

**Data Structures and Algorithms Using Java**  
**Professor Debasis Smanta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture: 35**  
**Topic: Graph Structures**

Let us start another very interesting data structure, it is called graph structure, graph data structure and there are a few data structures which are called classic data structures. We have already studied few such data structure including tree, linked list, array, stack, queue and everything. So, graph data structure is one very important candidate in the list of all data structures and this data structure has been received many attentions from theoritist, many scientist in the field of computer science actually and many theorem and many research paper on this data structure have been published so far.

(Refer Slide Time: 1:25)

**CONCEPTS COVERED**

- Concept of Graph
- Representation of Graphs
  - Set Representation
  - Linked List Representation
  - Matrix representation
- Operations on Graphs
  - Insertion
  - Deletion
  - Traversals

The diagram shows a graph with four nodes: A (top), B (bottom), C (left), and D (right). Edges are labeled as follows: 'a' connects A and B; 'b' connects A and C; 'c' connects A and D; 'd' connects C and D; 'e' connects B and D; 'f' connects C and B. There are also curved edges between C and B.

Now, so today we will try to understand about this new and interesting, fascinating data structure. So, first we will try to understand about the graph and then definitely like all other data structures we have to study about how such a data structure can be stored in computer memory that is, representation of graph and then we will try to see some important operations which can be applied to these data structure. So, this is basically the plan for this lecture.

(Refer Slide Time: 1:55)

**CONCEPTS COVERED**

- Concept of Graph
- Representation of Graphs
  - Set Representation
  - Linked List Representation
  - Matrix representation
- Operations on Graphs
  - Insertion
  - Deletion
  - Traversals

The diagram shows a graph with four nodes labeled A, B, C, and D. Node A is in the center, B is below it, C is above it, and D is to the right. There are multiple edges connecting these nodes, forming a complex network.

Now let us first come to the concept of the graph data structure. So, it is very early in the, may be very beginning point, mathematics grows in different application and that time Oiler who is called the Father of this Graph theory. So, Oiler started one, I mean introduced on problem this is actually called on puzzled actually which many people could not solve for many years and this puzzle related to one problem it is called Cronics Park Bris problem.

So, in Russia there is an island it is called the Cronics Park name and this island is connected with five different bridges and that problem why is that how one can travel all the lands that is there in the islands crossing all bridges exactly once. So, this problem famously called Cronics Park Bris problem and is basically the starting point of this graph structure actually and the graph concept there.

(Refer Slide Time: 3:16)

**Tree and Graph**

The slide shows two diagrams: a tree structure on the left and a graph structure on the right. The tree has a root node with several children, and the graph has a cycle of nodes connected by edges.

- Two structures are similar in the sense that they represent connectivity among the nodes
  - Both belong to the category of **non-linear data structures**
- They are different in the sense that tree is **acyclic** where as a graph usably **have cycle(s)**.

NPTEL Online Certification Courses  
IIT Kharagpur

Now, so graph is basically is very similar to a tree actually, and we have already learned about that

tree and in tree the elements in these data structures are of two types, they are called nodes and edges. In graph also same thing nodes and edges. So, nodes are there and all edges basically connects all the nodes whatever the way it is possible and for a particular tree called the binary tree where each node has at the most two edges actually and there may be some nodes which does not have any edge or whatever it is from it actually.

The difference between the tree and graph is that and both the things, the tree and graph they are termed as non linear data structure because if you want to store this graph you will not necessary to use them into a contiguous memory location. You can store whatever the memory it is available there, because they are basically all linked, connection where these nodes and other nodes have to be connected through link actually.

So, in that sense both the data structures are called non-linear data structures. But the difference is that in case of tree there will not be cycle. So, that means starting from a node you will be able to reach to another node, but you will never be able to come back to the previous node from where you have started. On the other hand, if the graph data structure is basically a cyclic. Cyclic in the sense that there is a possibility starting from a node you will be able to reach to the same node again.

For an example the first picture that I have shown is basically tree as you see. Starting from this node we can reach to any other node or this node to this node like this one. But here if you see from this node we can start here and go there and go there and go and then come back to this one again. In that sense it is called the cyclic. So, these are the main difference between the tree data structure and graph data structure from the first view of them but it has many other difference and other and many other issues which are basically makes the graph is an important concept to be studied actually.

(Refer Slide Time: 5:57)

**Example of a graph structure**

(a) Graphical representation of airlines in East Asia

(b) Graphical representation of 2 commodities in 3 destinations

(c) Königsberg's bridges and their graphical representation

(d) Flowchart

- Other example includes:
  - Flowchart, Electrical network, transportation network (road, rail, bus, etc.), postal communication, courier service, supply-chain, etc.

NPTEL Online Certification Courses  
IIT Kharagpur

Now, so this is basically so essentially what you can say a graph is basically a tree or rather we can say that a tree is a graph. But a tree is a graph but and then a graph is also a tree but all graphs is not necessarily a tree actually. That we can tell about. Now, let us come to this concept there are few examples that I want to tell about that why this data structure coming in the picture and what is its application importance rather.

There are many situations which basically can be represented using the graph structure. Now, for example, say suppose in this first picture if you see. So, this is a basically an example of a graphical representation of all the airports those are connected in say particular region or country like see East Asia. So, in this figure you can see so that these are the different nodes. Each nodes represents an airport and then an edge represents that from that airport to which other airport that you can fly.

So, to and fro fly from Delhi to Guwahati and these kind of things are there. This basically is a good example that is a graphical structure that means network of all the airlines that is the air is basically the graphical structure. And I told you the Königsberg's bridge. This is basically the Königsberg's bridge these are the basically 4 islands and 4 islands are separated, are connected rather connected by these are the 7 bridges. So, this is basically we can represent these islands and bridges by means of a graph which is basically shown here.

So, this basically the island Königsberg and this is graph structure. And as we can see here basically all islands are represented by nodes, so A, B, C, D. These are the 4 nodes and all bridges are connecting from one node to another, means one land to another part of land actually there. So, these are the different edges. What we can say that a geographically landscape also can be

represented with the help of a graph.

This is another one very popular applications which you are familiar to is basically source and distributions. Now this is related to many context, for example, here the different commodity that needs to be supply to different destination, for example, here 3 destinations unlike this one. Now, how is form each from each commodity can be supplied to different destination is depicted in this figure.

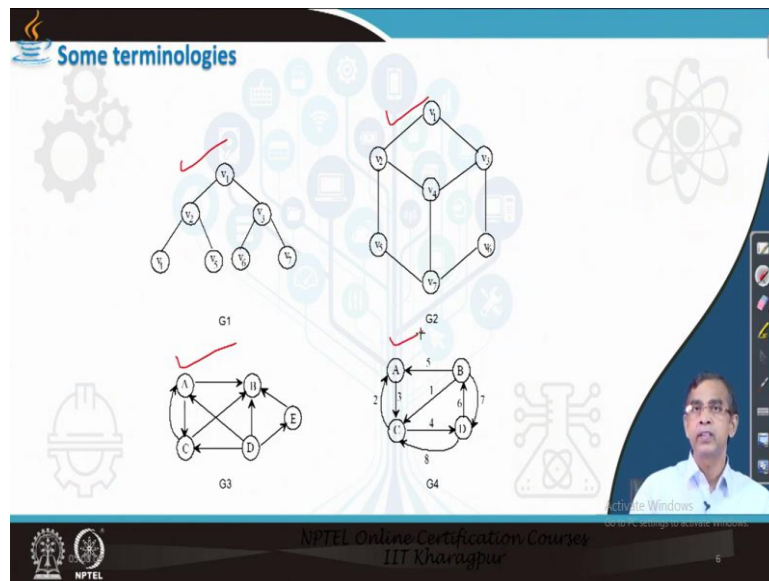
This is very simple example, but in actual situation, actual scenarios there may be many commodities and therefore many destination and there are many routes or possible paths from one to another, not necessary direct what is called the supply there may be via or intermediate supply. Then this graph structure will be really very complex actually.

And this is the another example this is related to software. And you know any program that you write is basically lead to a software. Now whenever you have to represent or you have to test a program, so basically the best idea it is that if we can have a flowgraph of the program. Alternatively if you have to write a program, then how to write the program so maybe your designer or your manager can tell you so this is the flowgraph.

And given the flowgraph you have to write the program, so write means how many loops will be there, where the loops will be there? What are the different paths? What is the decision statement and everything? That is basically that is depicted very nicely in a graph.

This is basically another example of a graph. This is called a flowchart or simply a flowgraph also. Here also you can see there are several nodes and then edges. And nodes and edges are connected and you can starting from one particular node you can reach to another nodes and this kind of things are there. So that I have mentioned few examples there are plenty of examples which I can mention and they are basically the graph concept is needs to be applied in order to understand each situation actually.

(Refer Slide Time: 10:55)



Now so we have understood about the graph concept and then we know that graph is very important one things in our daily life. In fact, to solving many problems in our daily life actually. Now, let us come to theory of this graph data structure. Now, there are basic terminologies which we have already studies related to their node, related to the tree it is also applicable here. Main terminology that is there is basically node and edges.

For example, here as we see this is one tree. It is essentially a graph also. This is another graph. And here you see the nodes are labeled with certain names or something like. Actually each node represents either label some string name or it can represent many other source or other important information, can be a number also. There are many other things that you can store in the form of a node actually. Like tree we have learned about that it can store a record, is a pointer to a record, it can store a file. It is like this, so it can be like this one.

Node can be in fact a server. Then another node can be a computer connected to the server. A node can be OMPS from there is a link. So, what is node? There are many information that you can store actually. So it is basically is a complex data structure, is an abstract data structure that we can put as the node. And edge basically the same thing as in tree that is there in the graph also is a link.

Link means connection from this and this. So, it basically predecessor successor relationship, who is the predecessors of what. It is just like parent, parent child relation like. There is another concept, which is basically different than the tree is that in case of tree, so direction is implied that means direction is from always parent to child. So from a given parent you will be able to reach to their children, but opposite is not possible.

This means they from a children, I will not be able to go their parents actually. But in case of graph, the predecessors successor relationship, so from a given node you will be able to reach to another nodes. Or from the another node you will be come back to the same node actually. This means that in case of graph the edges will be either undirected or directed.

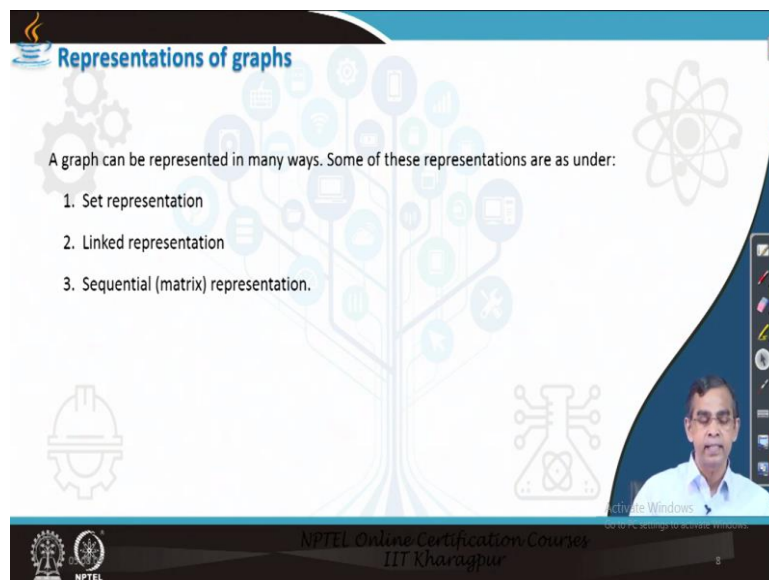
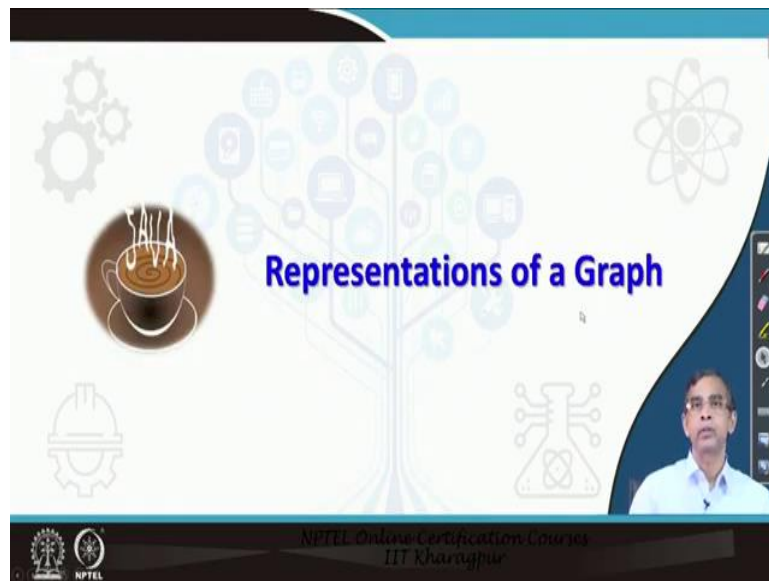
Now here, in this example, so this is an example of a graph why are edges are undirected. This is also another. But here in this example if you see graphs are, nodes are connected with directed edges. What is the difference is that in case of directed edges you will be able to go from a node to another node. If the direction permits otherwise it is not. For example, from A to B we can go, but from B to A you will not be able to go.

On the other hand, if it is undirected  $V_1$  to  $V_2$  or  $V_2$ ,  $V_1$  both way movement is possible. Now, so this is another difference that you can think about. I mean with respect to tree, the graph has the different characteristics. Now, so we have understand about directed undirected. Now, directed event or undirected, whatever it is there, they can be associated with some value that is called the weight. That means each edge can have their weight.

Now here for an example this is an example of a graph which is basically directed as well as weighted. What is the meaning of this weight is that say for example from B to A there is a supply. What is the maximum supply that you can say it is a 5. Or say B to A this is a road and then what is the load that traffic can flow through this road it is basically 5, so this number implies something meaningful.

So, this is the weight, these are the weights. And so in case of graph, that graph can have directed or undirected. Direct weighted, undirected weighted many possibilities are possible. So I have listed few examples here. There are many more examples also possible.

(Refer Slide Time: 15:03)



Now we have understood about basic concept about the graph and then some applications and then few basic terminologies and while we will study we will continue our discussion we will, I will introduce few more terminologies in this regards. Let us come to the discussion about representation of a graph. There are many ways that a graph can be represented. Now the three different, three major representations are very popular among the scientist.

First of all is very old one and it is called the set representation. Second is basically called the link representation. Link representation is very similar to tree type. And another is called matrix representation. It is called the sequential representation. It is just using a matrix and set representation is basically using a text type things it is there which is very compact but it has very limitations so far operation is concerned.



Whereas, the link representation is the best data structure representation for the graph. Sequential representation is very good, is very fast but it has limitation that it is only support static weight. That means one you can represent it you will be able to I mean extend it adding few nodes or deleting few nodes is bit difficult. Anyway so these are the three different representation. Let us visit each representation one by one with examples.

(Refer Slide Time: 16:35)

**Set Representations**

**Graph G1**  
 $V(G1) = \{v1, v2, v3, v4, v5, v6, v7\}$   
 $E(G1) = \{(v1, v2), (v1, v3), (v2, v4), (v2, v5), (v3, v6), (v3, v7)\}$

**Graph G2**  
 $V(G2) = \{v1, v2, v3, v4, v5, v6, v7\}$   
 $E(G2) = \{(v1, v2), (v1, v3), (v2, v4), (v2, v5), (v3, v4), (v3, v6), (v4, v7), (v5, v7), (v6, v7)\}$

**Graph G3**  
 $V(G3) = \{A, B, C, D, E\}$   
 $E(G3) = \{(A, B), (A, C), (C, B), (C, A), (D, A), (D, B), (D, C), (D, E), (E, B)\}$

**Graph G4**  
 $V(G4) = \{A, B, C, D\}$   
 $E(G4) = \{(3, A, C), (5, B, A), (1, B, C), (7, B, D), (2, C, A), (4, C, D), (6, D, B), (8, D, C)\}$

NPTEL Online Certification Courses  
 IIT Kharagpur

Now, I first discuss about Set Representation. Here I have given few graphs G1, G2, G3, G4 for few examples. Now whatever be the type whether directed or undirected or directed weighted and undirected weighted no issues you will be able to represent a graph using Set Representation.

Now, in this Set Representation actually there are two sets; one is called the sets of all vertices that is the nodes. Now, for example, for this graph G1 which is basically simple out of so many graphs we have listed here. These are the set of vertices involved V1 to V7. This is basically a set. Then another set is basically set of edges. What are the set of edges? V1 to V2 one edges, V1 to V3, V2 to V6, V6 to V7, V3 to V7.

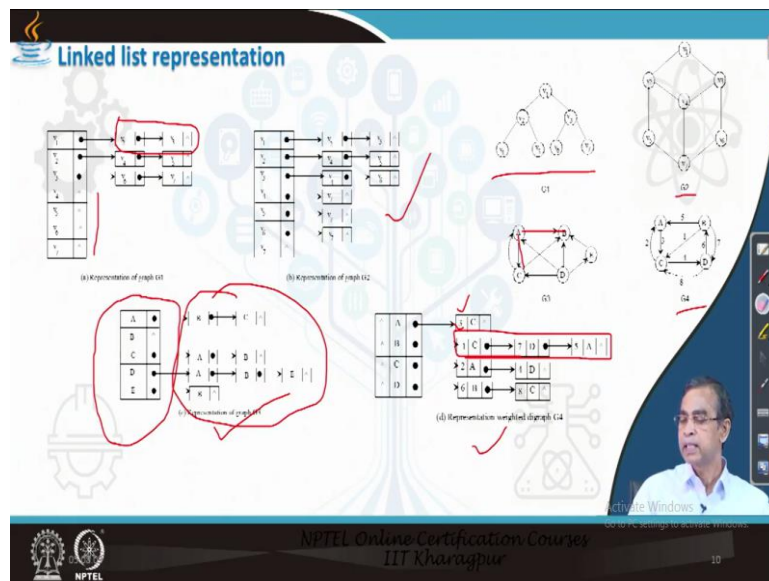
Now, all the set of edges are stored is another set. So, there are two sets V is the set of vertices and E set of edges. Similarly G2, you can check it that this is the graph set representation of the graph G2. Now directed graph also similar you can represent. We can include an edge from A to B if there is a direction. For example here A to B edge is there but B to A edge you see it is not there.

Now, so there is a problem whether this is directed graph or undirected graph. We assume that this is undirected graph unless if you say specific thing it is there. But whenever we say graph G3 we say that it is a directed graph. And the direction of edges is represented by the set. And then this set

of nodes or vertices are the same in any way whatever it is there. Now weights also can be, in case of weights there are so three data needs to be stored, first of either weights and then second 2 is basically the direction from which vertices, source vertices and then destination vertices.

So, 3A to C that means it is an edge, undirected which direction is from A to C. For example, here A to C and the weight of this edge is 3. So this is the representation of the graph G4. So, whatever be the graph if you have given you will be able to represent with the set representation very easily n very compact representation you will just you can check it. So, this is a Set Representation.

(Refer Slide Time: 19:15)



Let us come to the Link Representation. Link Representation use the link structure. The link structure means you have studied about single linked list, double linked list. Here we usually follow the single linked list but you can follow double linked list also no issue. That may be some advantages in some situation. But single linked list is enough actually.

Now in this link structure actually there is basically the master array, it is called the pointer array. Pointer array, size of the pointer array is basically same at the all total number of vertices are there in the graph. So, for example this is G1, this is graphical representation. This is linked list representation of a graph G1. And here you see 7 vertices are there so this pointer array stores 7 what is called the elements. So here V1 to V7, are 7 elements. And this is a pointer array because the another field is used to point to some other linked list.

So, here putting V1 points this is another linked list. So basically what you can say V1 in the pointer array basically header to the linked list basically. This is the header of the linked list, this is another header to linked list and so on, so on. Now here we can see that V1, from V1 we can go to V2. So,

V1 to V2 and from V2 to V3, from V1 to V3 also can be. So, there is a link from V1 to V2 and V2 to V3. Similarly from V2 to V4 this is the link. From V3 to V6, V7 it is there.

And there is no link from V4 or V5, V6, V7 because they are null, so these are the remains null. This is link representation of a simple graph G1. Now, likewise G2, the link representation is shown here, you can check it how the representation stands actually. It is the same as the G1 is represented, so number of vertices 7 so here the pointer 7, pointer array is 7. Each pointer points to linked list giving that what are the connection from V1 to others, V2 to others, V6 to others and so on, so on.

Here, you can see from V7 we will not be able to go anywhere, so there is no pointers whatever it is. Now, this is the directed graph representation, undirected graph representation rather. Now G3, G4 are the directed graphs. So, the directed graphs representation for G3, G4 for example G3 is shown here.

You can check it again why it is there. From A to, there is for node A to B there is path and A to C is a path. So, A to B and then A to C basically here, B to C implement that A to C is there. But it does not mean that B to C is there. Here basically it says that A to B, A to C the path is there. From B for example, there is no path. Here you see from B there is no path. So, it is basically null actually. And from C we will be able to go to B and A.

So, here C, A and B so two nodes are there which represents the successors actually. So, here basically they are representing successors and they are basically predecessor to any successor in this representation. Now, so far that weighted directed graph or weighted undirected graph, we have to add one more field. For example, here each node contains one more field this is basically to store the weight of the edge.

For example, here A to C in this graph G4. This is a representation of G4. A to C as we see there is an edge, so there is a link so that means A is predecessor C is a successor. And then here A to C edge is there and this edge is weight 3. So, this is basically weight 3 and so on, so on. Now, here, for example, another B to A, B to C and B to D. So, here B to A, B to C, B to D it is there.

And weight for each edges are there. And you can note one thing that ordering is not an important. For example, here B to A, B to C and B to D it can be in any order as it is shown here. So, ordering is not an important in this predecessor and successor relations actually. This is few examples that is that basically we discussed about how a graph can be represented using linked list structure. Now, let us come to discussion about another representation which is very fast and very compact again but it has limitations. Those are the other issues of there.

(Refer Slide Time: 24:04)

The slide displays the following matrices:

**G1 Matrix:**

$V_1$	$V_2$	$V_3$	$V_4$	$V_5$	$V_6$	$V_7$
0	1	0	0	0	0	0
1	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0

**G2 Matrix:**

$V_1$	$V_2$	$V_3$	$V_4$	$V_5$	$V_6$	$V_7$
1	0	1	1	0	0	0
0	1	0	1	1	0	0
1	0	0	1	0	0	0
1	1	0	1	1	0	0
0	1	0	0	1	0	0
0	0	1	0	0	1	0
0	0	0	1	0	0	1

**G3 Matrix:**

	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	0	0
C	1	1	0	0	0
D	1	1	1	0	1
E	0	1	0	0	0

**G4 Matrix:**

	A	B	C	D
A	0	0	1	0
B	5	0	1	7
C	2	0	0	4
D	0	6	8	0

Now, in this representation we usually use a 2D matrix to represent a graph. Now this 2D matrix has number of rows and number of columns. So, number of rows is a same as number of vertices or nodes are there. Number of columns is also same as the number of vertices are there. Now for example, in case of G1 number of rows will be  $V_1$  to  $V_7$  and number of columns also will be  $V_1$  to  $V_7$ .

Now, so this is basically (24:30) matrices to represent a graph with N number of vertices. Now, this is the matrix for example, in case of graphs G1 this is basically rows and columns for the graph. Now, entries in the graphs actually indicates that if there is an edge from this node to that node then we can put 1. If there is no edge then it is 0.

Now, here for example,  $V_1$  to  $V_2$  there is an edge. So  $V_1$  to  $V_2$  this is 1. Similarly,  $V_1$  to  $V_3$  also 1, but there is no edge from  $V_1$  to  $V_4$ ,  $V_5$  or  $V_5$ , so we put 0. Rest of the things will be understood like this one. Now this graph G2 is also similar to this graph again but is little bit complex and it is also shown here. So, this is basically, this matrix is called adjacent cemeteries.

That means from a node which are the other nodes are connected that can be obtain from the adjacent cemeteries and each entry represents whether there is a connection or not. So 0 means no connection, 1 means there is a connection. Now, again for the weighted graph also it can be, but in case of weighted graph instead of 0 and 1, we will put weight of each edge.

For example, this is a G4 and this is a graphical representation, representation of graph G4 using matrix and here you can see A to C, there is a path only. But there is no other path so all entries are 0 except C and this at actually 3. Unlike it is 0 and 1 in earlier cases actually. And for the directed

graph also without weight also it is there. If there is edge from  $V_1$  to  $V_2$  it is there.

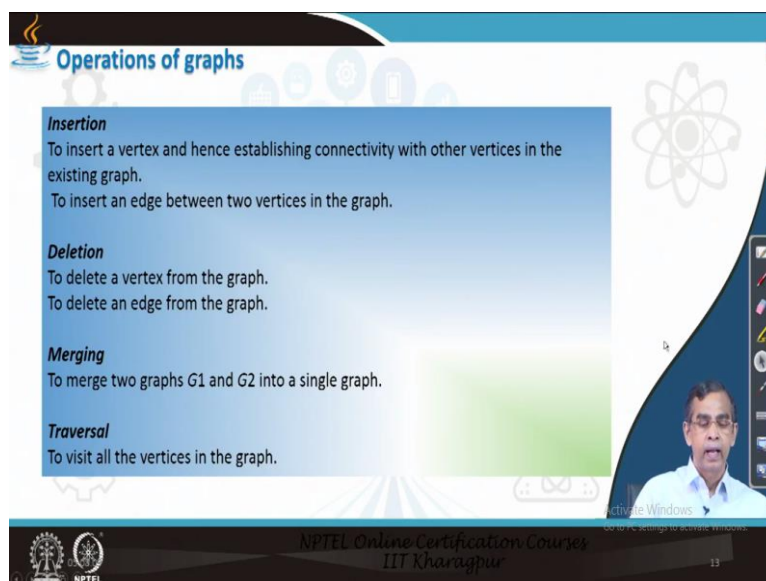
But in case of, in case of a graph if there is no loop, self loop rather, so if then we can see all the diagonal entries are 0. So, that is the concept that you can find it whether there is in a loop. That means loop means from  $V_1$  to  $V_1$  itself or  $V_i$  to  $V_i$  itself whether we can move. In case of theory of graph it is possible.

So selfloop is possible there. Now, whatever we whether loop self or whatever it is there, it is also possible to represent in case of this adjacent cemetric representation as well as in all other representation that we have already studied. So, this is about the representation of the graph with three different structure we have studied.

(Refer Slide Time: 27:05)



The slide features a central title "Operations on Graph" in blue text. To the left is an icon of a coffee cup with steam. To the right is a stylized tree diagram with various icons on its branches. The background is light blue with faint icons of gears, a hard hat, and a molecular structure. A small video inset of a man in a white shirt is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".



The slide is titled "Operations of graphs" in blue text. It lists four operations with their definitions:

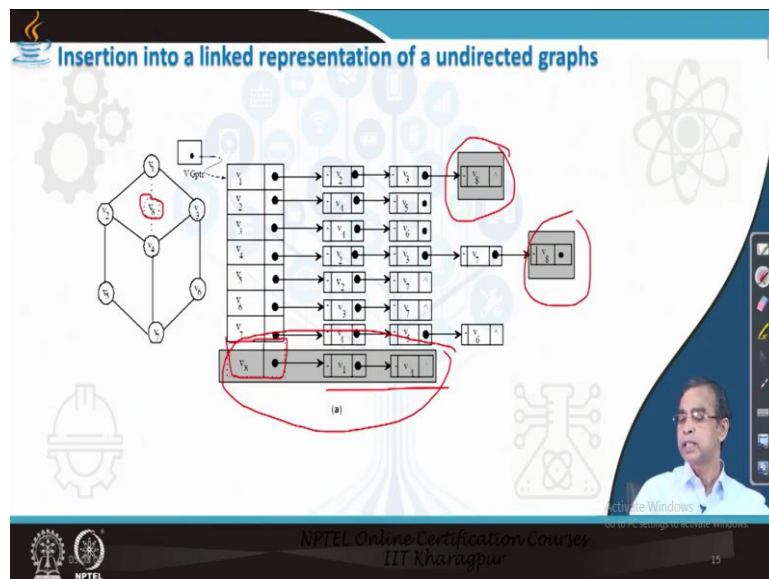
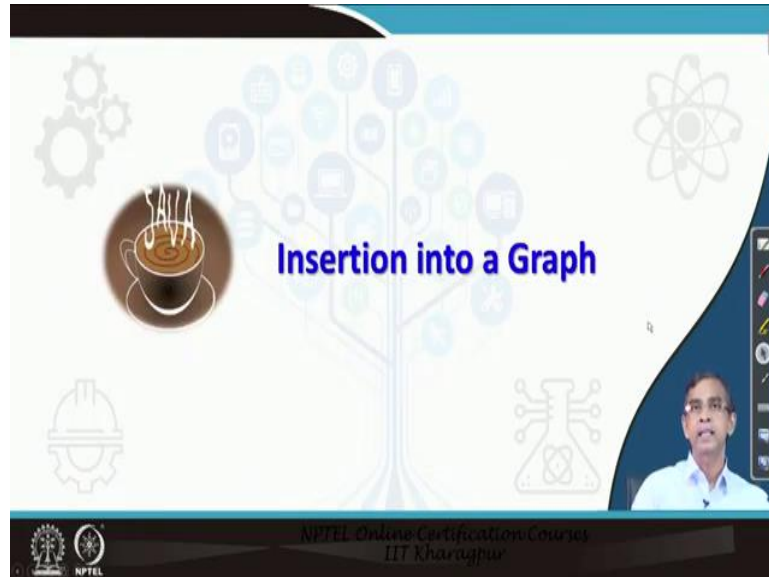
- Insertion**  
To insert a vertex and hence establishing connectivity with other vertices in the existing graph.  
To insert an edge between two vertices in the graph.
- Deletion**  
To delete a vertex from the graph.  
To delete an edge from the graph.
- Merging**  
To merge two graphs  $G_1$  and  $G_2$  into a single graph.
- Traversal**  
To visit all the vertices in the graph.

A small video inset of a man in a white shirt is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

Now let us discuss about few operations on graph and main operation that is possible so far the

graph is concerned is the insertion. That means if you want to add a new node into the graph, if you want to delete a node also. Merging means two graphs can be merged. It is basically insertion and deletion operation put together to merge. And traversal is very important how we can delete all the nodes it is there.

(Refer Slide Time: 27:32)



Now, let us have a quick discussion about all these three operations are there. Now, here is an example just I have mentioned how an insertion operation can be carried out in case of graph if it is represented using link. I will discuss only with link representation. All other representation you will be able to understand by this means and the matrix representation also I will discuss there.

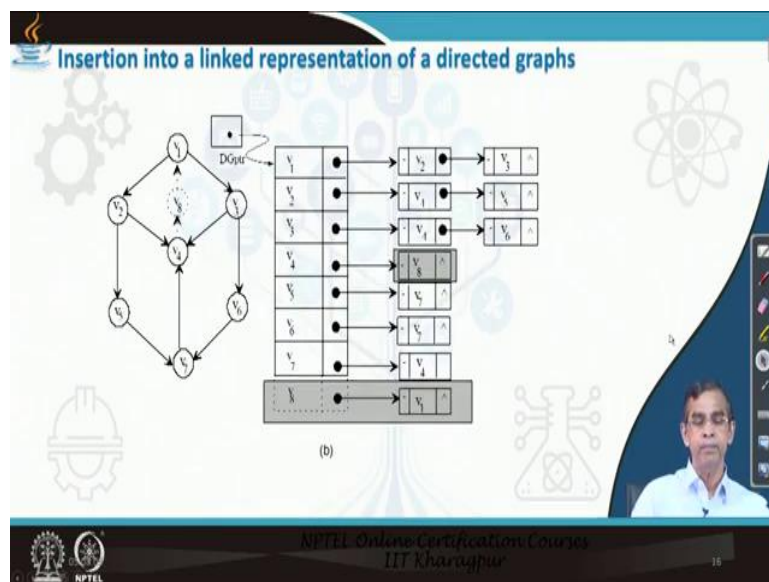
Now, here in this example suppose the original graph is there and we want to add one new node say  $V_8$  and suppose it is a undirected graph. Now, so what we should is that we should add another one point into the pointer array. So, that means we have to extend this array, this one.



Now, so we can add one more because it is new node to be added. Earlier V1 to V7 now V8 then what we have to get, we have to adjust the edges those, because of this addition which will be there. For example, if we add V8 and if there is an edge from V1 to V8, so from V1 pointer we should add this node. And here we see that V1 and V4, these two pointers have to be adjusted because they can be added here. In addition V8 also because V8 should be connected to V1, V4 that needs to be added.

These are the few addition that needs to be done, if we want to add new node into a graph. So this is just like extending previous structure into other as per the requirement. So, this is about the addition, insertion operation.

(Refer Slide Time: 29:04)



Now insertion operation for the directed graph or weighted graph. In case of weighted graph there is one field will be there which is basically not in this representation, whatever it is. Now the directed graph is also there but we have to be little bit careful about that, as the direction implies from V1 to V7 and V7 to V1, not necessary to be add the node V7 everywhere like. Because it is depends on if there is an edge then only from that header to that linked list the nodes will be added here.

For example, here there is no edge from V1 to V8, so there should not be nodes under linked list from V1 pointer to be adjusted. However V4 to V8 it is to be there. So, it is there and V8 to V1 it is there, so it is there. This way we can enforce the directivity among the different nodes if it is added into the graph. So, this is the insertion operation.

(Refer Slide Time: 30:05)

**Insertion into a matrix representation of graphs**

	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	0	1	1	0	0	0
3	1	0	0	1	0	1	0	0
4	0	1	1	0	0	0	1	0
5	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0
7	0	0	0	1	0	0	0	0
8	1	0	0	1	0	0	0	0

NPTEL Online Certification Courses  
IIT Kharagpur

And now let us consider the matrix representation. Matrix representation is easier also. We have to just increase one more dimension in the matrix. For example earlier  $V_1$  to  $V_7$  so this is the, this is the original matrix are there but we want to add another nodes say  $V_8$ , so we have to add one more row and then one more column that is representing the new node  $V_8$  is added here. And then the 0 and 1 entries the adjacent entries needs to be updated accordingly as per their edges. Directed edges also likewise we will be able to add it. So these are the insertion operation.

(Refer Slide Time: 30:46)

**Deletion from a Graph**

NPTEL Online Certification Courses  
IIT Kharagpur



The slide illustrates the deletion of a node from a graph. On the left, a graph with nodes  $V_1$  through  $V_8$  is shown. An arrow labeled  $V_8$  points to a specific node in the graph. In the center, an adjacency list is shown as a table of pointers to nodes. On the right, a matrix representation of the graph is shown. Red circles highlight the deletion of a node and the corresponding updates to the adjacency list and matrix.

(a)

Active Windows

NPTEL Online Certification Courses  
IIT Kharagpur

Deletion operation is very similar to the insertion operation. In case of deletion we have to remove, delete. So, some pointer elements needs to be deleted. Some nodes in the linked list needs to be deleted. Now which node is a removed accordingly and that pointers and that nodes in the all adjacent list needs to be removed. For example, here if the  $V_8$  needs to be removed then all these are the nodes are to be deleted from the nodes. And that deletion from the linked list not necessarily form end or from front. It can be from any positions. And here the pointer also needs to be updated or removed. This is the case of deletion and for the directed graph also similar concept that can be followed. It is the same as earlier.

(Refer Slide Time: 31:35)

The slide features a central graphic of a coffee cup with steam rising from it, set against a background of a tree with various icons on its branches. The title 'Graph Traversals' is prominently displayed in the center.

Graph Traversals

Active Windows

NPTEL Online Certification Courses  
IIT Kharagpur

And then matrix representation is also same. We have to, if we want to remove a particular node corresponding that node the rows and columns needs to be removed from the matrix.

(Refer Slide Time: 31:50)

**Traversals of graphs**

**Depth first traversal**  
Depth first search (DFS) traversal is similar to the inorder traversal of a binary tree. Starting from a given node, this traversal visits all the nodes up to the deepest level and so on..

**Breadth first traversal**  
The breadth first search (BFS) traversal is very similar to the level-by-level traversal of a tree.

Tree

Graph

NPTEL Online Certification Courses  
IIT Kharagpur

Now let us come to the graph Traversals. Graph Traversals idea is that we have to visit all the nodes in the graphs. That is the basic objective in the traversals. Now this traversals can be done in two ways. One is a level by level and another is this is called the breadth first traversals and one is depth, depth weight level is called depth first traversals.

Now let us first see what is depth first traversals. So for example, in this graph we have to start from T1, we have to go there, there and the deepest one. Then from there we have to go there, there and the deepest one. Then go there and then once it is finished then go there, then here and then here and then this one and then this one and then finally. This is basically the way the depth first traversal can be carried out.

(Refer Slide Time: 32:54)

**Traversals of graphs**

**Depth first traversal**  
Depth first search (DFS) traversal is similar to the inorder traversal of a binary tree. Starting from a given node, this traversal visits all the nodes up to the deepest level and so on..

**Breadth first traversal**  
The breadth first search (BFS) traversal is very similar to the level-by-level traversal of a tree.

Tree

Graph

NPTEL Online Certification Courses  
IIT Kharagpur

On the other hand if it is a breadth first traversals. So idea is that we should travel all the nodes in this level fast level by level. Then all nodes in the level, then all the nodes in this level and this level. So, this is the example of breadth first traversals. Now although it is very easy for tree but it is bit difficult for the graph. For example in case of graph which is the starting node or route node. Any node can be taken as a route node. So any node can be taken as a route node. So this is route node. Then breadth first traversal it is like this. So this then and then, then finally this, this.

(Refer Slide Time: 33:28)

The slide is titled "Traversals of graphs" and contains the following text and diagrams:

- Depth first traversal**  
Depth first search (DFS) traversal is similar to the inorder traversal of a binary tree. Starting from a given node, this traversal visits all the nodes up to the deepest level and so on..
- Breadth first traversal**  
The breadth first search (BFS) traversal is very similar to the level-by-level traversal of a tree.

Below the text, there are two diagrams: a "Tree" diagram showing a hierarchical structure of nodes, and a "Graph" diagram showing a network of nodes connected by edges. A small video inset of a speaker is visible in the bottom right corner of the slide.

On the other hand in case of this graph if it is the depth first traversal then we can go this, this, this, this, this then this, this then this, this also. This is the example of depth first traversal that can be from starting from any node. So definitely starting node matters because if this is starting node we can follow different path, same path maybe that depends on direction is there or not there, whatever it is there.

Now, this is graph traversals, there are mainly 2 traversals are possible, the depth first traversals and breadth first traversals. But in order to implement there is an algorithm which we can follow and then given an input graph, it is either in link representation or set representation or matrix representation.

Obviously representation matter because for different representation algorithm will also change. It is not that same algorithm can be applied to the graph, whatever the representation you follow. Algorithm is there and that algorithm you can follow to implement the traversals algorithm.

(Refer Slide Time: 34:31)

**BFS traversal of graphs**

$G_1$   $\xrightarrow{\text{BFS}}$   $G_2$

$\text{BFS}(G_1) = v_1-v_2-v_8-v_3-v_5-v_4-v_6-v_7$        $\text{BFS}(G_2) = v_1-v_2-v_3-v_5-v_4-v_6-v_7-v_8$

**Note:**

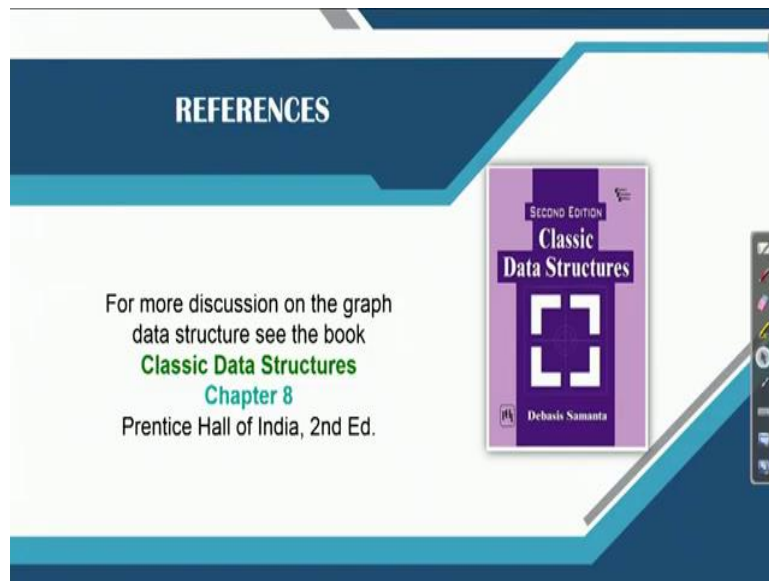
- A **queue** can be used to maintain the track of all paths from any vertex so as to help backtracking.

NPTEL Online Certification Courses  
IIT Kharagpur

But in order to implement all this traversals algorithm there are some additional data structure this is required. For example DFS traversals you need one stack that needs to be, to keep a track that from this point which points has to be traverse and if we traverse it then what is the return point to traverse others. Similarly, for BFS traversals the data structure that you should consider is queue.

So, two data structure matters and then you just simply push some elements into the queue if it needs to be visited and you just pop it if it is visited is over. This way you can keep a track that we should not be loop or we should complete the visit to all nodes this kind of things are there. I have given few examples in this graph where it basically shows how the queue structure can be used and then the BFS traversals can be done. In the previous lecture we have shown how the DFS traverse can be done using stack.

(Refer Slide Time: 35:25)



However the algorithm will be discussed in details later on so that you can think about how those algorithm can be implemented. And this chapter is very exhaustive actually there are many more things had to be discussed but because of the limited time and we are not able to cope with different concepts those are there. But this is the book you can follow here everything is very in details it is discussed.

The algorithm is given, the different representation, many application whatever it is there. This is complete discussion, detail discussion you can find it those are, if you are interested to study graph structure much more in details then I should suggest you to go through this book. So this is a topic so far the graph data structure is concerned and we will discuss something very interesting algorithms related to the graphs in the next lectures. So, thank you, thank you for your attention.