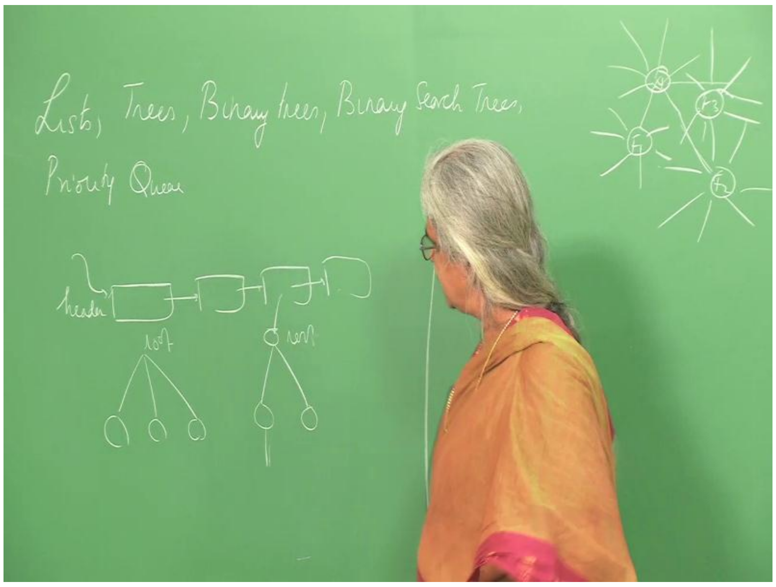
Programming, Data Structures and Algorithms

Prof. Hema Murthy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 15
Lecture -- 52
Hierarchical versus non-bierarchical data structures
Graph:nodes(vertices),edges (arcs)
Directed versus undirected graphs
Paths on graphs
Labeled versus unlabeled graphs
Example of a graph problem:traffc light on a five-point intersection
Representation of a graph:adjacency matrix adjacency list

(Refer Slide Time: 00:15)



The last few lectures we learnt, you know about different types of data structures. We talked about list, then we talked about trees, then we will specifically binary trees, binary search trees and priority queues, this what we are studied. Now, if we look at all these kinds of data structures that we have studied, whether it is a list, it is a tree, binary tree, binary search tree or a priority queue. We always had some kind of a node which had kind of a higher priority and this is a header node, in the case of list which then pointed to other nodes, does not matter whatever the implementation array are we put in hierarchy there.

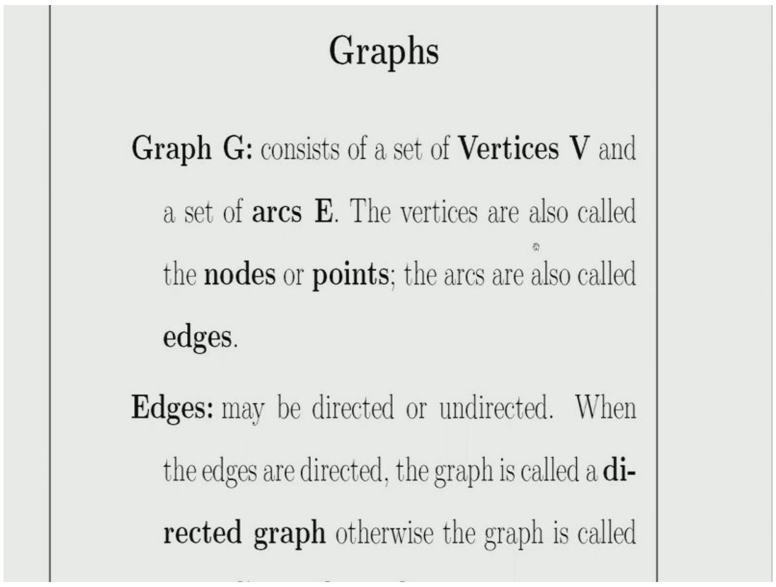
For a tree for example, we said there is a root and then there are children and so on. So, if are a binary tree we had what is called a left child and right child, and we had a node that was designated as the root. Now, let us think about other problem, today with the all of you are on face book I am sure and when you look at face book what do you doing, you have you have a set of friends. And so set of friends watch you, and then you watch somebody else and so on, you are connected to somebody else who is also having a set of friends, you also watch them and so on.

Now, in certain example and this one has perhaps and other set of friends saw the may be disjoined with this respective, let us say this is you, this is your friend 1 friend, 2 and so on. So, what is the interesting is this is called a social network and interestingly this social network has a lot of branches. And in the social network there is nothing like you know in your friends circle. For example, you have some x friends and among is x friend one of your x friends has another set of friends, who disjoined with you is possible.

And it nothing like a hierarchy, in the earlier data structures that we saw, we saw the header known, then we saw the root node and we said access to any other node is why are the root node here, why are the header node in the list it is in all that, here in the other hand there is no hierarchy amongs the nodes. So, you want to represent something like this, this is a very, very common natural phenomena in nature.

But, generally things can be represented such that there are different nodes each one has it is functionality and your connected to each other by some property. Then to represent something like this for example, I may want to find out who are all the friends that are connected to you or I may want to find all the friends are connected to f 1 or f 2 or f 3 for that matter. Then if I was using a tree then I would say if you are the root, then I would say at to access any of your friends after go through only you, then what happens people who are not friends of you I cannot get access to them.

(Refer Slide Time: 03:38)



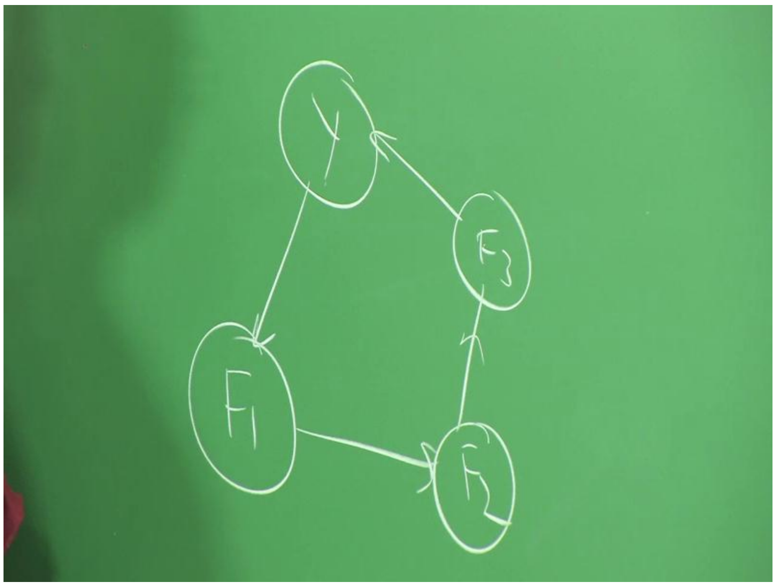
Graphs

Graph G: consists of a set of **Vertices V** and a set of **arcs E**. The vertices are also called the **nodes** or **points**; the arcs are also called **edges**.

Edges: may be directed or undirected. When the edges are directed, the graph is called a **directed graph** otherwise the graph is called

So, another nice type of a data structure is what is called a graph, a graph is essentially consists of a set of vertices. The vertices here are your, you and your friends in the social network and these vertices are connected by arcs, if there is some connection over here. So, I have a set of vertices and if there is a connection between the two of them something you are a friend or whatever it may be, then that is represented by an arc. The vertices are also called nodes or points, there arcs are also called edges. And edges may be for example, you may think for example, that f 1 is your best friend, but f 1 does not think that you are he is or her best friend.

(Refer Slide Time: 04:20)



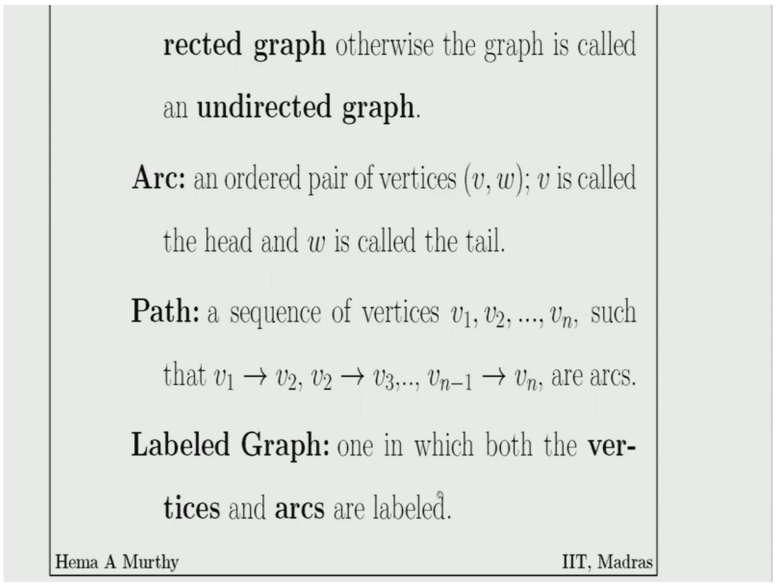
So, in that graph for example, when I am looking edit of over here and it have if I am looking at only best friends you think F 1 is your best friend. But, F 1 thinks F 2 is his best friend and. so on is possible something like this and F 3 perhaps thinks, why is you are his best friend, but you do not feel the same way. Such graphs are what we called directed graphs, can have directed graphs or you can have undirected graph. For example, if I am looking at you know network communication across how do you connect to various websites.

(Refer Slide Time: 04:54)

Then you are sitting at your home with your computer over here, then you are connected to let say if you using BSNL, connected to a BSNL router then this BSNL router goes through number of various links, through the internet and perhaps your server is somewhere come over here. And normally these links are what we called bidirectional links, and then we say that this is a undirected graph, both the graphs are important in practice we like to analyze both of these graphs.

So, basically what is that defines whether a graph is directed or undirected, the edges may be directed or undirected, when the adjust are directed the graph is called a directed graph, otherwise the graph is called an undirected graph.

(Refer Slide Time: 05:54)



rected graph otherwise the graph is called an **undirected graph**.

Arc: an ordered pair of vertices (v, w); v is called the head and w is called the tail.

Path: a sequence of vertices $v_1, v_2, ..., v_n$, such that $v_1 \rightarrow v_2$, $v_2 \rightarrow v_3$,.., $v_{n-1} \rightarrow v_n$, are arcs.

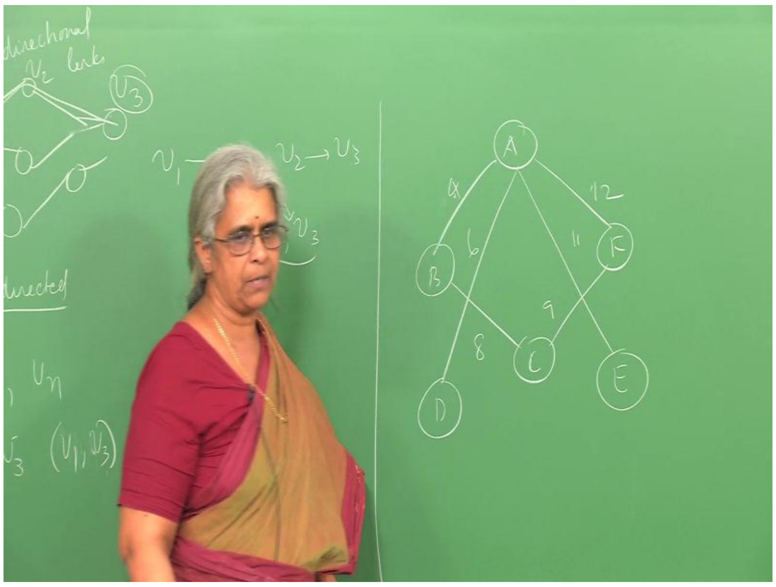
Labeled Graph: one in which both the **vertices** and **arcs** are labeled.

Hema A Murthy IIT, Madras

How do you define an arc or an edge, we represented as an ordered pair of vertices. See the example over here, if there are n vertices in the graph v 1 to v n, if there is an edge between v 1 to v 3. Let us say, we represented by a symbol like this or if you writing it you put it is an ordered pair u comma v, v 1 comma v 2 in this particular example there is an edge between v 1 and v 3 you write it as an ordered pair like this. Now, then you also have what are called.

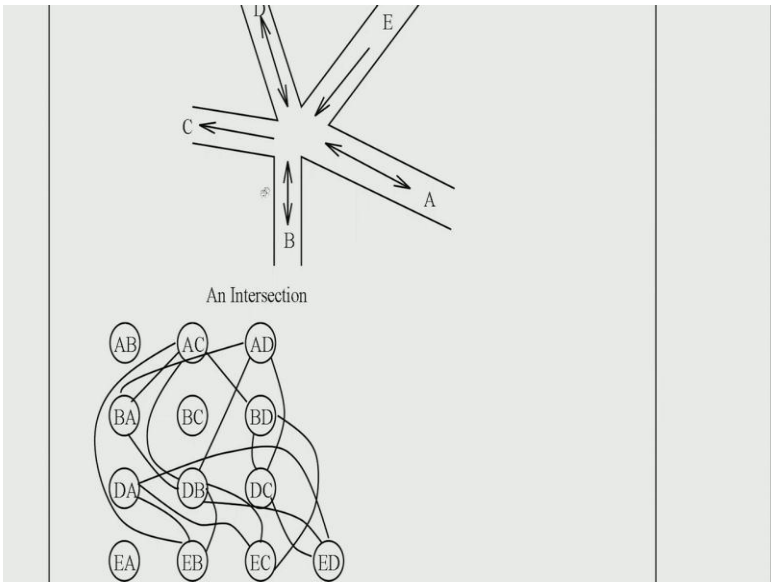
So, if you want to defined a path, how do I defined a path if I want to find a path from here to here, then let us say this is node v 1, v 2, v 3, then I would say my path is from v 1 to v 2 to and v 2 to v 3 is my path. So, this corresponds to path, so you can have a set of sequence of vertices which gives you the path. So, at piece what am I saying here, I can represented this is v 1, v 2, v 3 corresponds to a path; that means, there is an edge from v 1 to v 2, v 2 to v 3 and that represents the path starting from here to reach this particular node. Now, you can have both labeled and unlabeled graphs, but labeled graphs are what we are going to look at.

(Refer Slide Time: 07:26)



So, when I look at a labeled graph I have something like this vertices are labeled here. So, here let us say and if have weights on this 4, 6, 8, 9, so both edges and nodes can be labeled the nodes and there are edges it can be labeled, what is this weight's mean to give you an idea, if you want to let say travel from one place to another. And then this weight could correspond to the amount of time, it takes speed to reach from point A to point B for that matter. So, this is what a labeled graphs.

(Refer Slide Time: 08:19)



Now, I want to just leave you with one example, all the way we will not we discussing this. Graph problems can be found in various applications here is a very, very interesting example, let us see this there is you know here is a graph, where I want to this is a 5 point intersection, there are 5 lines which are intersecting over here. There one road is called A and another road is called B, C, D and E, C and E are one way, A B D are bidirectional.

Now, I want it let us say you know my job is to design a traffic signal for this, I want a put a traffic signal such that no accidents happen, if you follow the signaling rules. So, how do I go I defining it, first thing let us assume that this is a India and all left hands are free for the timing, we will assume that therefore, B to C is permitted. Similarly, A to B is permitted and E to A is permitted and D to D to is of course, is a right turn as not permitted with out of signal.

So, we will assume what will we do is, I wanted in just see what I have done here. I have converted this problem of designing at traffic signal for this. Such that, I have converted it to a graph problem, what have I done over here, I have barely a numerated all the possible parts that exist, there is I can go from D to B, I can go from A to C, I can go from E to C and go from A to D, we can go from D to A, D to B and so on. So, whatever

I do at look at all the parts that all the roads that can be reached from A, from A I can go to B from A I can go to C and A I can go to D, so at put make the most nodes.

Similarly, from B what are the parts that I can take, I can go from B to A, I can go from B to C and I can also go from B to D. So, there are three parts again similarly I do it for that s 2 also D to I have D to A, D to B and D to C and E to A from E to I have E to A, E to B, E to C and E to D all the four parts are possible. So, basically what are the nodes the nodes are this graph correspond to the roads, that the parts that are possible. If I am traveling by on this road and if I am taking a vehicle which are all the legal parts that are allowed.

Now, my job is to design a traffic signal, then what I do is I join a pair of nodes by an edge, if both of them cannot happens simultaneously. So, let us look at this, let us look at A to D and E D and let us say A D and let say that I am looking at A C A D and D B. So, this A D and this is D B remember is a left and right left hand it traffic. So, this is going this way and this is coming this way, so both of them cannot happen simultaneously.

So, what I do is I connect here this may be a little incomplete, but what I have done is, I connected all the nodes in this graph, which cannot happen simultaneously by an edge. Then what happens is this becomes what is called, now as I said I want to define a traffic signal. So, what is a problem of traffic signal now, basically it is a question of colors, what are the colors that you are familiar with you have green, yellow and red or green and red for the timing let us say that.

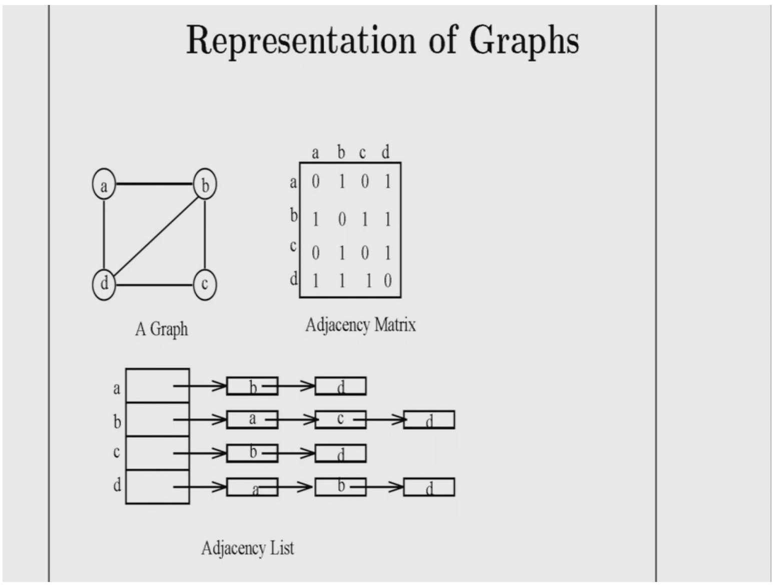
So, what this means is that when I am doing defining colors for this two parts which intersect cannot have green at the same time. So, what I essentially do is I try to find out how many different colors are required to design this traffic signal. How do I find the number of colors. So, what happens is it is very interesting that designing the traffic signal for this intersection, simply becomes a problem of finding a edges in the graph.

And ensuring that two nodes which are adjusting to each other that is as two nodes which are connected by an edge do not have the same color. So, first for example, so a in this example if you see A B C E is simply does not matter at all, they you can do not have draw any color for that. Because, you do not have any edges, they completely free left hands AC if I color as green then B A and B D cannot have the same color.

So, you do this kind of then you basically if I given then you find out what are all the colors that A C can go with, what are all the turns that A and C can happen together. For example, if I can go from A to C, and I can of course, make a D C this not I can make a free left turn from B to C. So, these two do not intersect, so it need not have to have the same color. So, basically what you do is, you look at this particular graph start off with coloring, this put a give a different color to this.

And similarly keep doing this and till you have satisfied all of them. So, if you find the until all the nodes in the graph are colored. So, this is just an example coloring problem is a difficult problem. In fact, finding the minimum number of colors is not a simple solution, but this is just to be illustrate that any there are you start finding graph kind of problems, even for something like this you want to think about it, if an intersection that you want to design and for this intersection you can convert into graph problem and then it simply biased on to question of not giving to adjacent nodes the same color.

(Refer Slide Time: 14:49)



Representation of Graphs

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 1 & 0 & 1 & 1 \\ c & 0 & 1 & 0 & 1 \\ d & 1 & 1 & 1 & 0 \end{array}$$

Developing the algorithm for this problem is a little tough, it is a little beyond is scope of this course, but you just give you an idea. Now, how do you represent graphs, now graphs are can be represented by matrices here is a graph and what it tells me is, whenever be this is what is called an adjacent c matrix. This is a two dimensional matrix, where you have what is this on the y axis and also on the x axis and if between a pair of nodes there is a edge you mark it as one otherwise 0.

If it has this is an un weighted graph, if it is a weighted graph that if it has numbers on the edges when you put that number over here. Then other representation called the adjacent c list representation, the adjacent c list representation for every node it is an array of nodes and at every node you have a link list, which says to which node it is connected to. For example, a has edges to b and d and b has edges to a c d and so on put all of for every node you give the...

Since, this is an undirected graph we also show that d has edges to a b and d, because both the forward and the backward edges are present that is what we assume into b. So, these are two different representations of graphs. So, now we have I think I have motivated you sufficiently to look at graph problems, and what we will do is in the next 1 or 2 lectures, we will look at some different problems and graphs. And so we will stop with this introduction on graphs now.