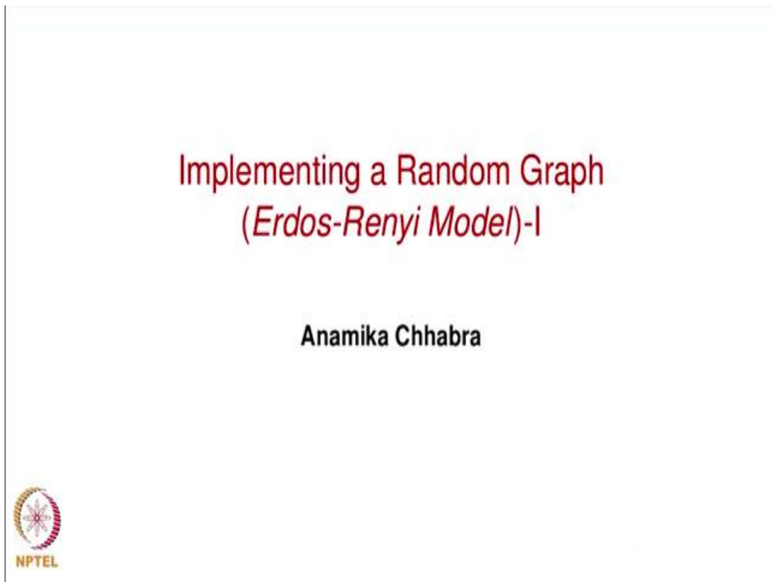
Social Network
Prof. S. R. S. Iyengar
Prof. Anamika Chhabra
Department of Computer Science
Indian Institute of Technology, Ropar

Rich Get Richer Phenomenon
Lecture - 123
Implementing a Random Graph (Erdos– Renyi Model) - 1

(Refer Slide Time: 00:05)



Implementing a Random Graph
(*Erdos-Renyi Model*)-I

Anamika Chhabra

Hey everyone, in the previous video we implemented Barabasi-Albert model which follows preferential attachment that is also called rich getting richer phenomena. In this video sequence we are going to implement Erdos Renyi model which is used to create random networks. In this model each edge has a fixed probability of being present or absent independent of the other edges.

The idea is that we will start with the network with some n nodes with no edges amongst them. Now there are going to be ${}^nC_2$ pairs of nodes in the network. We have to decide whether which whether an edge should be added between the nodes of each of these pairs or not.

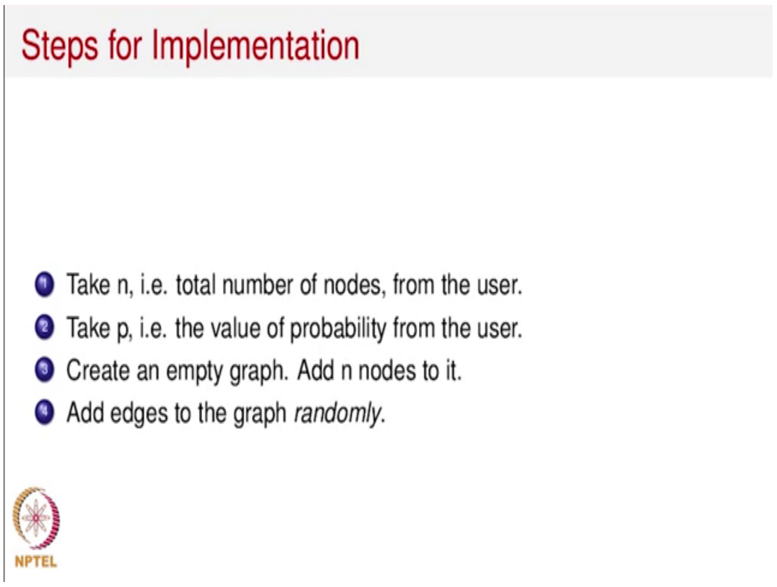
Now, let me explain that method to you using example. So, we will take the first pair of nodes and we will take a coin we will toss it. If we get a head, we will put an edge between this pair of nodes, if we get a tail, we will not put an edge between this pair of nodes. Then will take the next pair of nodes and we will toss the coin and we will do the same.

This we will keep doing for each pair of nodes in the network which is ${}^{n}C_2$. So, in the end we would have added some number of edges in the network. To generalize and to understand how it actually happens in the network in the model, we basically take some value of p. So, when we toss a coin which is not biased p is equal to 0.5 that is you get the head and tail with equal probability.

However, in the model we take the value of p from the user or we fix it this value, this can be any value between 0 and 1. So, for example, assume p = 0.3. In that case if you if you correlate with the example that I gave, the coin will be biased with the probability 0.3 that is head will appear with probability 0.3 and tail will appear with probability 0.7 ok.

So, that's the whole idea and you can imagine that if the value of p is high, the more number of edges will be added to the network and if the value of p is less than less number of edges will be added in the network, if value of p = 0 no edges will get added. If the value of p = 1, then all the edges will be added to the network and it will be a complete graph. So, that is the whole idea behind this Erdos Renyi model. Let us look at this steps that we are going to follow for the implementation.

(Refer Slide Time: 02:47)



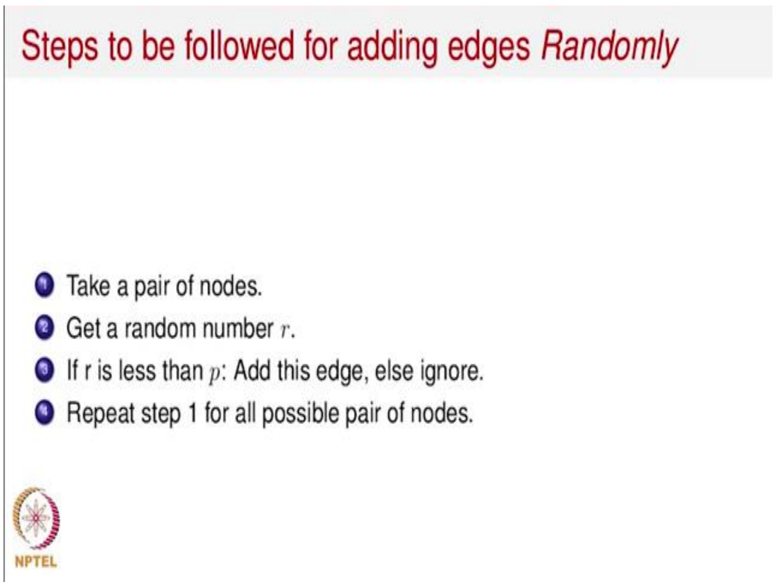
Steps for Implementation

1. Take n, i.e. total number of nodes, from the user.
2. Take p, i.e. the value of probability from the user.
3. Create an empty graph. Add n nodes to it.
4. Add edges to the graph *randomly*.

So, there are 2 parameters that are required for implementing this model that is n and p, where n is the total number of nodes in the network and p is the value of probability as I told you, this will be input to the model and based on that only the edges will be added. So, we are going to take value of n from the user and then we will be taking the value of p from the user.

So, initially the network has only the nodes. So, we will take an empty graph and we will add n nodes to it and there are no edges in the network as of now. So, edges will be added to the network randomly Now, let me explain you how we are going to randomly add the edges.

(Refer Slide Time: 03:34)



Steps to be followed for adding edges *Randomly*

1. Take a pair of nodes.
2. Get a random number r.
3. If r is less than p: Add this edge, else ignore.
4. Repeat step 1 for all possible pair of nodes.

So, we are going to take a pair of nodes ok. Basically we have to process all the pairs of nodes which is ${}^nC_2$ pairs, we will take the first pair and we will get a random number r between 0 and 1. If the value of r <= p we will add an edge between this pair of nodes and if value of r > p we will not add an edge between this pair of nodes and we will check the next one. We will keep repeating this step 1; step 1 2 3 actually ok.

We will keep repeating the first 3 steps for all possible pair of nodes and by the end after we have processed all the pairs, we would have added certain number of edges to the network. So, this is the whole idea that we are going to follow for the implementation in the next video and we will also see the degree distribution that this kind of network follows.