Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-8

Lecture 26: Basics of Graph Theory

So, now we are going to be shifting gears a little bit and also look at some I mean in basics of graph theory because eventually when we are going to be establishing the network in the context of distributed optimization that is when graph theory like tools from graph theory would be very useful ok. So, why do we care about basics of graph theory? So, a typical distributed optimization problem is of this form. So, let us say we want to solve the centralized problem and for now let us also assume that the problem is unconstrained. So, we want to minimize this function f of x and this f of x is basically nothing, but So, it is made up of several decomposable functions f i and you can think that there is a I mean there are different agents or different entities in the network which are essentially trying to minimize the sum of their objective function. This is and a particular example can be when we try and let us say when you train a neural network right you try and minimize the loss function over your own data sample right. So, f i of x essentially in that case or let me not use f at x here.

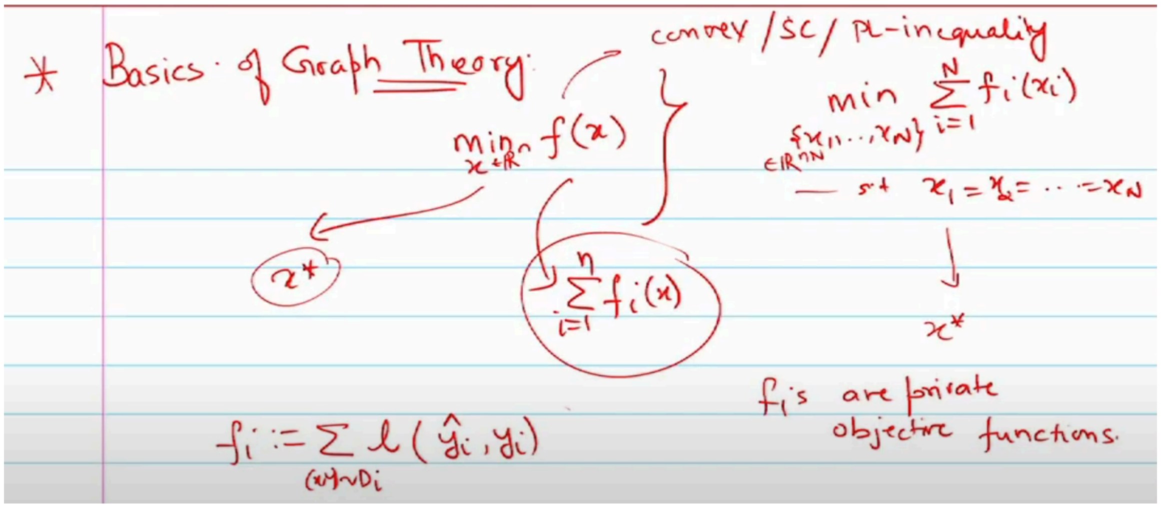
So, f i for instance can be defined as when you say I want to minimize the mismatch between the predicted values and the true values for the ith agent and this is this needs to be done over all the data points that ith agent. So, for all x y sample from data set d i. right. So, even though it is the same objective function analytically, but because it is evaluated on different points, every agent has its own fi right.

And if you want to have a centralized update of the neural network, then ideally what you want is you want to compute the gradient of summation of fi. right before you can update the weights of the neural network. So, this kind of objective function is pretty common minimize the sum of and we are going to largely focus on functions that that either that are either convex strongly convex or basically for these f i's or function that satisfy PL inequality something that we have already looked at. So, the objective is to minimize the sum of these objective functions right where every individual entity has got its own objective function But the challenge is I do not know anything about your dataset or basically your own objective function. So, I can find basically an unconstrained minimizer of my own objective, but that may be very different from what you end up finding on your own data.

And then what is a good value? I mean a good value is something that basically works for all the agents and not just for me or just for you. So, there must be some kind of consensus between the solution that I end up finding and you end up finding or anyone or for that matter anyone in the network ends up finding right. So, for what we are basically trying to say is there will be one x star that basically minimizes this. However, this cannot be achieved because f i is not known to everyone. So, we look at an equivalent problem of this form minimize So, we introduce, so every agent has its own primal variable now x i.

So, we minimize with respect to x 1, x 2 all the way x n subject to x 1 equal to x 2. It is the same optimization problem right, because of this particular constraint. Now the centralized optimization problem, we can write this as sort of decentralized optimization problem. So, everyone has its own optimal variable. So, instead of working with an x which is in let us say R n, we now have n copies of capital N copies of x i's, all of them are in going to be R n.

So, everyone has its own copy of what the optimal solution would look like. I am going to be minimizing my own objective function, but at the same time I am also going to keep a track on what you are doing, you as my neighbor is basically arriving at and that will give me an idea about how everyone else is evolving. And that way you can try and arrive at this x star. So, the goal is eventually all these x i's will converge to x star. And if that happens then we are trying to solve the same optimization problem in a rather distributed fashion.


* Basics of Graph Theory:
convex / SC / PL-inequality
$\min_{x \in \mathbb{R}^n} f(x)$
x^*
$\sum_{i=1}^{n} f_i(x)$
$\min_{\{x_1,\dots,x_N\} \in \mathbb{R}^{nN}} \sum_{i=1}^{N} f_i(x_i)$
s.t. $x_1 = x_2 = \dots = x_N$
x^*
f_i's are private objective functions.
$f_i := \sum_{(x,y) \sim D_i} \ell(\hat{y}_i, y_i)$

Distributed because this computation is going to be distributed at each edge. So, you can exchange certain information. So, for instance you can exchange your current x i with your neighbors or you can maybe exchange the gradient of your f i with your neighbors at any iteration, but you have your own private objective function. So, f i's are private.

Yeah. So, as I said like you have your own data to work with right. So, you do not want to reveal your private data to any of your like neighbors that way. Yeah, it has to be something has to be conveyed otherwise you cannot achieve that. But then that means I know about your. No, but then it is like I have a computer, you have a computer, I have my own data set, you have your own data set.

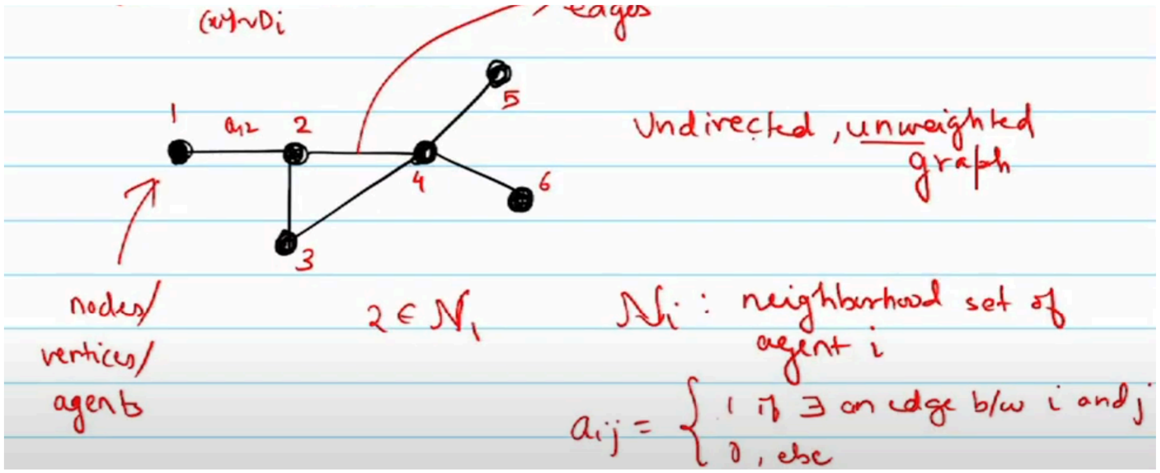
So, we are doing computations. It is a same dimensional vector and dimensional vector that we have both of us are working with. You are computing gradient on your data set, I am computing gradient on my data set and we are just going to be exchanging what the gradients may look like or let us say what the if I were to use my gradient to update the neural network, what do my weights look like and your weight like what you are going to be arriving at and we are exchanging that kind of information. So, it is not like I am going to be computing for everyone else's in the like everyone else's we have on the network right. I am just going to be exchange like computing on my data like gradient on my data set and just exchanging the information with the neighbors.

So, the additional overhead that you get is in terms of communication exchange right and, but that has to be there even with the ADMM or the dual decomposition we had this information exchange either with the centralized aggregator or either I mean or with the neighboring agents right. So, that information sharing has to be there, but it would not be directly about the f i's. So, the kind of setup that you have is usually. So, in this context we are going to study graphs and typically this is one particular schematic. Let us say this is something like this.

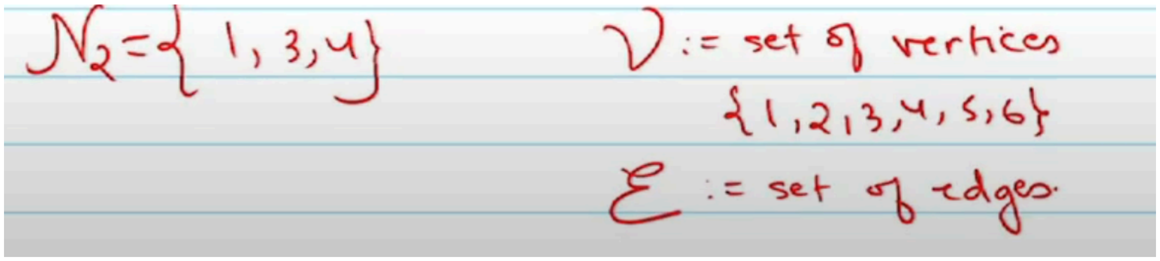
So, these black entries are called. So, nodes. So, we are going to be using either nodes, vertices, agents. So, all these terms we are going to be using interchangeably. So, that and in this.

So, let us let us number these right 1, 2, 3, 4. So, 2 is a neighbor of 1 ok. So 2 lies in the neighborhood set of 1. So the neighborhood set, so n sub i, this cursive n sub i, this is the neighborhood set of agent i. And what are these quantities here? These are edges right.

edges or connections between different nodes. And our presence of edge would indicate that in this case for instance agent 1 can exchange information with agent 2 and vice versa, but 1 cannot for there is no edge between 1 and 3. So, 1 cannot exchange information with 3 ok. So, that is the idea behind placing these edges. So, edges indicate connectivity between different nodes.


edges
1
a_{12}
2
3
4
5
6
Undirected, unweighted graph
nodes/vertices/agents
$2 \in \mathcal{N}_1$
$\mathcal{N}_i$: neighborhood set of agent i
$a_{ij} = \begin{cases} 1 \text{ if } \exists \text{ an edge b/w } i \text{ and } j \\ 0, \text{ else} \end{cases}$

This is an example of undirected graph, but I mean for most of this course we are going to be focusing on undirected graphs, but you will I mean there are results which also extend for directed and weighted graphs and things like that. So, what is an undirected graph? So, when there is no specific directionality between 1 and 2. So, for instance, so this is an example of undirected and weighted graphs. So, I will tell you what these terms represent. So, undirected because there is no directed edge bit from 1 and 2.


$\mathcal{N}_2 = \{1, 3, 4\}$
$\mathcal{V}$:= set of vertices
$\{1, 2, 3, 4, 5, 6\}$
$\mathcal{E}$:= set of edges.

So, it is not like 1 can exchange information with 2, but 2 cannot exchange it with 1. So, had there been such kind of directionality, then it would have become directed graphs. unweighted because I mean so we in some sense we are just going to indicate the presence and the absence of the edges between the nodes, but sometimes we also want to indicate the weights of these edges. So, if you remember the first example about the temperature measurement that we looked at right. So, every agent updated its value based on the averaging of all other agents in the network.

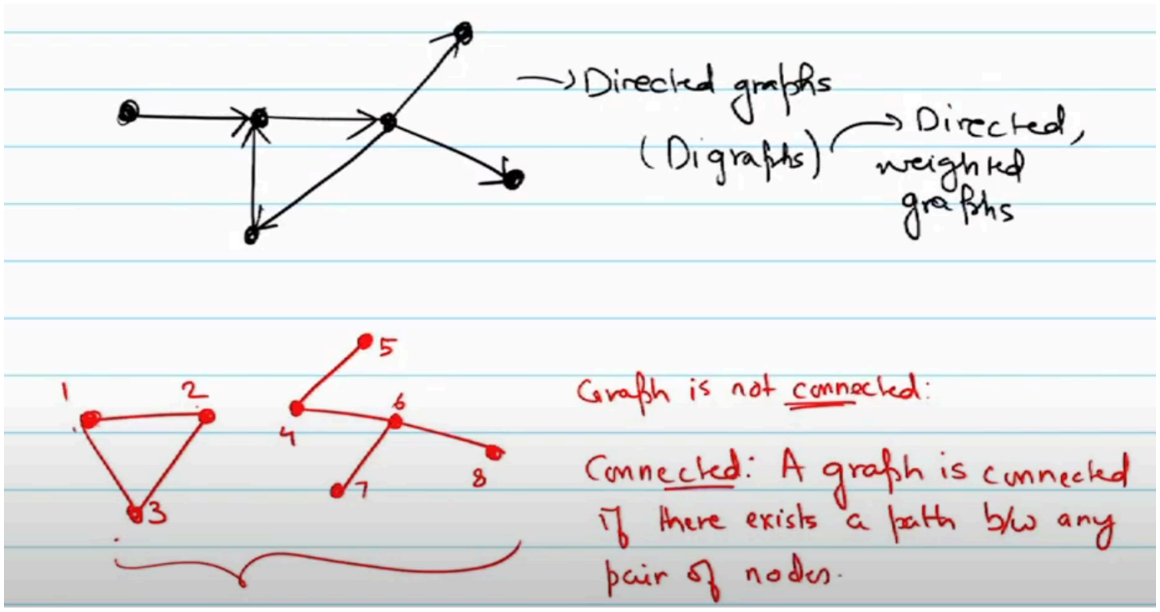
So, in that case agent 1 for instance is going to place a weight to this particular edge and using that weight it is going to receive its information right. So, there can be let us call it a 1 2 ok. So, a 1 2 I mean if it is unweighted graph we just say that a 1 2 is equal to 1 if there is an edge it is 0 otherwise right. So, aij is going to be 1 if there exists an edge between i and j and its 0 else. But sometimes you can have certain weights of these edges and in that case aij would have would be a number let us say 0.

5, 0.3 things like that. So, even if there is an edge you are not going to be assuming it to be just binary kind of quantity, but there is also an edge. So, in that case you would say that it is an undirected weighted graph. So, the set of vertices we are going to be denoted we are going to be denoting this by this capital calligraphic V. So, this is the set of vertices which in this case is 1, 2, 3, 4, 5, 6 and all the set of edges we are going to be denoting this by this capital V. Is everyone ok with the notations? So, n sub i again is the neighborhood set of agent i.

So, what is the neighborhood set of agent 2? In this example 1, 3 and 4 right ok. So, that means 2 can exchange information with 1, 3 and 4 and so on. So, as I said I mean there is also you can have some in certain cases you can have directionality between these edges. So, if I consider the same example, but this time we also have certain kind of directionality. So, you can you can assume that 1 can exchange information with 2, but then 2 would not be able to exchange information with 1 and so on.

So, these kind of graphs are called directed graphs. And they basically indicate the flow of information and the flow of information is not bidirectional, it is going to be in one particular direction. And if you also have weights on top of these edges, so it is called directed weighted graphs or there is a better term called digraphs. which is basically for directed weighted graphs. But for the interest of this course we are just going to be largely we are going to be focusing on undirected graphs.

Most of the results that are there for undirected graphs with slight bit of modification you can extend this to directed graphs. So, I think it is much easier to understand it through undirected and weighted graphs in most cases. So, in this case, so what is the difference between? So, let us say if I have a graph which looks something like this. So, what can you conclude about this kind of graph? This is one graph by the way.


→ Directed graphs
(Digraphs) → Directed, weighted graphs
1
2
3
4
5
6
7
8
Graph is not connected:
Connected: A graph is connected if there exists a path b/w any pair of nodes.

So the graph is not connected. So what do you mean by connected? So what do you mean by connected? So there must exist a path from one node to any other node. So basically I can reach from 1 to 2, I can reach from 1 to 3, I can reach from 2 to 3, but I cannot reach from 1 to 4. So, a graph is connected if there exists a path between any pair of nodes. So, is this graph connected the one over here that is connected right you can reach from one node to any other node. So, this graph is connected while this one is not ok this graph is not connected ok.

So, why is connectedness important in let us say in the context of distributed optimization why do you think connectedness is important. So, when we are trying to exchange let us say our estimates of what this x star looks like. So, agent 1 would have its own estimate x 1, 2 would have its own estimate x 2 and x 3 and so on. But we would not be able to know what the other part of the network is basically arriving at if there is no connection. So, connectedness is important in most cases at least I mean in some form connectedness is important.

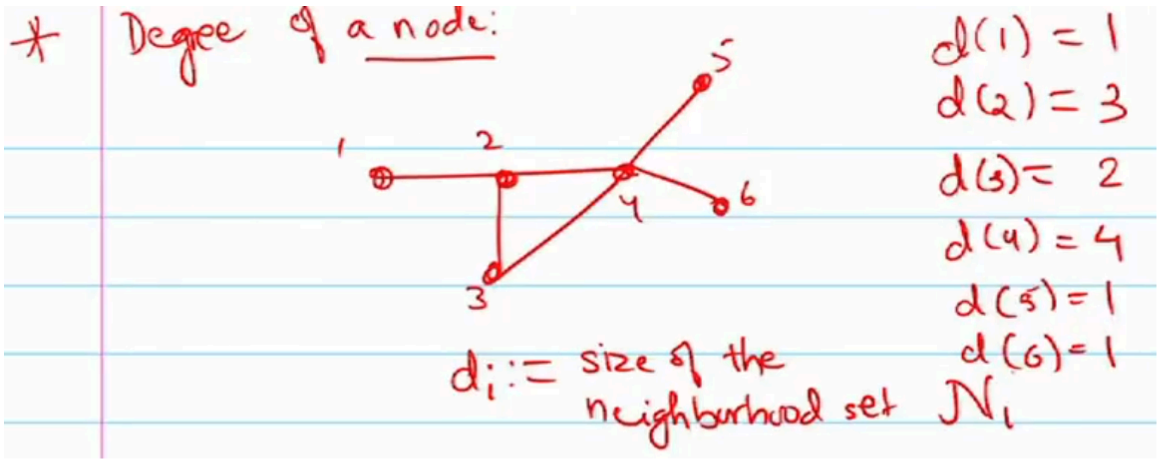
It may not always be connected at all times, but like let us say on an average over a period of let us say 10 time points, there should exist a path for you to be able to go from one node to another. So, you need some kind of connectedness to be able to exchange information within the entire network that is one thing. So, these individual subgraphs. So, this is these are called subgraphs by the way and these subgraphs are called connected components of a graph ok. So, in this case, there are two connected components in this graph.

If the graph is connected, then there is just one connected component. What about this graph? Is this graph connected? So, is this graph connected? Let me also number these so that it is also clear. is this graph connected why or why not yeah you cannot go from two to one for instance So, in the directed setting the definition of connectedness has to go beyond just having an edge right. It is in fact about having a directed edge. So, there should always exist a path for you to be able to go from point a node like node i to a node j and if it is possible for every pair i j then it is a connected.

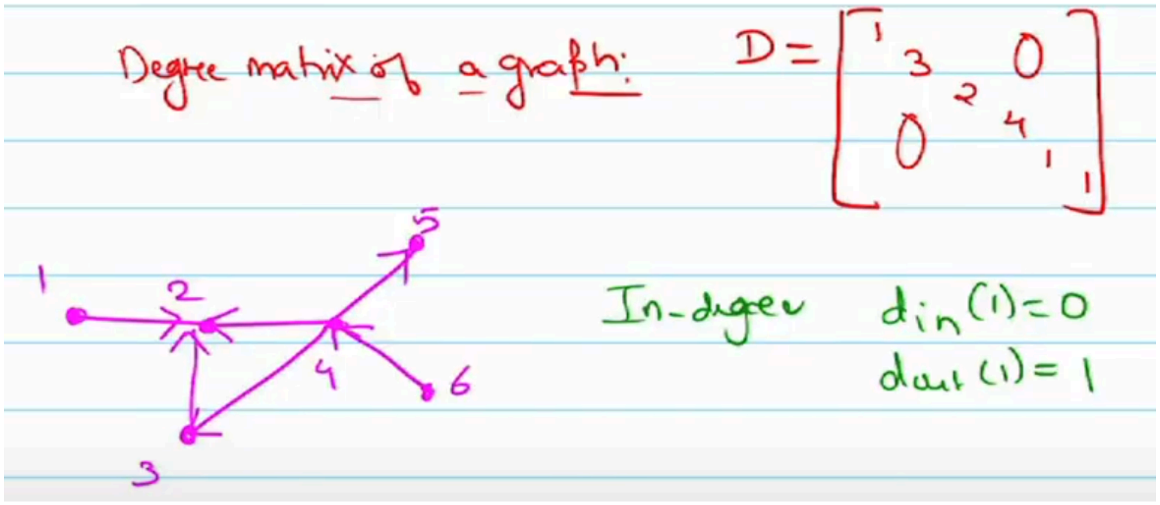
So, in the context of directed graphs we define something called strongly connectedness or strong connectedness. So, not just not just that the entire the underlying undirected graph needs to be connected. I mean the connectivity should be there through these directions. So, we call it a strongly connected. So, you can reach from any node to any other node using these directed edges.

I mean, we do not even talk about the connectedness in that case, we just talk about strongly connectedness of the strong connectedness. This is not here. So, this graph is not this graph. is not strong ok. So, the next thing is next important concept is degree of a node.

So, again if I consider the same example. So, what is degree of a node in a graph? Yeah, number of connections are basically size of the neighborhood set. So, degree of the first node is let us say 1. So, d sub i is basically size of the neighborhood set. d 2 is 3, d 3 is 2. So, based on this degree we can also define something called degree matrix of a graph, which is basically a diagonal matrix with the diagonal entries being d y.



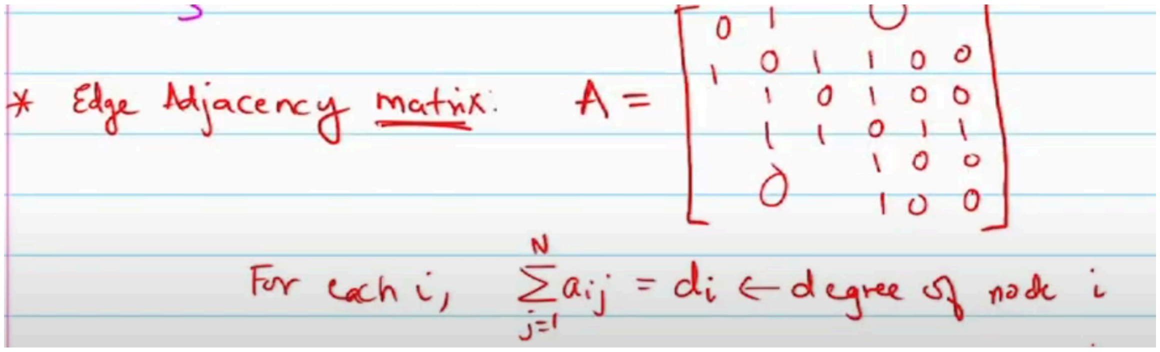
So, degree matrix of a graph, so in this case capital D is going to be 1, 3, 2, 4, 1, 1 across the diagonals and everywhere else 0. It is a diagonal matrix of the degrees of the nodes. How do we define degree for, now let us say we have a directed graph. what is degree of 1? So, how do we define degree of a directed graph? Yeah, so we in fact have a notion of in degree and out degree.



So, in degree and out degree. So, in degree would be number of incoming connections and out degree would be number of outgoing connections from a particular node. So, d in 1 in this case is 0. but d out 1 is essentially 1 and so on. So, then we have this notion of edge adjacency matrix. So, what is the adjacency matrix of a graph? So, basically it denotes the connection between any two nodes and in case of this undirected graph for instance 1 is connected to 2 right.

So, first of all there are no self edges. So, that means all the diagonal entries are going to be 0. So, 1 is connected to 2 and likewise 2 is also connected to 1 right. So, you can see for undirected graph this adjacency matrix is going to be symmetric ok. So, 1 is connected to 2, 2 is connected to 3, basically 1, 3 and 4, 1, 3 and 4, then 3 is connected to essentially 2, 4, 4 is connected to 2, 3, 5 and 6, 4 is connected to 2, 5 and 6 and 5 is connected to 4 and 6 is also connected to 4 everywhere else at 0 and this is your this is your adjacency matrix ok.

So, for each i. So, we are going to be denoting as I said we have this thing a i j right a i j is going to be. So, every element of this adjacency matrix is a i j. So, for each i what is what is this quantity? Yeah.


* Edge Adjacency matrix:
A =
For each i, $\sum_{j=1}^{N} a_{ij} = d_i \leftarrow$ degree of node i

So, d sub i right. So, it is basically degree. So, the row sum in fact because this matrix is symmetric the row sum or the column sum both are in fact the degree of the degree of that node. So, this is the degree of node. Thank you.