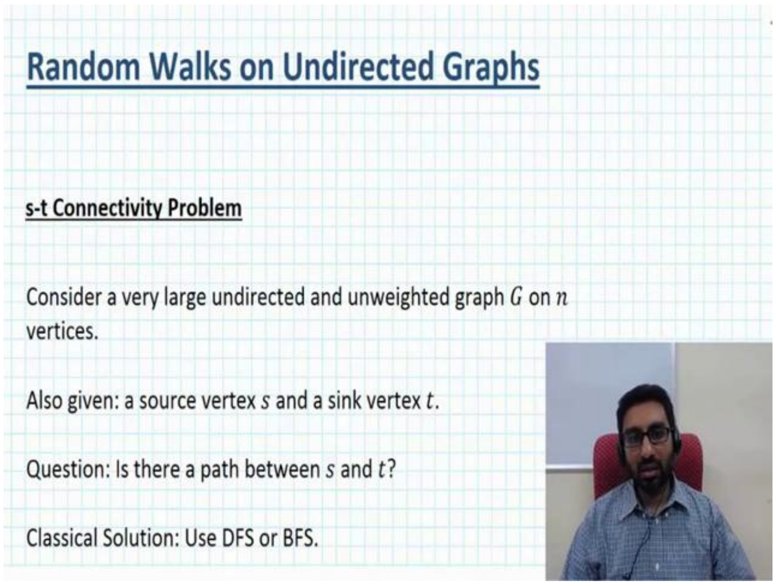
Algorithms for Big Data

Prof. John Ebenezer Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 19
Random Walks on Undirected Graphs

(Refer Slide Time: 00:15)



Random Walks on Undirected Graphs

s-t Connectivity Problem

Consider a very large undirected and unweighted graph G on n vertices.

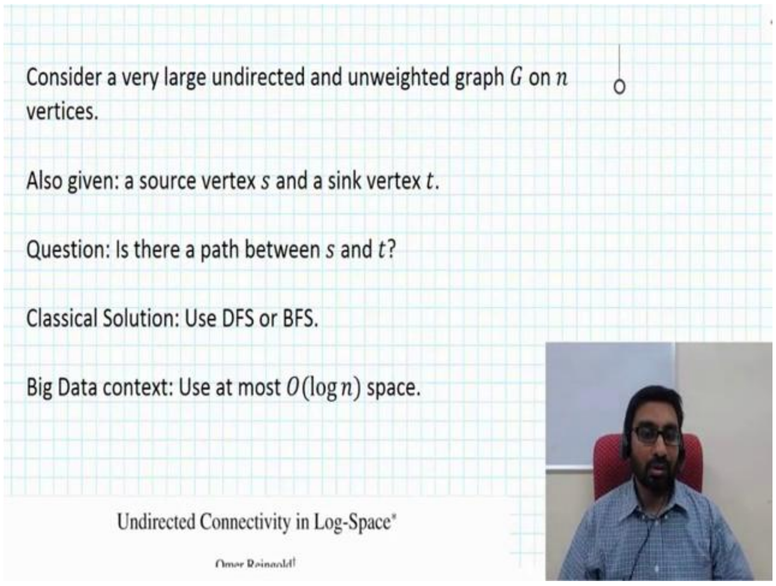
Also given: a source vertex s and a sink vertex t.

Question: Is there a path between s and t?

Classical Solution: Use DFS or BFS.

In this last segment, we are going to talk about random walks on undirected graphs. And to motivate this problem, we are going to talk about the s-t connectivity problem.

(Refer Slide Time: 00:29)



Consider a very large undirected and unweighted graph G on n vertices.

Also given: a source vertex s and a sink vertex t.

Question: Is there a path between s and t?

Classical Solution: Use DFS or BFS.

Big Data context: Use at most $O(\log n)$ space.

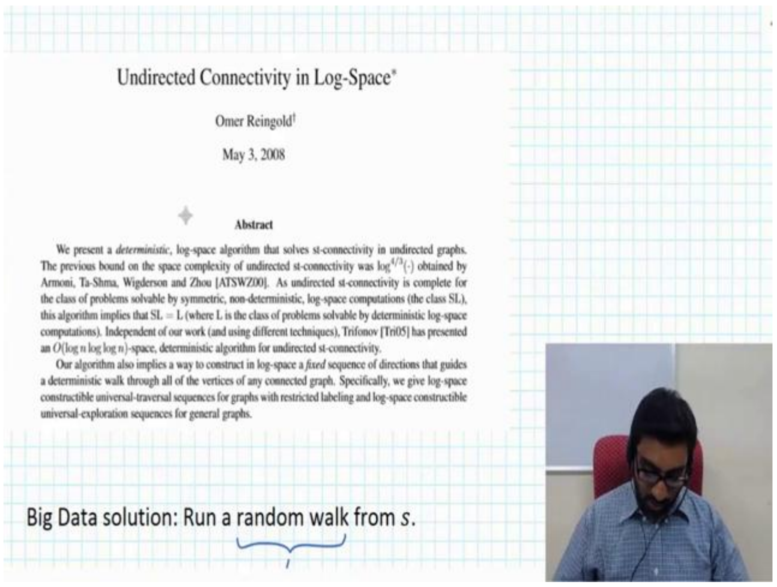
Undirected Connectivity in Log-Space*

Omer Reingold†

And this problem consider a very large graph G, it is undirected and unweighted; this graph has N vertices N is large. So, the classic big data problem, it also has lot of interesting implication of square, but let us focus on the big data aspect. You given a source vertex s and also sink vertex t.

Now, we do not know this graph is connected and particular we do not know whether there is a path between s and t and that is the question we are interested in. We want to know there is a path between s and t. This is not very difficult. We walked into through some fall of algorithms course, and you studied BFS DFS, so it is easy to solve, but the big data context comes from the fact that you are not allowed to use lot of space. Now if you were to use BFS, you will need to have a queue of up to theta of N size. Similarly, even if it is DFS, you will need to have a stack of theta of n size, and either of these is unacceptable. And therefore, we need a solution that takes a very little space.

(Refer Slide Time: 01:51)



Undirected Connectivity in Log-Space*

Omer Reingold†

May 3, 2008

Abstract

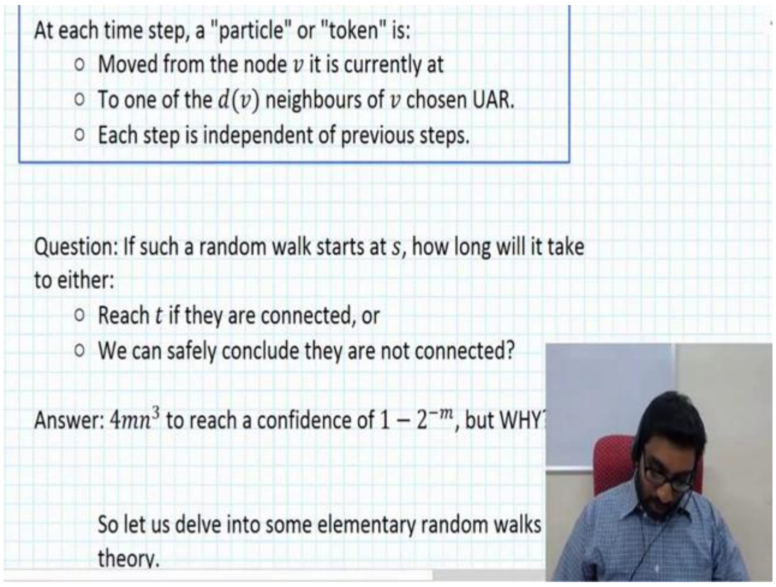
We present a *deterministic*, log-space algorithm that solves st-connectivity in undirected graphs. The previous bound on the space complexity of undirected st-connectivity was $\log^{4/3}(\cdot)$ obtained by Armoni, Ta-Shma, Wigderson and Zhou [ATSWZ00]. As undirected st-connectivity is complete for the class of problems solvable by symmetric, non-deterministic, log-space computations (the class SL), this algorithm implies that SL = L (where L is the class of problems solvable by deterministic log-space computations). Independent of our work (and using different techniques), Trifonov [Tri05] has presented an $O(\log n \log\log n)$-space, deterministic algorithm for undirected st-connectivity.

Our algorithm also implies a way to construct in log-space a *fixed* sequence of directions that guides a deterministic walk through all of the vertices of any connected graph. Specifically, we give log-space constructible universal-traversal sequences for graphs with restricted labeling and log-space constructible universal-exploration sequences for general graphs.

Big Data solution: Run a $\underbrace{\text{random walk}}$ from s.

And as I looked into this problem has a very interesting complexly theory implications. In particular, the question of whether a symmetric, non-deterministic log space computation equals deterministic log space computation. This was a big open question that got result in mid 2000 (Refer Time: 02:14) to Omer Reingold. So, this is a very famous paper, but this talks about a deterministic solution and a sort of very practical solution to implemented in real world application. And so, we are more interested in a solution that is easy to implement easy to analyze and easy to appreciate.

(Refer Slide Time: 02:38)



At each time step, a "particle" or "token" is:

- Moved from the node v it is currently at
- To one of the $d(v)$ neighbours of v chosen UAR.
- Each step is independent of previous steps.

Question: If such a random walk starts at s, how long will it take to either:

- Reach t if they are connected, or
- We can safely conclude they are not connected?

Answer: $4mn^3$ to reach a confidence of $1 - 2^{-m}$, but WHY

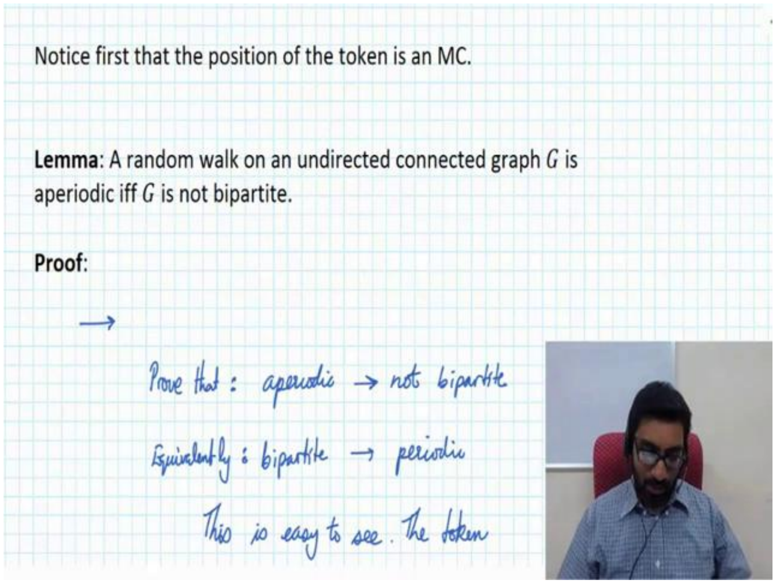
So let us delve into some elementary random walks theory.

So, we are interested in a random walk space solution. What is a random walk? At each time step, think about a particle or token that is at some vertex, at each time step, this token takes random walk step. That means is from current location, let us say let us call that v, it chooses one of the d v neighbours of v uniformly at random, and moves to that neighbor. And each step is independent of previous steps that it took. So, it only depends on where its current location is and that will and from there one of the neighbors chosen uniformly random. And so, what we do is simple. We start a random walk from s and we allow it to walk for some number of steps, and hope that it will reach t.

So, of course, if it reaches t then; that means, s and t are connected. But if it is does not reach t, that mean s and t are not connected; it might just have be the case that it this random walk just somehow avoided t. And so, we need to be concern about that we need to understand, what will happen. And from your previous lectures, where we talked about finite irreducible and ergodic Markov Chain, you probably have some that it would not miss some part of the maze.

But the problem is how long does it take to actually cover the entire maze that is important, because that tells you that if you walk that long and still do not see t then s and t probably are not connected, so that is the intuition that we are going to work with. So, we want to find that amount of time, which that random walk should walk before we can safely conclude, whether s and t are connected or not. And it is turns out that number of steps is 4 m n cubed in order to get a confidence of 1 minus 2 raise to the minus m, but why is that well that is what we are going to delve into in this segment.

(Refer Slide Time: 05:15)



Notice first that the position of the token is an MC.

Lemma: A random walk on an undirected connected graph G is aperiodic iff G is not bipartite.

Proof:

$\longrightarrow$

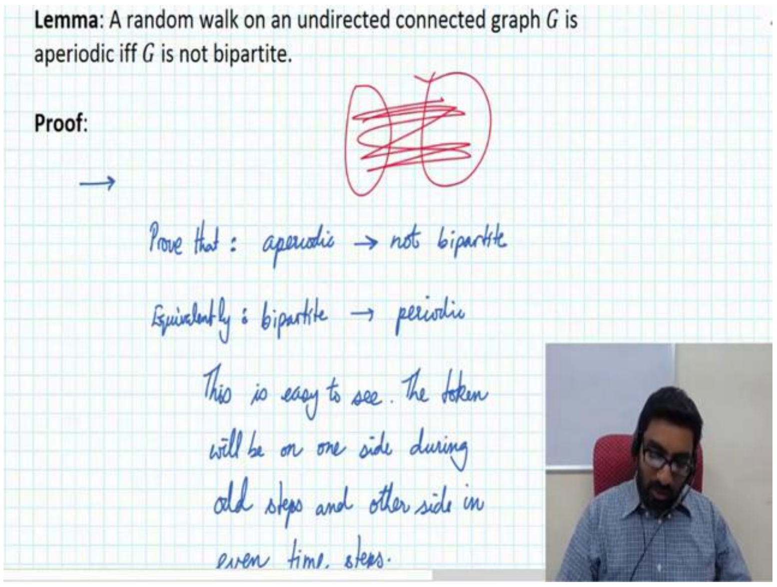
Prove that : aperiodic $\rightarrow$ not bipartite

Equivalently : bipartite $\rightarrow$ periodic

This is easy to see. The token

And just to incase you missed this point, notice that the position of the token can be thought of a Markov chain. So, the position is a vertex. So, the Markov chain has N states basically called each states corresponding to some vertex in which that token is present. And let us start delving into these notional random walks. Let us look at the first lemma here. A random walk on an undirected connected graph G is aperiodic, if and only if G is not bipartite. So, in other words, the graphs are not bipartite or more interesting in if you care for a periodicity.

(Refer Slide Time: 06:10)



Lemma: A random walk on an undirected connected graph G is aperiodic iff G is not bipartite.

Proof:

$\rightarrow$

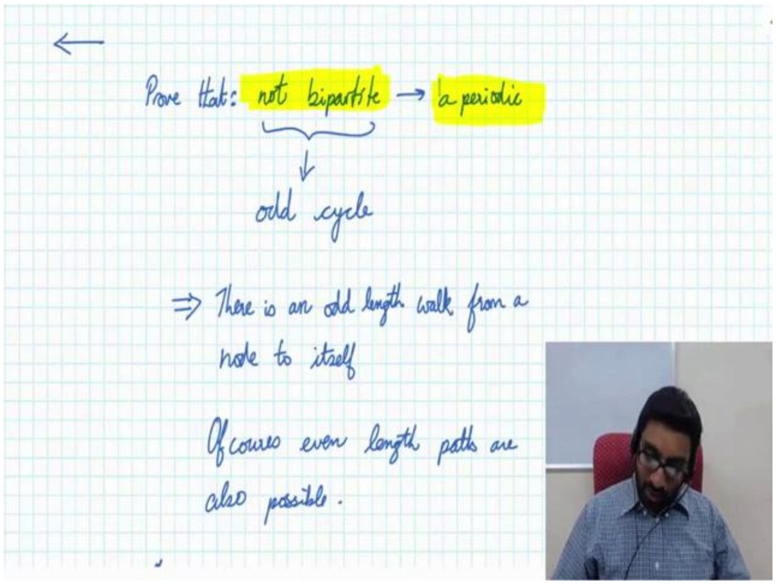
Prove that : aperiodic $\rightarrow$ not bipartite

Equivalently : bipartite $\rightarrow$ periodic

This is easy to see. The token will be on one side during odd steps and other side in even time steps.

Let us see how this lemma can be proved, so this is if and only if statement. So, the proof as gone two directions, the first direction that we are going to prove is aperiodic implies not bipartite or equivalently let us look at the (Refer Time: 06:31) bipartite implies periodic. This is very easy to see, because if it is bipartite then what happens is there are two parts to the graph and the random walk will stay at one side during odd times and other side during even times. So, it is quite easy to see that bipartite graph is going to random walk bipartite graph is going to be periodic. So, the forward direction is proved that way.

(Refer Slide Time: 07:12)



$\leftarrow$

Prove that: not bipartite $\rightarrow$ a periodic

$\downarrow$

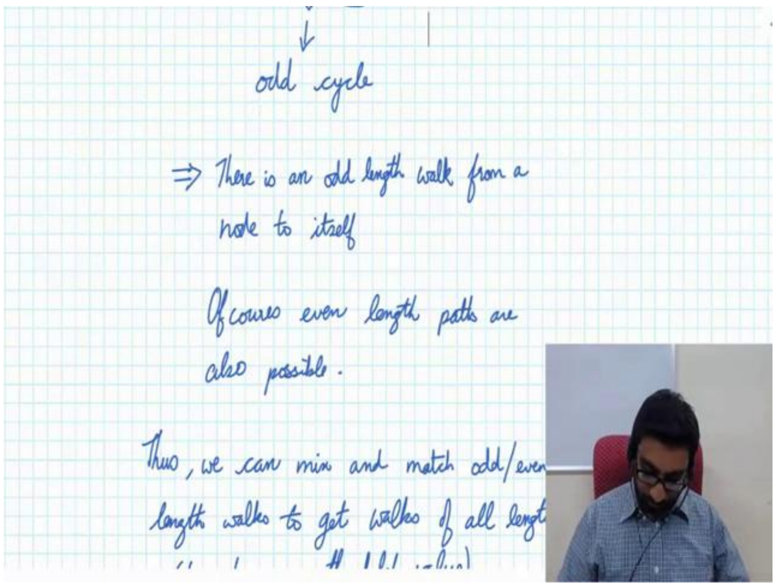
odd cycle

$\Rightarrow$ There is an odd length walk from a node to itself

Of course even length paths are also possible.

What about the reverse direction, well this one, we need to show that a graph that is not bipartite random walk on that such a graph will be aperiodic. An odd graph that is not bipartite is must have an odd cycle. And if there is an odd cycle what that means, is that you can always go from 1 vertex, you can find the way, so when the graph is connected, you can find the way to take a walk from vertex, and then return to it after an odd number of steps. And of course, you can also do even number of steps, you can just go back and forth between nodes and its neighbor. So, combining these two, you can mix and match them and create walks of any length beyond the certain threshold.

(Refer Slide Time: 08:18)



$\downarrow$

odd cycle

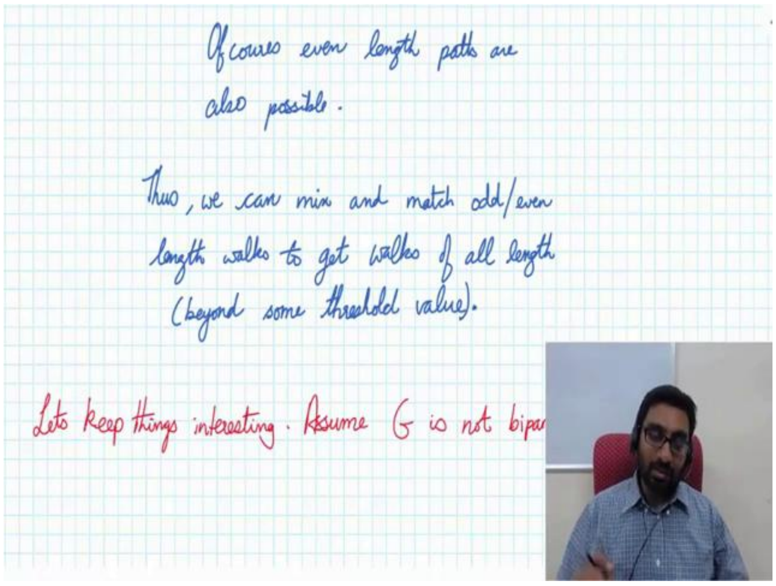
$\Rightarrow$ There is an odd length walk from a node to itself

Of course even length paths are also possible.

Thus, we can mix and match odd/even length walks to get walks of all lengt

And this is the crucial idea by which we can establish that with some nonzero probability, you can actually go from any vertex to another vertex at variety of different time steps. In another words, such a random walk will be aperiodic, so that is the intuition. This is not complete proof, but I want to leave you with an intuition of why this statement holds.

(Refer Slide Time: 08:47)



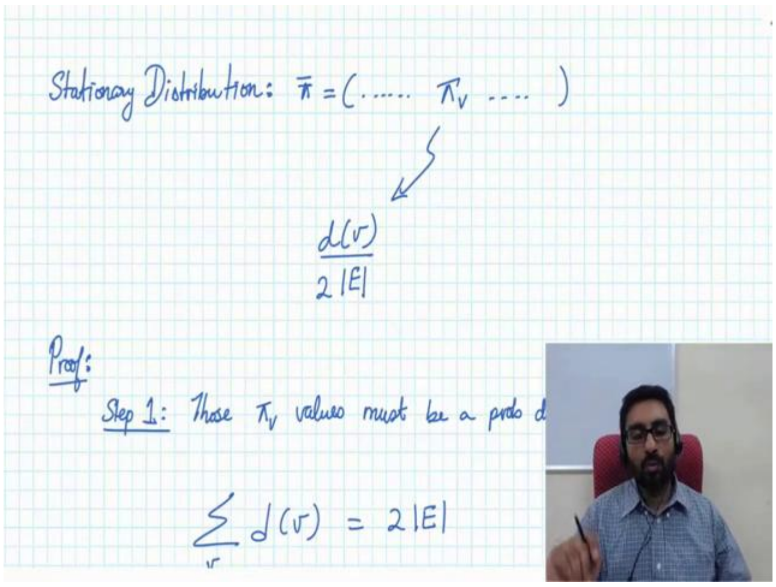
Of course even length paths are also possible.

Thus, we can mix and match odd/even length walks to get walks of all length (beyond some threshold value).

Lets keep things interesting. Assume G is not bipar

So, from rest of the segment, let us keep things interesting, we have been interested primarily in aperiodic Markov chains. So, we are going to assume that G is not bipartite. So this means that the random walk on such a graph is going to be aperiodic.

(Refer Slide Time: 09:09)



Stationary Distribution: $\bar{\pi} = (\ldots\ldots \; \pi_v \; \ldots\ldots)$

$$\frac{d(v)}{2|E|}$$

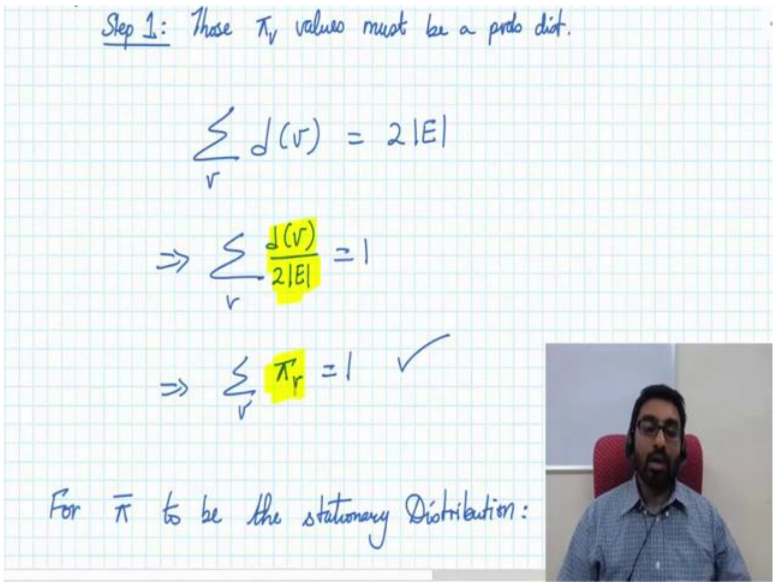
Proof:

Step 1: These π_v values must be a prob d

$$\sum_v d(v) = 2|E|$$

Now, let us look at stationary distribution of a random walk on a non-bipartite connected graph. Claim is that the component associated with the vertex v is given by d v over 2 times number of edges. Why is this, first for we need to show that this stationary this is first of all distribution, this vector of numbers quantities is first of all distribution, we need to show that. These quantities are going to be between 0 and 1 that is clear.

(Refer Slide Time: 10:06)



Step 1: These π_v values must be a prob dist.

$$\sum_v d(v) = 2|E|$$

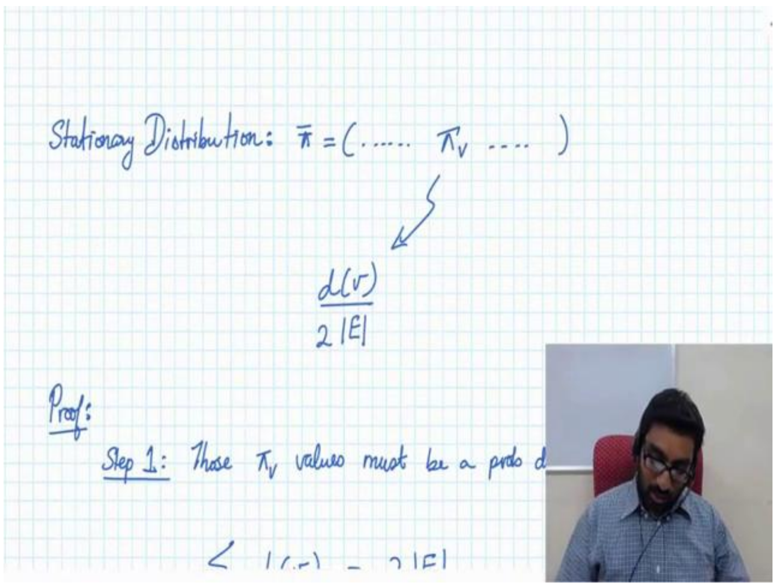
$$\Rightarrow \sum_v \frac{d(v)}{2|E|} = 1$$

$$\Rightarrow \sum_v \pi_v = 1 \quad \checkmark$$

For $\bar{\pi}$ to be the stationary Distribution:

What we need to show is add up to 1, well let us see, how that is going to work now. If you sum up the degrees that going to equal two times the number of edges. So if you rearrange, you are going to get d v over to E summed over all (Refer Time: 10:26) we going to add up to 1. So, d v over E is nothing but the component associated v, and so this is exactly what we want.

(Refer Slide Time: 10:41)



Stationary Distribution: $\bar{\pi} = (\ldots\ldots \; \pi_v \; \ldots\ldots)$

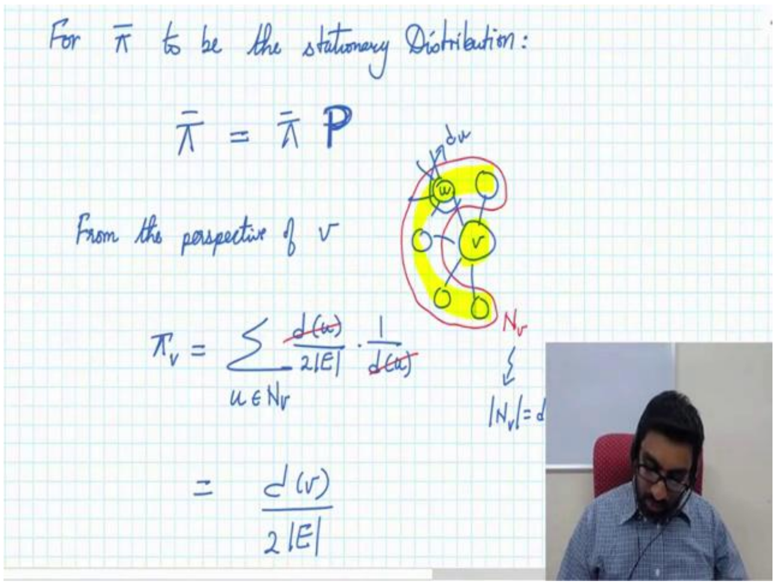
$$\frac{d(v)}{2|E|}$$

Proof:

Step 1: These π_v values must be a prob d

So, we know that this vector of numbers is a probability distribution.

(Refer Slide Time: 10:50)



For $\bar{\pi}$ to be the stationary Distribution:

$$\bar{\pi} = \bar{\pi} \mathbf{P}$$

From the perspective of v

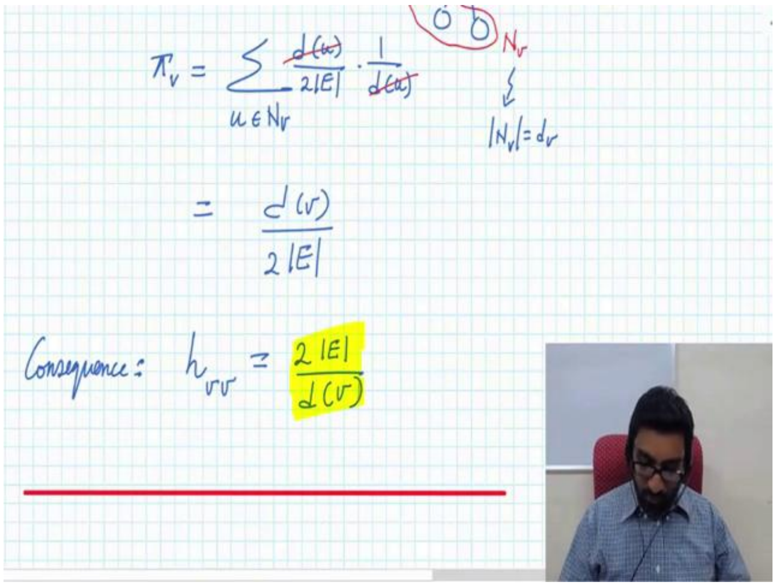
$$\pi_v = \sum_{u \in N_v} \frac{\cancel{d(u)}}{2|E|} \cdot \frac{1}{\cancel{d(u)}}$$

$$= \frac{d(v)}{2|E|}$$

Now, we need to show that this equation holds; this probability distribution vector times the transaction matrix will give you back the probability distribution vector that is what needs to be shown. Let us look at this equation from the perspective of a single vertex, so that is pi v, when you apply this matrix, when you multiply by this matrix. What is that mean? Well, the pi v is, look at the vertex v, its neighborhood these are neighboring vertices and that is denoted by N v.

In order to get this component pi v, what we have to do is sum over all the neighbors of p, and take their corresponding vector components times the probability with which a random walk token from 1 of this neighbors say u will move to v and that is this quality 1 by d u. Why is that, if you look at the vertex u there are d u neighbors, and each one of them chosen equal likely. So, the probability there are random walk token that is at u will take this edge and come to v is 1 over d u. So, pi v is summation over all the neighbors u, the stationary distribution component corresponding to u times the probability with which the token will move from u to v that is essentially what we do over here.

(Refer Slide Time: 13:05)



$$\pi_v = \sum_{u \in N_v} \frac{\cancel{d(u)}}{2|E|} \cdot \frac{1}{\cancel{d(u)}}$$

N_v

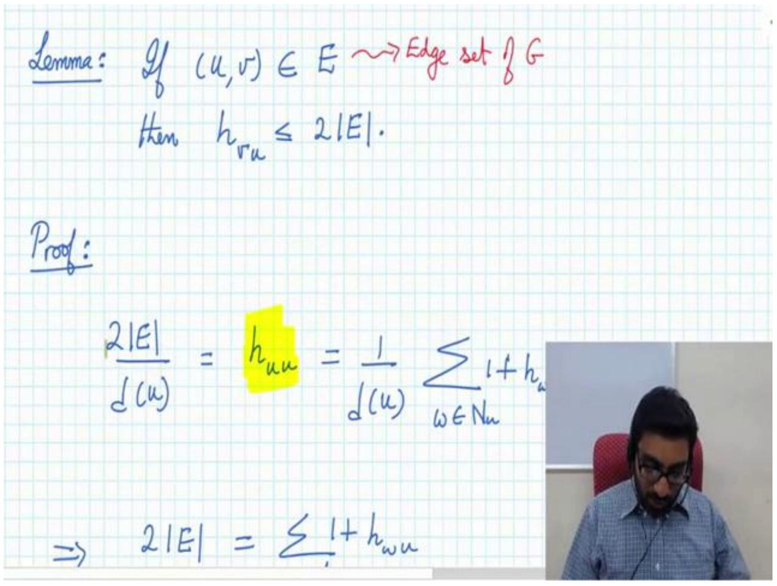
$|N_v| = d_v$

$$= \frac{d(v)}{2|E|}$$

Consequence: $h_{vv} = \frac{2|E|}{d(v)}$

And canceling this two d u we will get 1 over 2 times the cardinality of the edge set, but then we are summing over the neighborhood of u and so we get d v over to 2 times the cardinality of E. So, this completes the proof of our claim that the vector form by these components is the stationary distribution of this random walk. And of course, we immediately conclude that the time the expected time for a random walk to leave v and return back to v is the reciprocal of pi v which is 2 E over d v.

(Refer Slide Time: 14:00)



Lemma: If $(u,v) \in E \rightsquigarrow$ Edge set of G

then $h_{vu} \leq 2|E|$.

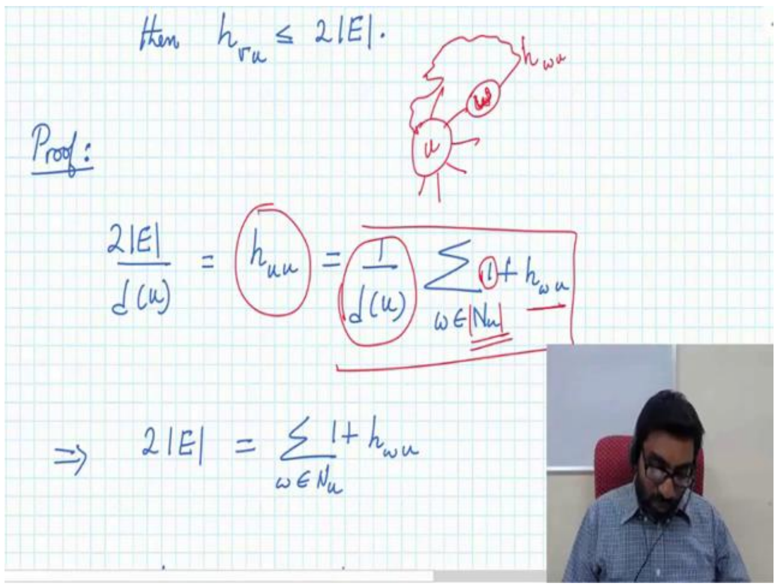
Proof:

$$\frac{2|E|}{d(u)} = h_{uu} = \frac{1}{d(u)} \sum_{w \in N_u} 1 + h_{w}$$

$$\Rightarrow 2|E| = \sum 1 + h_{wu}$$

Let us look at slightly more general h v u. So, let us assume that let us focus on an h in the in the graph. Now, the expected time to go from v to u is at most 2 times cardinality of E of course. If you are lucky you will go on one step, but what it this claims is that even if you are unlucky the expected time to go from a vertex v to its neighbor u is at most 2 times E. Here have the proof of course. So, let us look at h u u you already know that h u u is 2 E over d u, and this what got over here.

(Refer Slide Time: 15:11)



then $h_{vu} \le 2|E|$.

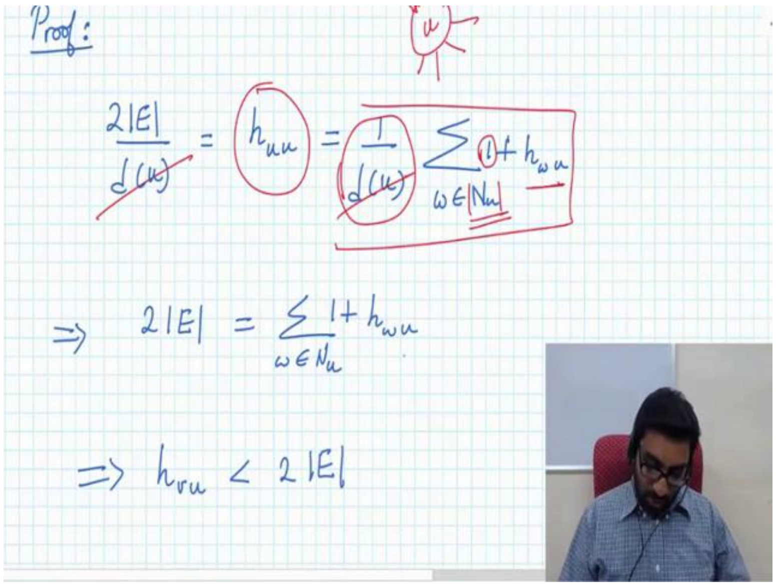
Proof:

$$\frac{2|E|}{d(u)} = h_{uu} = \frac{1}{d(u)} \sum_{w \in N_u} 1 + h_{wu}$$

$$\Rightarrow 2|E| = \sum_{w \in N_u} 1 + h_{wu}$$

But there is also another way to look at this, so from u with probability 1 over d u, we will go to neighbor w; and from that w, we will eventually make a way back to u and that is h u u, but then that is just the option in going to w. Using the law of total probability, we can sum this up over all possible all neighbors of u. And the as result, we get this quantity. So, the submission over all the neighbors w, you go to this one correspond to the one step, we take go to w and this h w u is expected number of steps that we take from w to u. And this option is exercised with probability 1 over d u, and this is just one of many options in particular one of N u number of options. This is the number of neighbors of u; and applying the law of total probability, we get this expression for h u u.

(Refer Slide Time: 16:46)



Proof:

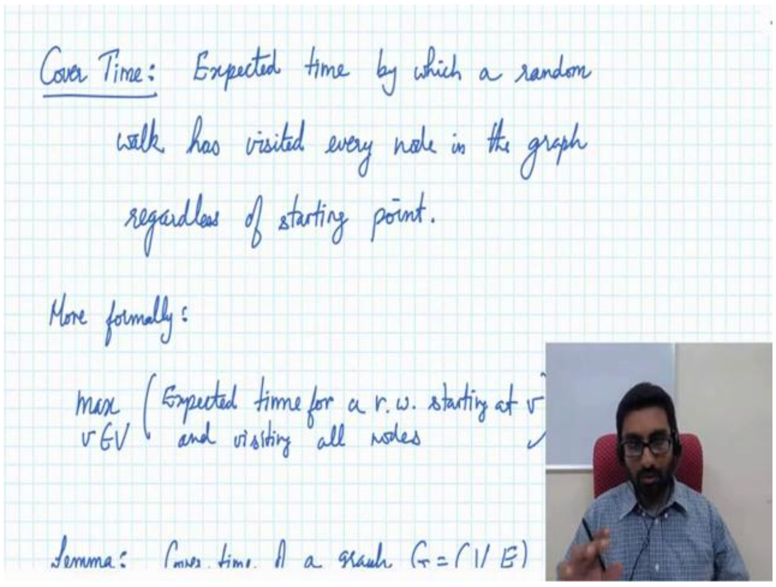
$$\cancel{\frac{2|E|}{d(u)}} = \boxed{h_{uu}} = \boxed{\cancel{\frac{1}{d(u)}} \sum_{w \in N_u} 1 + h_{wu}}$$

$$\Rightarrow \quad 2|E| = \sum_{w \in N_u} 1 + h_{wu}$$

$$\Rightarrow \quad h_{vu} < 2|E|$$

So, this basically this is two ways of computing h u u and so the d u is get canceled will get 2 E equals summation w belong to the neighborhood of u 1 plus h w u. Then of course, one of these w is the E is the v the neighbor that v we are interested in. So, clearly looking at the summation, the expected time to go from v to u must be in fact, strictly less than 2 times number of edges.

(Refer Slide Time: 17:28)



Cover Time: Expected time by which a random walk has visited every node in the graph regardless of starting point.

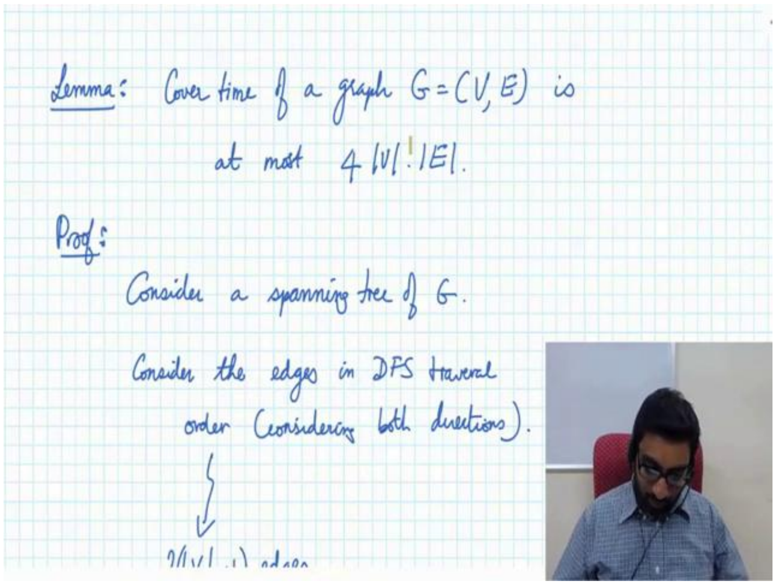
More formally:

$\max_{v \in V}$ (Expected time for a r.w. starting at v and visiting all nodes)

Lemma: Cover time of a graph G = (V, E)

So, why did we even look at this, this actually an interesting consequence, interesting notion called the cover time. You may start to realize, why we are going to talk about the cover time, because this may this will have a bearing on the problem that we started out with. What is the cover time? It is the expected time by which a random walk has visited every node in the graph regardless of its starting point. So, it is the worst case time expected time for a random walk to explode the entire graph. Of course, we are talking about connected non-bipartite graph. More formally, it is the maximum over all vertices in the graph, the expected time for a random walk starting at v and visiting all the nodes.

(Refer Slide Time: 18:26)



Lemma: Cover time of a graph $G = (V, E)$ is at most $4|V| \cdot |E|$.

Proof:

Consider a spanning tree of G.

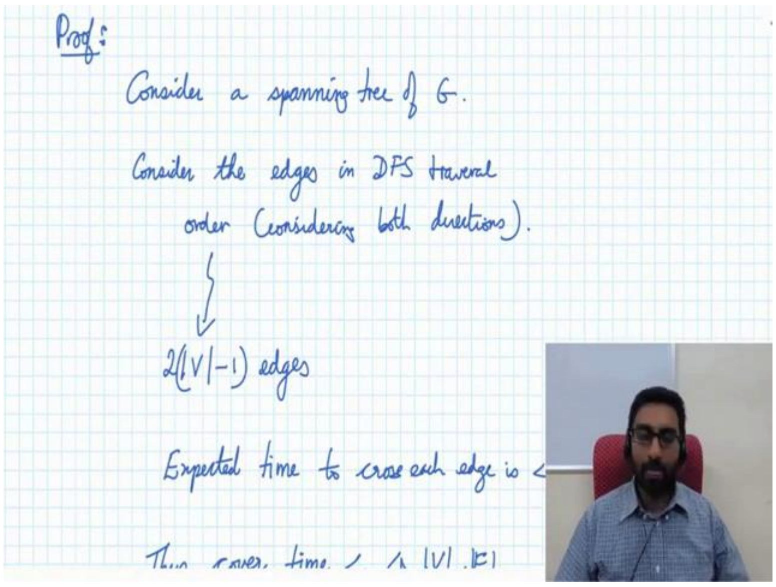
Consider the edges in DFS traversal order (considering both directions).

$\downarrow$

$2(|V|-1)$ edges

The cover time of a graph G equals V comma E is at most four times the number of vertices times a number of edges, this true for any graph. For special graphs, we have better bounds on the cover time, but this is true for any connected non-bipartite graph.

(Refer Slide Time: 18:52)



Proof:

Consider a spanning tree of G.

Consider the edges in DFS traversal order (considering both directions).

$\downarrow$

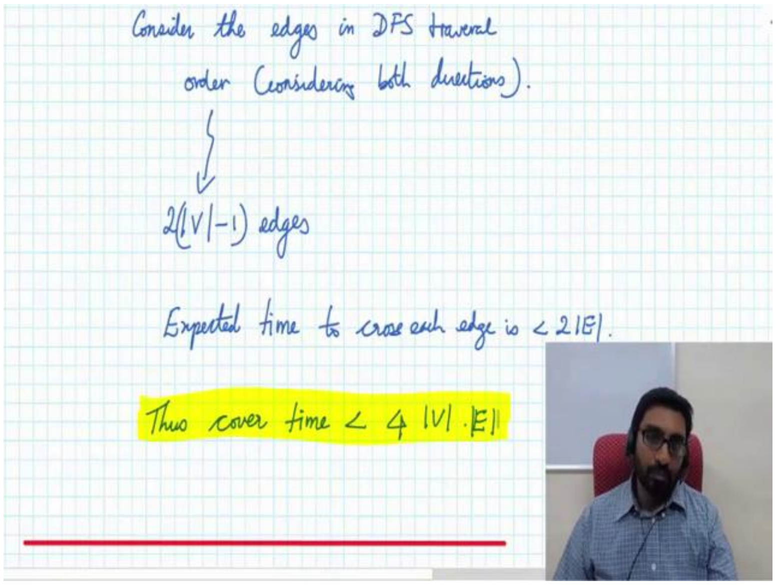
$2(|V|-1)$ edges

Expected time to cross each edge is $<$

Thus cover time $< 4\,|V|\,.|E|$

So, why is this the case? We will consider a spanning tree of G. Now, if you consider the edges in DFS traversal order, now you have to consider both directions. So, each edge will be traverse to two directions that ordering will have at most two times, the number of edges minus 1 number of vertices V minus 1 number of edges.

(Refer Slide Time: 19:34)



Consider the edges in DFS traversal order (considering both directions).

$\downarrow$

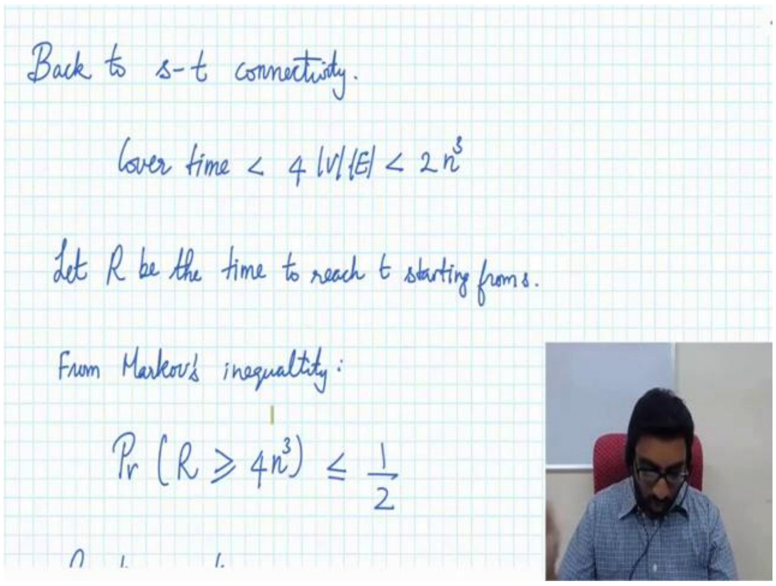
$2(|V|-1)$ edges

Expected time to cross each edge is $< 2|E|$.

Thus cover time $< 4\,|V|\,.|E|$

Why because minimum spanning tree has in minus 1 edge; and each of those edges will traverse it most twice. So, these are all edges in the graph, so the expected time to go from one end to the other is at most 2 E. So, this naturally leads us to the conclusion that the cover time is at most 4 times V times E.

(Refer Slide Time: 20:12)



Back to s-t connectivity.

$$\text{Cover time} < 4|V||E| < 2n^3$$

Let R be the time to reach t starting from s.

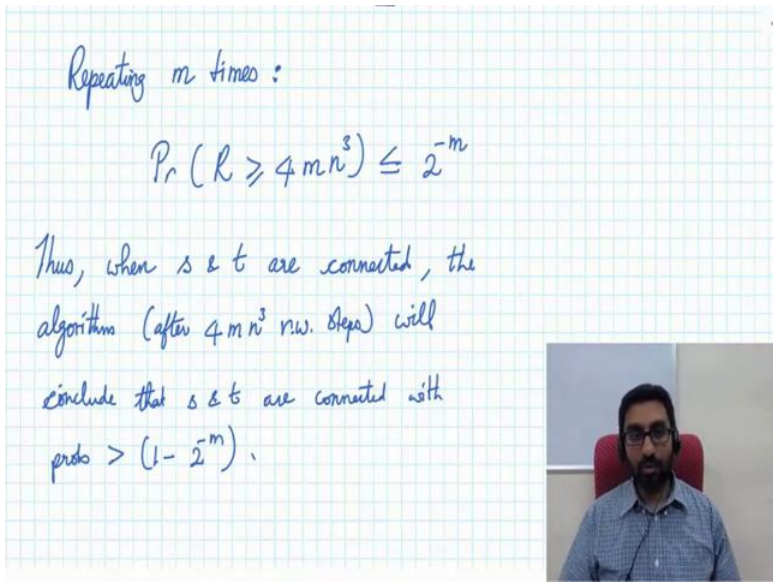
From Markov's inequality:

$$\Pr(R \geq 4n^3) \leq \frac{1}{2}$$

So, back to the s-t connectivity, we know cover time is less than 4 V E at that is at most in 2 n cubed, because the cardinality of edge set is n square less than n squared by 2. First use R the random variable R to denote the time to reach t starting from s.

Remember the algorithm, we from s we start the random walk and we allow this random walk to walk. And R is the now let us assume that the graph is connected of the purpose of analysis. R is the random variable that denotes a time to reach t from s by this random walk. And from Markov's inequality, we know that, the expected expectation of R is 2 n cued. From Markov's inequality, what is the probability that R exceeds 4 n cubed well that is at most half.

(Refer Slide Time: 21:27)



Repeating m times:

$$Pr(R \geq 4mn^3) \leq 2^{-m}$$

Thus, when s & t are connected, the algorithm (after $4mn^3$ r.w. steps) will conclude that s & t are connected with prob $> (1 - 2^{-m})$.

So, if we run it 4 m n cubed, the number of times, and here m is just parameter not be confused with the number of edges because m is after (Refer Time: 21:38) the number of edges, here is the parameter. If what is the probability that R exceeds 4 m n cubed, well it is at most 2 raise to the minus m. So, if you make m, for example, something like log n, this quantity will become 1 by m. So, when s and t are connected, the algorithm after these 4 m n cubed number of steps will conclude that s and t are connected with high probability.

On the other hand, if it is after 4 m n cubed number of steps, you do not reach t then you conclude that the graph is I mean s and t are not connected. And what is the probability, which will be wrong well let us know at most 2 raise to the minus m, so that brings us to the end of this series of segments.

Thank you.