Programming and Data Structure

Dr. P. P. Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

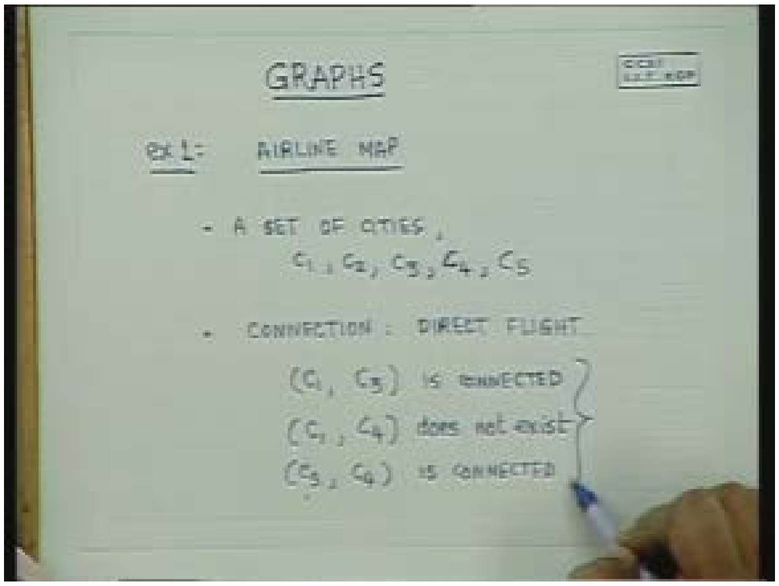
Lecture # 29

Graphs – I

Today we shall discuss the topic of graphs. Graphs are one of the most widely used data structures in computer science and they are used for modeling a variety of problems. So today we shall first see examples where graphs arise then we shall see the structure of graphs and then we shall see how graphs are represented as data structures and what sort of operations or algorithms we would like to implement on graphs. Later on we shall study several algorithms on graphs and these during this study, we shall try and utilize our algorithm design techniques that we have learnt in the previous few classes and combine them with a standard data structure technique to obtain algorithms for solving problems on graphs. So graph is one of the most important structures and today we shall start our study on graphs.

Let us first see examples where graph arise. Consider the first example. Consider an airline map, what does it consist of? It consists of a set of cities, let there be n cities c_1 c_2 c_3 let us say till, let us take only for 5 cities and there is a connection, the connection indicates a direct flight. Therefore suppose there is a connection between city c_1 and say city c_3 that means that there is a direct flight between c_1 and c_3. Now there will not be direct flights between various cities, so there will not be a direct flight between suppose c_1 to c_4 does not exist but c_3 to c_4 is connected, we also know that this is connected. So there is a hopping flight from c_1 to c_4 via c_3.

(Refer Slide Time: 00:03:55)



GRAPHS

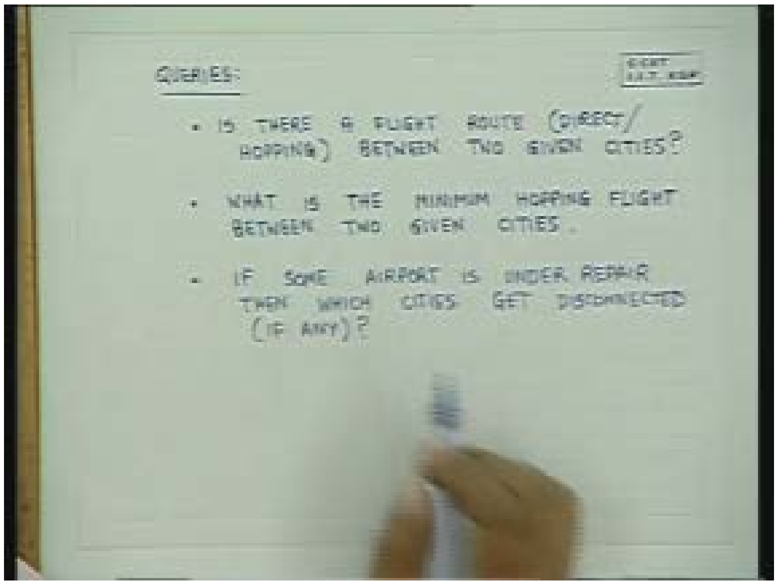
ex 1:- AIRLINE MAP

- A SET OF CITIES,
  c_1, c_2, c_3, c_4, c_5
- CONNECTION : DIRECT FLIGHT
  (c_1, c_3) is connected
  (c_1, c_4) does not exist
  (c_3, c_4) is connected

Now given such a structure where you are given a set of cities and some direct flight connectivities maybe you should be able to answer questions like is there a hopping flight form one city to another, is the cities disconnected or connected. So we could answer various questions. So the queries that we could answer, we could be asked to answer are, is there a flight route direct or hopping between two given cities.

The second question which maybe we could be asked, what is the minimum hop hopping flight between two given cities. The third question is if some airport is under repair that is a fight cannot land then which cities get disconnected in terms of airline flights if any and there are various other questions which could be asked when we traverse between cities. That is you are given an airline system and you are asked whether there are flights between cities, you are asked whether if and city goes down what is there, you are asked to find that minimum hopping flights etc etc.

(Refer Slide Time: 00:06:12)

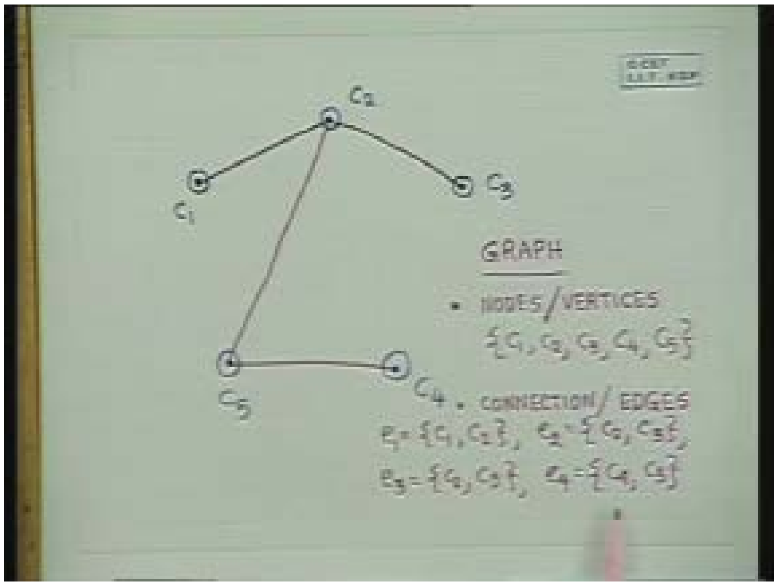


QUERIES:

- IS THERE A FLIGHT ROUTE (DIRECT/HOPPING) BETWEEN TWO GIVEN CITIES?
- WHAT IS THE MINIMUM HOPPING FLIGHT BETWEEN TWO GIVEN CITIES.
- IF SOME AIRPORT IS UNDER REPAIR THEN WHICH CITIES GET DISCONNECTED (IF ANY)?

So how would you model such a problem? You would have to model the cities and you would have to model the connectivities. The natural way to model this cities is in terms of a diagram which we shall see now.

(Refer Slide Time: 00:10:42)


c_2
c_1
c_3
c_5
c_4
GRAPH
• NODES/VERTICES
$\{c_1, c_2, c_3, c_4, c_5\}$
• CONNECTION/EDGES
$e_1 = \{c_1, c_2\}$, $e_2 = \{c_2, c_3\}$,
$e_3 = \{c_2, c_5\}$, $e_4 = \{c_4, c_5\}$

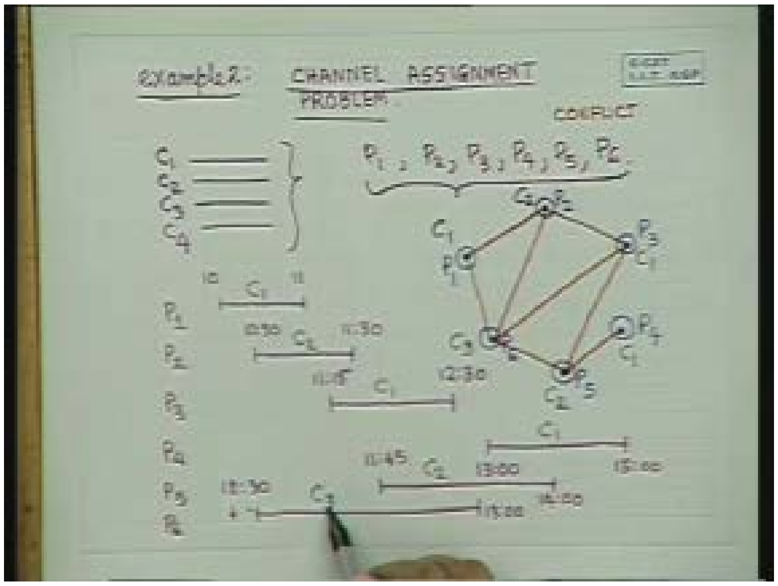
For every city we mark it by a point. Suppose we are conceptualizing it ourselves, so we have 5 cities, we have marked it by 5 points and numbered them c_1 c_2 c_3 c_4 c_5 and then we marked the connectivities. Suppose there is a path between c_1 and c_2, suppose there is of direct flight between c_4 and c_5, suppose there is a direct flight between c_1 and c_3, suppose these are the connectivities which are provided. Now you should be able to answer questions is that a direct flight between c_1 and c_5 that is obvious, we check the connectivity it was not there. We should also be able to answer a question like is there a flight between c_1 and c_3, the answer will be yes. Is there a flight between c_1 and c_4, the answer will be no because there is no flight between c_1 and c_4. That is there is the connectivity is absent between, is clustered between a set of city. You should be able to identify which sets of cities are connected.

So suppose we make it connected. Now there is a flight between every city to every other city, so the property that there is a flight between from every city to every other city will come from the concept of the connectivity of this point and connection diagram. Also we should be able to answer question is if this airport is down then which flights get affected, obviously lot of flights get affected. If this airport is down then only one flight gets affected that is too this airport, none other flights do get affected. So how would we answer these questions, how would we represent this information etc etc. In graph theory, you represent this information in terms of a structure called a graph.

A graph is a natural representation of this connectivity, another graph for these points you have what are called a set of nodes or a set of vertices, nodes or vertices. Here you have, the set consist of c_1 c_2 c_3 c_4 and c_5. You also have a set of connections which in graph theoretic terminology is called a set of edges. And here what are the edges? Every edge is actually a pair of vertices.

Now when we are saying a flight, we mean that there is a flight in both directions, alright or there is a root direct route or if you look at it as a road map then there is a road. So there is a connection between c_1, e_1 is $\{c_1, c_2\}$, e_2 is the next edge which is $\{c_2, c_3\}$, e_3 is the third edge which is $\{c_2, c_5\}$ and e_4 is the fourth edge which is $\{c_4, c_5\}$. So a graphs is a data is a representation of such information where you have got a set of nodes and a set of connections.

(Refer Slide Time: 00:18:10)



Now in order to solve the problems that we have mentioned earlier, we need to represent this information and we need to have good algorithms to implement such a structure. Let us take another example. Previous one which was a road map was a natural phenomena, was a natural representation. We shall see another example where the problem is well modeled by a graph. Let us take a channel assignment problem. Suppose we have got a channel, a channel maybe a radio channel or a telephone lines that is suppose we have got a set of 5 channels.

Channels are c_1 c_2 c_3 and let us say we have got just 4 channel c_4 and we have got a set of programs which are to be aired and the programs can be aired on these. This is the limitation of the cable operator. So the cable operator has got 4 channels and the user requirements are a set of say 10 programs p_1, say we have got to p_3 p_4 p_5. Let us take 6 programs. These are 6 programs which all the subscribers of this cable operator worked and this program p_1 runs from say 10 o clock to 11 o clock. P_2 say runs from 10.30 to 11.30, p_3 runs from say 11.15 to 12.30, p_4 say runs from 13 hours to 15 hours, p_5 maybe runs from 11.45 to 14 hours. Then let us say p_6 runs from 10.30 again right up to 13 hours.

Now we have to find out whether we can air all these programs through these channels of the operator and at least we were on the restriction that the same program must continue on the same channel. So the operator is to assign channels to these programs. So how will we do? So we will see how a graph is a natural way of modeling such a problem.

In a graph the nodes are these channels, at the nodes are these programs p_1 I am sorry p_2 p_3 p_4 p_5 and p_6. And how do we define the edges? We say that two programs cannot run on the same channel if they overlap and then they would be constrained by an edge. So we now put in the edges. P_1 and p_2 overlap in time, so they cannot run on the same channel so that is an edge. This is called a conflict edge. This is called a conflict; they are in conflict so there is an edge, p_2 and p_3 are in conflict, p_1 and p_6 is in conflict. P_1 is not in conflict with anybody else but p_3 and p_6 is in conflict, p_3 and p_5 is in conflict, p_4 and p_5 is in conflict, so this is the conflict.

1 is a conflict with 2 and 6, 2 is in conflict with 1 which is there and 2 is in conflict with 3 and 6, 2 is in conflict with 3 and 6. 3 is in conflict with 5 and 6, 4 is in conflict with 5 and 5 is in conflict with 6. So this is how we get our graph structure. Now what is the solution to the problem? The solution is to assign a channel; we are to assign channels to programs. That is how we have to solve the problem. Now to assign channels to programs, we need to give a channel number. So suppose we give channel number 1 here, suppose we assign channel 1 to program 1. Then we cannot assign channel 1 to program 2 or channel 1 to program 6. That is in graph theoretic terminology; if we assign a channel here the same channel cannot be assigned to another node to which it is connected. So, two nodes which are connected by an edge cannot have the same channel. So we have to assign channel 2 here.

Now here we cannot assign channel 1 or channel 2, so we have to assign some other channel. So let us as assign channel 3 because this cannot assign get channel 1 because it is connected here which has got an assignment, this is cannot take channel 1, this is got an assignment. But this one cannot take 2 or 3 but it can take 1, so we can assign program p_3 to channel 1.

Similarly p_5 can be assigned to channel 2 and p_4 can again be assigned to channel 1. So by giving this assignment, what we do? We run this program in channel 1, this program on channel 2, we run this p_3 on channel 1 again after this one is over. We run p_4 on channel 1 again when this one is over and we run p_5 on channel 2 when this one is over and we run p_6 on channel 3. So by modeling it as a graph, we are now able to model the problem as a channel assignment problem so that this and we have seen that we could solve the problem by 3 channels only.

Therefore given a graph, you want to assign it to a set of numbers in such a way that two nodes which are connected by an edge do not have the same number. This is known as the chromatic numbering problem or the graph coloring problem. And the interesting thing is to find out the minimum number of such values which can be used to consistently assign, to which can be consistently assigned to nodes in this graph. That will give us in this particular problem, the minimum number of channels which the cable operator can use. It is not difficult to see that this problem will require at least 3 channels because of this situation and we have solved in 3 channels. Therefore this obviously must be minimum but finding the minimum is not so easy.

The point which we need, which we wish to stress here is that even complex problems like this which are not direct path problems as we saw in that road map or the airline map can also be modeled in terms of nodes and edges of a graph and the connectivity can model various kinds of things. Now this channel assignment problem or the graph coloring problem which is the general version of this problem has got its use in many situations.

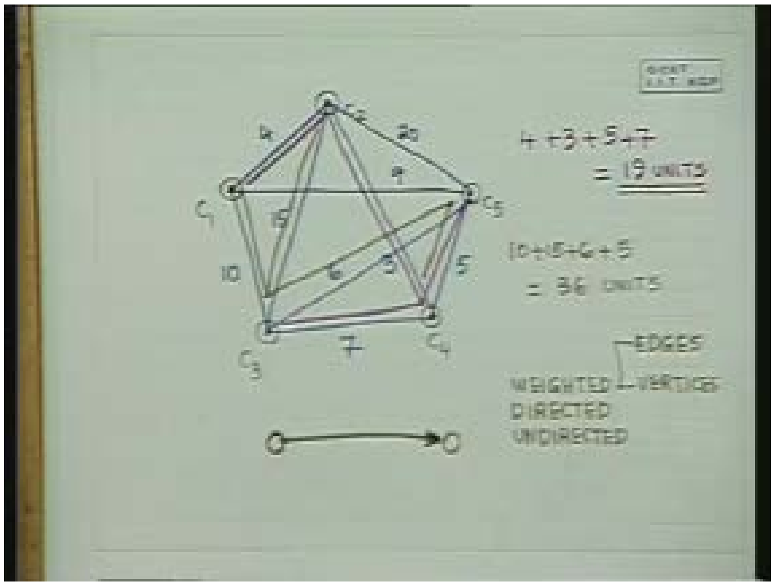
Therefore the issue that we wish to stress here is this that graphs have got one of the most widely used representative or one of the most widely representative structures in computer problem solving and therefore we need to have data structures to efficiently represent graphs and efficiently solved problems in graph, specially problems of connectivity, problems of path finding, problems of finding minimum paths, problems of finding spanning trees etc, etc, etc.

Let us see, let us motivate another problem and see in terms of the road map which is another interesting problem which comes in terms of telephone lines or road maps or other situations. Consider a set of cities and let us model every city as a node in a graph. Now for every city, we have a possible direct connective, a steady connectivity line of a telephone or the same reason, same thing goes for a road map let us take it for a telephone and every thing has got a cost.

So let us try and see what is, how we will do a cable layout problem of minimum cost to link up all the cities in a given country. So let there be cities c_1 c_2 c_3 c_4 c_5 and let us see that what a connection means. Here an edge or a connection means that if I connect a direct telephone connection, if I make a direct cable by satellite or by laying a cable from c_1 to c_2 my cost will be say 4, 4 units. From here to here, my cost will be 10 units, from here to here my cost will be 15 units, from here to here my cost will be 20 units, from here to here my cost will be 3 units, this will be say 5 units, 6 units, 7 units, 9 units.

So this way if I connect everybody to everybody this will be my cost. Now what am I interested in? I am interested in finding out a minimum cost connectivity so that everyone is connected to everyone but the total cost is minimized. That is let us see what are the possibilities. One possibility is this one, this one, this one and this one. So this pink color is a connectivity. Now if I just laid these cables then I am connected, everyone is connected to everybody and my total cost is 4 for this plus 3 for this plus 5 for this plus 7 for this which is equal to 19 units.

(Refer Slide Time: 00:25:21)

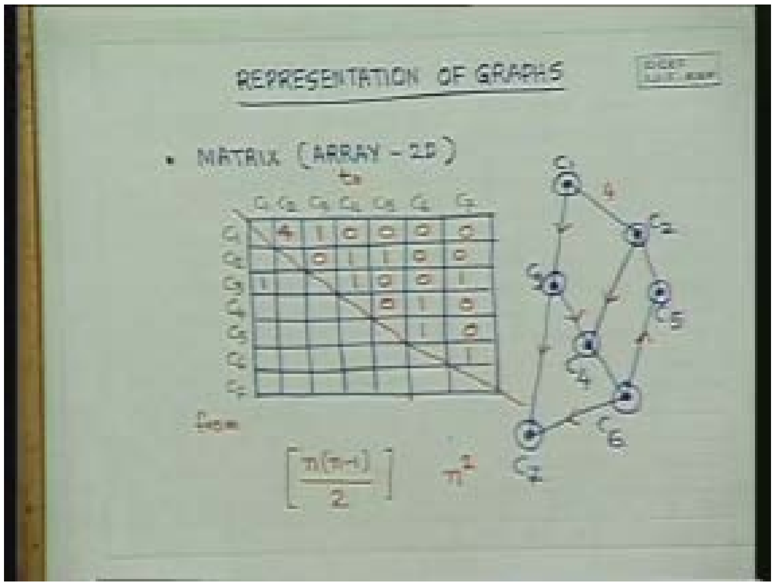


On the other hand if I make some other connectivity say this one, this one, this one and say this one, the brown one again I connect all of them and my cost is 10 plus 15 plus 6 plus 5 and my total cost is 36 units. So this is more desired in terms of cost. That is I would like to connect all of them, so that I get a minimum cost connectivity. So this is another problem which is modeled on graph where the edges now have got a weight.

So we need to model weights on the edges. In some cases vertices may also have weights; in other cases edges may have directions. That is there is a path from this city to this city but there maybe no path from this city to this city, so there maybe a direction. That is there is a valve in a pipeline and therefore there is a direction. So the graph maybe weighted, the graph maybe directed, the graph maybe undirected and the weights can be on the edges of the graph, the weights can be on the vertices of the graph or the weights can be on both.

Therefore we need to first in data structuring, we need to be able to effectively represent graphs first and then try to manipulate graphs. So let us first see how we would represent graphs. The most, the easiest way to represent a graph would be by a matrix which in our terms is an array, a two dimensional array can be used. That is if you are given a connectivity, you represent it by a matrix form where each row represents a city and each column represents the, another, also columns also represent cities and suppose we have a graph which is like this. Suppose this is our graph then we would represent it in c_1 to c_2, c_1 to c_3, c_2 to c_4, c_2 to c_5, c_3 to c_4, c_3 to c_7, c_4 to c_6, c_5 to c_6 and c_6 to c_7 and obviously since its undirected, there is also c_3 to c_1. There are the, either you can look at it only at the top half and say it is symmetric across this diagonal. So this is sufficient to represent your information because if c_1 c_3 is there then c_3 c_1 is also there. So you can look at it this way for getting one information and this way for the other or you can represent it both in both sides.

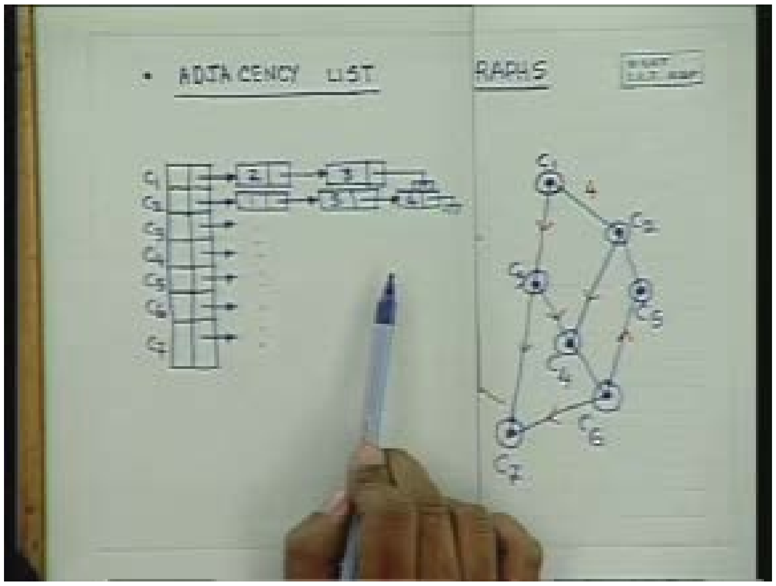
(Refer Slide Time: 00:30:36)



If there are weights on the edges then this value, the other values will all be 0. If there is no connection the values will be 0. For the weights on the edges, suppose you want to represent weights on the edges then instead of connectivity being one it will be represented by the weight. For example if this weight was 4 then c_1 to c_2 would be 4. Directions can be easily implemented by making it this, the from side and this the to side maybe if you have just only these directions then you would represent only these. So matrix or a 2 d array is a useful representation scheme but in a graph all problems usually do not have a very high degree of connectivity that is only very few edges are there compared to the nodes.

For example if there are n nodes then there can be a total of n into n minus 1 by 2 undirected vertices, edges. So these n into n minus 2 undirected edges will be quiet large in compared to, so therefore the matrix will be n square in size, in incase of a directed graph it can be n square whereas the total number of edges maybe very small and we would have to find out only those edges, so scanning these would take more time as we shall see in some algorithms. So the other alternative representation is to maintain with every vertex only the edges to which it is connected by a link list structure.

(Refer Slide Time: 00:33:28)

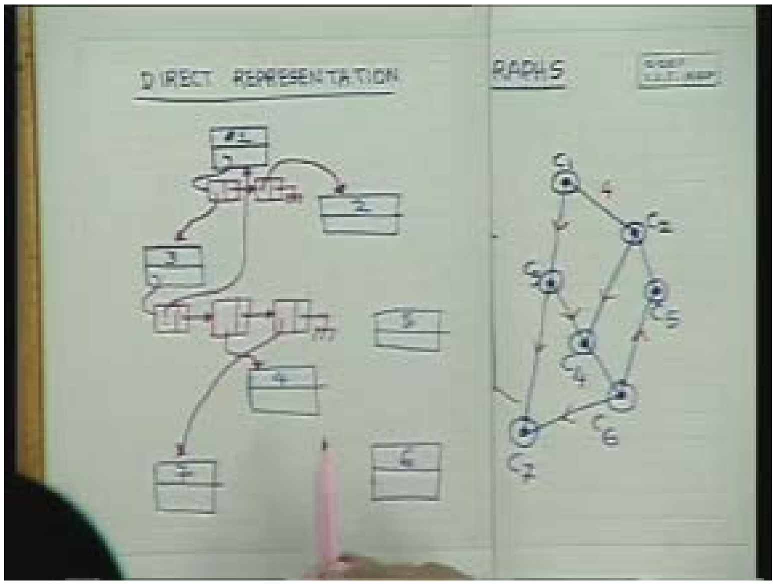


So the next representation scheme is called the adjacency list. The previous one could have been called the adjacency matrix. Now let us see how the adjacency list works out for the problem that we have seen just now. Let us consider this problem where we try to find out the adjacency list structure. Here we maintain a set of header nodes for each vertex c_1 c_2. These are actually array indices, they can be array pointers and they can contain vertex information like weight of a vertex etc etc or any other information or they can just be a set of pointers and they will have pointers to edge information.

For example c_1 is connected to c_2 and c_3 so there will be a link list like this indicating this is connected to 2 and 3. Similarly 2 is connected to 1 5 and 4, so there will be 3 1 5 and 4 and so you can write out all of them. Now here the advantage is if the number of edges is less, you have only maintained the edge list and if you want in many problem it is sufficient to maintain the edge list but in problems where you want to find out which edges are not connected and you want to do that quickly then it would be difficult to do. So because here you would have to scan the edge list, is c_1 connected to c_2? To answer that question and an adjacency matrix you have to just see c_1 c_2 but here you have to scan all of them. So depending on the problem, depending on the edge size you have to have a representation scheme and we shall see as and when we come to problems which representation scheme is more appropriate.

The third and the most general type of scheme for such representation is a general graph structure which is identical to this graph like structure. So the last scheme will be a direct representation. In a direct representation we have, we represent it just as we have this graph. So we have a vertex, for that we have a vertex number say c_1 will have a vertex number 1 and it will have a connectivity to c_2 and c_3. Now this connectivity cannot be an array here because their connectivity is variable, so we maintain the connectivity by a link list. So for every node, we store a node structure just like this graph. So this is a direct representation using complete dynamic allocation where we represent a node just as we represent a graph like this. And how do we represent the connectivity from here to here? For this we maintain what are called link nodes, link edge nodes. So a link edge node either it is grounded like this and this points here and this points here.

(Refer Slide Time: 00:36:30)



Similarly the link edge node for this one will maintain connection to this, to this and to this. So here you will have 3, incase of directed edges, so for undirected graph we maintain both the edges one here, one here and the third here. Similarly you can maintain a connectivity to all of them and this is how you got a direct representation which is often useful in solving many problems. Here again the advantages of adjacency list are there but it is very similar to the structure of an adjacency list, unlike an adjacency list here it is difficult to access the node directly because other way you will have to maintain an array pointed to these nodes. But this has got all the features which are directly available here and represents it here.

Here again the adjacency matrix can directly access a connectivity which this cannot but this can move along the graph in a much more efficient manner than an adjacency matrix. So we shall see various problems, the c structure representations are left as an exercise. Later on we shall see how to solve various problems on such graph structures, namely the problems of whether there is a path, there is a direct path, there is a connectivity etc etc and we shall solve problems like we mentioned in the telephone connection. Can we find, given a complete graph can we find how we will quickly find out the minimum cost, total spanning tree as it is called so that we can connect layout a cable which is of minimum cost.

So we will study all these problems route problems, path problems, tree problems and other design problems on graphs and then we shall come to how we shall represent the graph as an adjacency list, adjacency matrix and we shall see how algorithms on graphs are. And in those algorithms we shall study and we shall see, how we use the concept of recursion, balancing, branch and bound and dynamic programming in a integrated fashion to solve a number of graph problem. So graphs will give us an idea of how to solve problems, how to make data structures and how to solve problems in an integrated fashion. So using the graphs as an application paradigm not only for data structuring but also algorithm design, we shall see several problems in future classes.