly connected if for every pair of vertices u, v there is a path from u to v or a path from v to u or both. If both the paths are there its okay and this should be true for very pair of vertices right. [Hindi] right, good. So I have told you what is strongly connected graph is and what a weakly connected graph is.

(Refer Slide Time: 49:36)



So natural question is given a graph, is it strongly connected? [Hindi] given a graph G, is G strongly connected? So what are you saying? We are doing a DFS starting from here. So we have to check, so strongly connected means we have to check that from between every pair of vertices, every ordered pair of vertices there is a path. So from u to v there should be a path and from v to u there should be a path. So one solution that is being suggested is that you take one vertex, do a DFS from here. It should visit all vertices and then take another vertex and do a DFS from there. It should also visit all vertices. Take a third vertex and do a DFS from there, it should also visit all vertices. So do this DFS. How many times? n times from every vertex. If each of those DFS's visit all the vertices then the graph is strongly connected. Yes, is that clear. Is this argument clear? If all of these DFS visit each and every vertex then the graph is strongly connected. Perfectly okay, except how much time does it take? Order m n time. That's too much for us. We want do it in linear time. So we want to do it in order m plus n time.

(Refer Slide Time: 51:45)



So question is how did you do that? [Hindi] So try to think about this and think in terms of the procedure we used for two edge connectivity. We will borrow ideas from there. In two edge connectivity what did we require? We required that for every sub tree there should be an edge, back edge going out of the sub tree so to say, going to a ancestor of the root. Here also we require something similar that's the hint. So you will have to see what that is and we will discuss it in more detail in the next class.