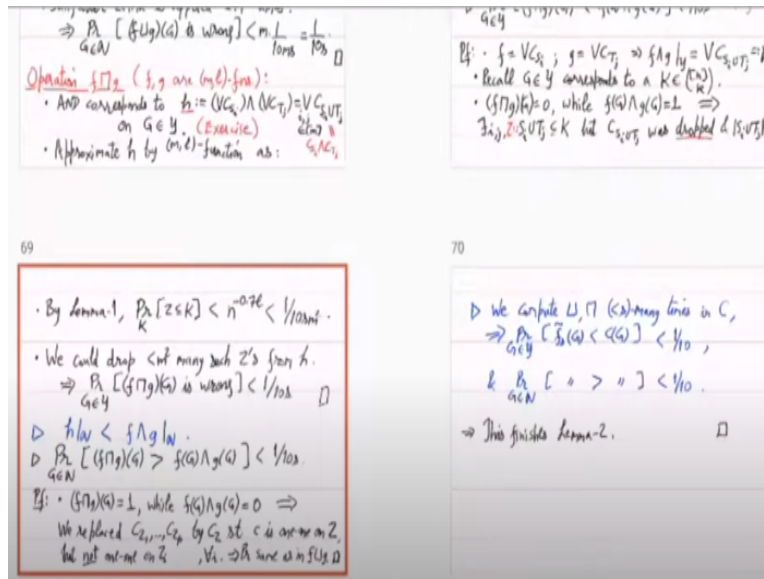


Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science & Engineering
Indian Institute of Technology-Kanpur

Lecture - 09
Undirected Graph Connectivity in Randomized Logspace

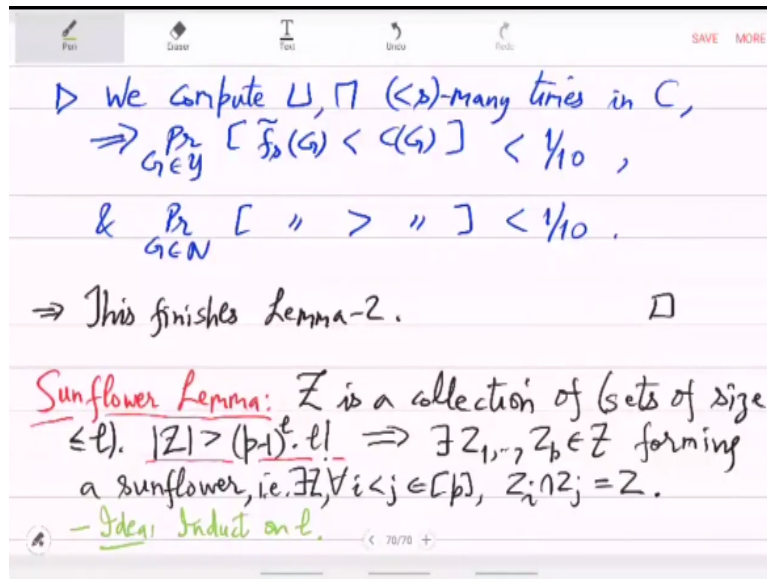
(Refer Slide Time: 00:17)



In previous classes we were in the middle of the theorem by Razborov about the hardness of the clique problem for monotone circuits. Basically the proof strategy was to convert any monotone circuit that solves the clique problem on these two distributions that we have defined - the yes distribution, and the no distribution. Any such monotone circuit we will convert it into these few clique indicators. As long as the monotone circuit is smaller than $n^{\sqrt{k}/20}$.

We can actually collapse it into an or of clique indicators. And on the other hand, we had shown that for these distributions clique indicators cannot solve the problem. So that is how we showed that monotone circuits will be large. While we convert these \wedge, \vee gates in the monotone circuit to this or of clique indicators, we employed the sunflower lemma. So today we have to finish that proof.

(Refer Slide Time: 01:28)

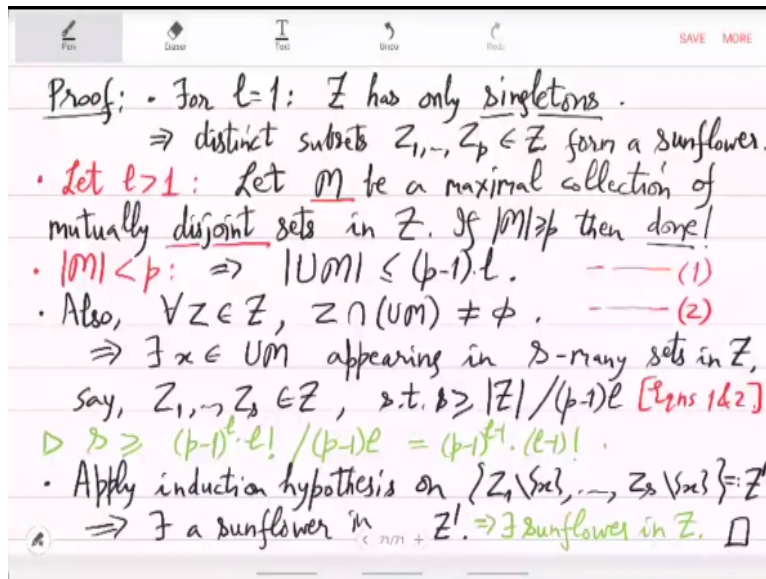


So the sunflower lemma talked about the existence of a sunflower. Basically a collection of sets or subsets that have the same mutual intersection. So Z is a collection of sets of size at most l . It says that as long as you have a lot of sets - more than $(p - 1)^l \cdot l!$. That is exponential in l many subsets then what happens is there is a sunflower.

Then you will have these p subsets forming a sunflower. That is for every pair i, j the intersection is Z , for some subset. This subset Z may not be in the collection. It is just something in the universe. So there exists a Z such that this thing is true for every pair. So we have to basically use this condition in the proof. The way we will use it is we will do induction on l .

So for this expression with $l - 1$, suppose you have a proof then we will show it for l . So the idea is to induct on l .

(Refer Slide Time: 04:20)



Let us first look at the base case. What happens when l is 1? When l is 1 then you are saying that there are at least p subsets and moreover, their sizes each of them has size only 1. These are singletons. So obviously if they are singletons and they are distinct then the usual intersection is empty. They are disjoint, which means that Z_1 to Z_p in Z form a sunflower.

Notice that we never said that intersection will be non-empty, we just want in a sunflower the intersection to be the same. It can also be empty, in this case it will be. This Z has a collection that has at least p subsets singletons, you pick them, that is a sunflower. So that case is done. Now let us move to the induction step which is l greater than 1. So here just taking a cue from the base case, let us first try to collect disjoint subsets.

So let M be a maximal collection of mutually disjoint sets in Z . Maximal means that if you take any extra set and put it in M , then there will be an intersection with something already in M . Collect as many disjoint sets as you can. If you already have p then you are done. We are then in the case $|M| < p$. What do you do in this case? So you do not have enough disjoint sets.

Now whatever you add it will intersect. So in this case, we look at the union of all the sets in M . How many elements will you get? This will be M has at most $p - 1$ set, each of size at most l . So $|\cup M| \leq (p - 1) \cdot l$. Also the second property is that I

already said that for every subset in the collection Z this set Z will intersect with something in M . So in the union of M there is some intersection, it is non-empty.

So this means that there exists an x in $\cup M$. Note that M has at most $p - 1$ sets. First let us look at the frequency of x . So there is an x appearing in s -many sets in Z say Z_1 to Z_s where I want this s to appear in many subsets in M . I want to maximize this. So what can I say about s ? How big can I choose? So how many x 's are there? So let us first look at this equation 1 and this equation 2.

So from equations 2 you get that in the union of M there is an element or the other way. So every subset in the collection Z will contain some element from $\cup M$. So how many subsets are there? These many subsets are there right, size of this Z . And how many elements are there in $\cup M$ that is at most $(p - 1)/l$ by equation 1.

So what I want to say is this ratio if you look at the size of Z divided by the number of elements, number of subsets divided by the number of elements in $\cup M$ there is some element which appears in that many sets. We are basically doing an averaging argument by which we can say that if you look at this ratio of sets, number of sets divided by number of elements there is some element which appears in so many sets.

Because if that is not the case, if every element appears in fewer than these then what you will get is a contradiction. Then you will be contradicting equation 2. Because equation 2 is saying that every subset has to intersect. So some element x in $\cup M$ has to appear a lot in many subsets. Now what is s ?

So s is greater than equal to the size of Z we have picked to be $(p - 1)^l \cdot l! / (p - 1) \cdot l$, which is equal to $(p - 1)^{(l-1)} \cdot (l - 1)!$. This expression is familiar. This is the same expression if l was $l - 1$. But how do we invoke the induction hypothesis?

The way we will do it is that we will actually look at those sets where x appears. We will find a sunflower in Z_1 to Z_s . That is the idea. So apply the induction hypothesis on $\{Z_1 \setminus \{x\}, \dots, Z_s \setminus \{x\}\} = Z'$. These are subsets whose size is $l - 1$ or less. And s is

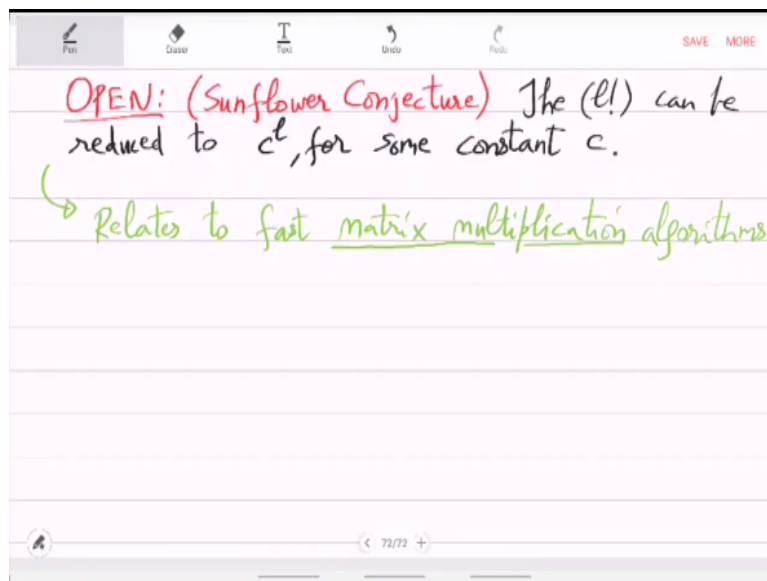
at least $(p - 1)^{(l-1)} \cdot (l - 1)!$. You can use the induction hypothesis and what you will get is there exists a sunflower.

There exists a sunflower in Z' , which means that p subsets are here with the same intersection. If you bring x back, the intersection will still remain the same. It is a sunflower in Z . That finishes the inductive proof. We did the base case and we did the case where M is not large enough, it is $p - 1$ or less.

The idea was that then $U \setminus M$ will have fewer elements and you can use the induction hypothesis by removing this element x . This also finishes Razborov's theorem. We finished this theorem that we started many lectures back saying that for $k = n^{1/4}$ any monotone circuit solving k -clique will require this $n^{\sqrt{k}/20}$ sized monotone circuit.

Further this sunflower lemma itself is actually a very interesting combinatorial fact and one open question I can state.

(Refer Slide Time: 15:14)

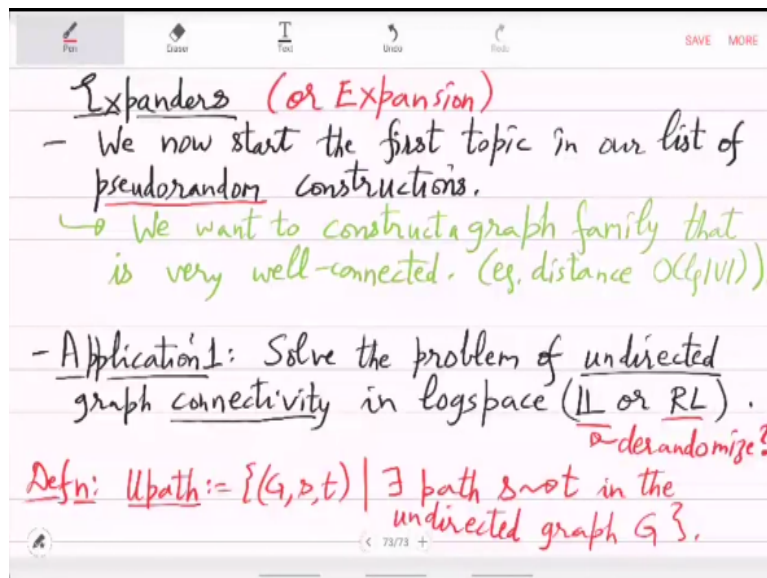


So what is open is the sunflower conjecture. It says that this bound that you had, $l!$, this can be slightly reduced. It can be reduced to let us say 2^l or some constant to the l . Whether this bound is optimal or not, can you reduce this bound? So specific conjecture is that the $l!$ can be reduced to c^l . This is called the sunflower conjecture.

It is still open and one interesting thing is that this conjecture actually appears in some other problems. For example, it appears in matrix multiplication algorithms. If you can prove this then there are these tricks by which you can do faster matrix multiplication. If somebody wants to read this as an advanced topic, they are welcome. At this point now we will change the topic kind of drastically.

From proving lower bounds for these restricted circuit models we will now move to Pseudorandom constructions. One aspect of this course as we had advertised is that we will try to construct these mathematical objects which have pseudorandom properties. The first example of that phenomenon that we will see is in the area of graph theory, specifically called spectral graph theory and the topic is expanders.

(Refer Slide Time: 18:06)



This will be the first topic in our list of pseudorandom constructions. It may not be clear what is the precise meaning of pseudorandom object or pseudorandom construction, but here you can think of properties of a randomly chosen graph. One property that you can show a random graph has is that the connectivity is very high. If there are n vertices then you can reach from any vertex s to any vertex t in short paths.

That kind of property we want in an explicitly constructed graph. Well-connected graph which is explicit. Construct a graph family that is very well connected. Not only can you reach from any vertex to any vertex, but you can actually reach in very few steps. For n vertices instead of the distance being n , the distance will only be $\log n$, let us say.

In terms of vertices, it will be logarithmic distance or diameter. That is just one property. These are highly useful graphs. They are also useful in practice and they have great applications in proving theorems, even unrelated problems; sometimes expanders are used.

One such application is already in graph theory, where expanders are used or the concept of expansion is used. We can call this application number 1. This is the major application. We want to solve the problem of undirected graph connectivity in logspace. Given a graph in the Turing machine, the work tape will only have $\log n$ or $O(\log n)$ space, only those many cells will be used.

And just by $O(\log n)$ space, you want to decide whether there is a path from s to t . You want to solve the connectivity problem. Now note that the number of paths can be like $n!$ or at least 2^n , the number of paths can be exponential. You cannot just blindly or by brute force search the space of paths, you have to be more systematic.

In particular, if your graph already had diameter $\log n$, then maybe you can come up with a brute force algorithm and then you can solve the problem in logspace. But what if your graph was not well connected, that is the diameter was much bigger than $\log n$? How do you solve those connectivity problems on those input instances? That is a major area in computer science.

In the end of the study on expansion we will solve this problem completely. You will see a solution. First we will start with RL. So RL is basically a randomized logspace algorithm. This will be much easier, and once we have developed this area enough, then we will show that actually it can be derandomized.

This will be a derandomized version. We will do random first and then we will derandomize the randomized algorithm. So let us define the problem formally. This Upath the input instance is graph, source vertex, target vertex; G, s, t such that there is a path from s to t in the undirected graph G .

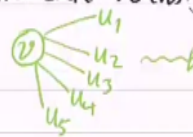
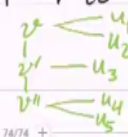
So this is the language and the decision problem is whether a given input instance is in this set, in this language or not. That is how we have modeled our problem.

(Refer Slide Time: 25:39)

Theorem (Aleliunas, Karp, Lipton, Lovász, Rackoff '79):
 $U_{\text{path}} \in \text{RL}$ (randomized-logspace-algo)

Proof: Idea - Consider adjacency matrix A & simulate a random walk on graph G as matrix powering.

- Suppose G is the given undirected graph with n vertices.
- We need G to be d -regular (ie. $\forall v \in V(G), \text{deg}(v)=d$).
- If we can transform G to get $d=3$:

eg.   Δ Apply this gadget on all the vertices, in logspace: $O(\log n)$.

And now comes the main attraction which is that well, first of all U_{path} in practice is a very easy problem. Connectivity you can solve by doing a depth-first search. You can solve it in linear time, it is obvious that U_{path} is in P. But you want to solve it much better than polynomial time.

So this major non-trivial algorithm for U_{path} was given by Aleliunas, Karp, Lipton, Lovasz and Rackoff in 1979. This AKLLR algorithm puts the problem in in RL. So imagine that you are given a map of a city, and then you want to move from place s to place t . Obviously, the map of a city can be very large.

You do not want to remember the whole map, but you can pick any point in the map and then you can look at the adjoining locations basically connected by a street. And then you can start moving on a street, go to another location from there, again pick a street, go to the third location and so on. The same thing, can you simulate this thing by using very little space. Your workspace is very small.

You cannot store the whole map. You can only look at local neighborhoods in the map and make decisions based on the local neighborhood, not the global information or the global map. So this theorem says that yes that can be done. How is that possible? This

seems impossible. How can you not store the whole map and just based on very local decisions, solve this global problem?

This will be the starting point of spectral graph theory. We will basically simulate this walk on an undirected graph. We will simulate this walk as a matrix multiplication problem. Look at the adjacency matrix. Consider an adjacency matrix and simulate a random walk on graph G as matrix multiplication or matrix powering. Kind of multiplying by matrix A , then multiplying A again, so A^2 , then A^3 .

Basically, you want to start from your vertex S and then randomly make choices of neighbors and keep on walking and simulate this as if you are multiplying by this adjacency matrix or vector. And then after we have done this for a long time, study where you reach. So what is the probability of reaching vertex t in let us say l steps? That is the analysis we want to do.

We will use Matrix analysis and probability. The tools here will be very interesting. Suppose G is the given undirected graph with n vertices. First what we will do is in the matrix analysis part or in this random walk analysis part we will need this graph to have a special property. Basically the property we want is whenever we are at a vertex the number of choices should be constant.

It should be fixed and it should not change. In other words the degree of every vertex of the graph be the same. We want G to be regular. That is for all vertices v the degree of is d . If you want every vertex to have the same degree, obviously, the input graph may not have this property, so we will transform the graph. Again, we have to transform it in a logspace way.

So for example we can transform G to get $d = 3$. Why is that? Let us just see this in a picture. Suppose you have a vertex v with 5 neighbors. Then we can add more vertices and achieve degree 3 everywhere. How do you do that? Let us make three copies or let us create these different vertices v, v', v'' . And v is connected to u_1, u_2 .

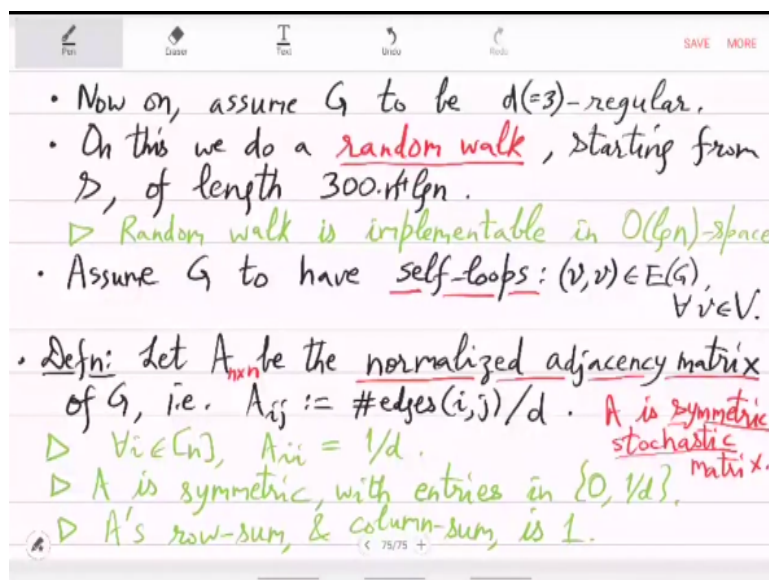
And v is also connected to v' and v' is connected to v'' . So v now already has degree 3. Let us now move to v' . Give u_3 to this and also, and now degree of v' is 3 and degree

of v'' can be made 3 by adding u_4, u_5 . Let us say these copies of $v; v, v'$ and v'' , these have degree 3. At least this vertex we have made degree 3.

And then on the other vertices you do the same trick. Apply this gadget on all the vertices and for this you do not need global information. You only need information about the neighbors. So this can be done in logspace. Because the neighbors are at most n . You can actually access the neighbors just by their indices, which will only require $\log n$ space.

In terms of vertices n or even the edges, which can only be n^2 . In log of that much space is enough and so G can be transformed to a new graph which can be written on an output tape. Work tape is only $\log n$ space.

(Refer Slide Time: 36:06)



What we have done is now assume G to be 3 regular. So we can now assume G to be d regular where d we can even pick 3. Notice that we cannot manage this in $d = 2, 3$ is the minimum. Now the algorithm is as suggested before, a random walk. The question is how long the random walk will be? How many steps do you have to take?

On this we do a random walk starting from some vertex s of length $300 \cdot n^4 \log n$. Do not worry too much about these factors, just focus on n^4 . What we are saying is that we will take random steps around n^4 many times starting from s . And the claim is

that if t is connected to s , then we will reach it. By a random walk, we can reach t in these many steps with a high probability.

And since it is a random walk, it does not need global information, we will only look at the neighbors. Since the length is n^4 , again when you want to count the steps, you only need a log of that many. So $\log n^4$ many bits. Overall everything you can simulate or implement in $\log n$ space. This is definitely a log space algorithm.

The only question left is what is the error probability or what is the success probability. Also assume G to have self-loops. Which means that for every vertex v we need this edge v . These self-loops are also present. This obviously would not change the connectivity. If s to t was connected then it will remain connected. We just throw in these self-loops also. This will help in the analysis.

It may not be clear why but somehow it will. So in the end you will understand why self-loops are added. And now comes an important definition of the adjacency matrix, but we will use a normalized version. So let A be the normalized adjacency matrix of graph G . They have great applications in proving theorems, even for unrelated problems as well.

By that we mean every entry A_{ij} is either ij is an edge or it is not. Either it is 0 or 1. But you also normalize it by the degree. So degree normalized adjacency matrix. We can write it as the number of edges from i to j divided by d . There are fractions either 0 or $1/d$, which is one-third actually, so either zero or one-third.

Since there were self-loops, in the diagonals you will have $1/d$. And since it is undirected it is a symmetric matrix. These are the simple observations. Let us note them down anyways. For all i , 1 to n , A_{ii} is $1/d$ because of the self-loops. A is symmetric with entries 0 or $1/d$. And finally, the row sum and column sum is 1.

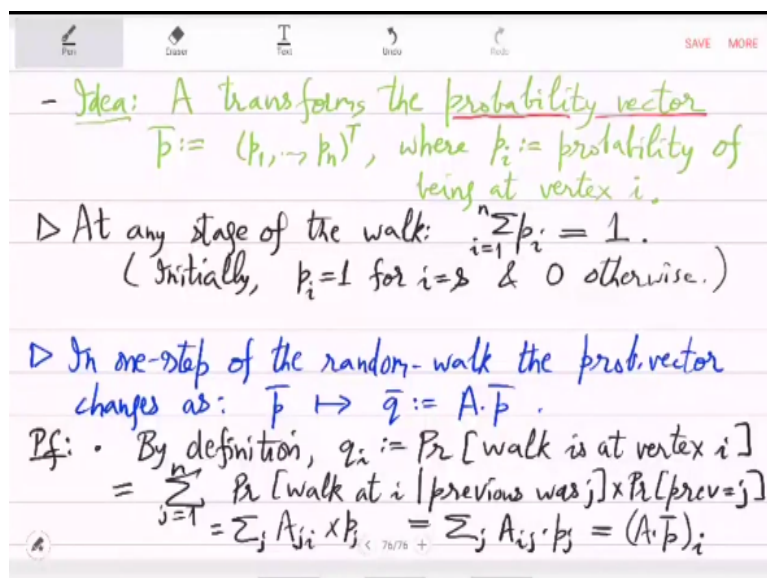
So this is the reason why we normalized it because when you look at a row, i -th row basically the number of neighbors of i have is d . So when you normalize you get the

row sum 1. In matrix analysis they are called Stochastic. That is the property of A , symmetric and it is stochastic. And the entries are not arbitrary.

Actually the entries are also very special, they are just 0 or $1/d$ non-negative entries, but very special. This is an $n \times n$ matrix. We suggested that we will think of this matrix as a transformation matrix. So what is it transforming? It will transform or we will view this as transforming the probability vector.

So for the i -th vertex there is a probability of u being there at any point of time in the algorithm. At any time there is a vector of probabilities. There are n -probabilities. Their sum is 1 because you will be at least somewhere. The sum will be exactly 1. And that vector this matrix transforms as the random walk proceeds.

(Refer Slide Time: 45:13)



So A transforms the probability vector. We will prove this formally. Right now it is only an idea. Next we will do the Calculations. So A , think of A as transforming the probability vector $\bar{p} = (p_1, \dots, p_n)$. Think of it as a column vector where p_i is the probability of being at vertex i . That is the idea. That is how we will think of this matrix.

At any stage of the random walk $\sum p_i = 1$ and initially p_i is 1 for $i = s$ because s was the source vertex where you were in initially and 0 otherwise. Initially \bar{p} was the elementary vector and then as you move forward, as you do the random walk, the

probability will kind of distribute. From s it will go to its neighbors, then to neighbors of neighbors and so on.

As your network grows, the probability spreads to other vertices and ultimately after many steps of the random walk you might have a decent probability of being everywhere. That is what we will prove. We have now study this probability vector \bar{p} and see how it will change in one step.

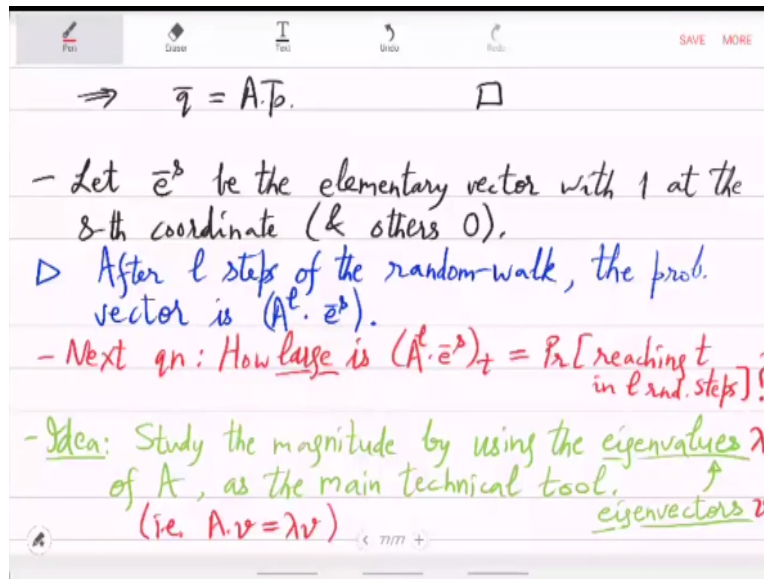
That is the key property which will help you understand or help you see matrix A as a transformation of probabilities or even a matrix of probabilities. In one step of the random walk, the probability changes, so \bar{p} will become a probability vector $\bar{q}: = A \cdot \bar{p}$. The transformation is very simple, you just multiply with the matrix. That is one step. Once you see this, the proof is quite easy.

By definition, q_i is the probability of the walk being at vertex i . Now if you are at i now then in the previous step you were at some other j and you have to look at the probability of reaching i from j in one step. So that is a sum over probabilities. Probability of walk at i given that previous was j , let us say, times the probability that previous was j . And j is over all the vertices, so 1 to n .

The probability that the walk is at i coming from j , that probability is j to i . Now in a random walk you will pick a neighbor with equal probability. Only if j to i there is an edge. That probability is exactly given by the matrix entry.

Matrix entry is A_{ji} times the probability that previously you were at j , which is p_j . Now remember that j_i is i_j by symmetry. This is the inner product of two vectors, the i -th row vector and probability vector \bar{p} . So we can write it as $(A \cdot \bar{p})_i$. This is the same as the inner product of the i -th row of A with \bar{p} . And this is true for every i , which means, that \bar{q} is $A \cdot \bar{p}$.

(Refer Slide Time: 53:14)



Now once you understand one step, you understand a lot more because every step is matrix multiplication. Two steps is $A^2 \cdot \bar{p}$, and 1 steps is $A^l \cdot \bar{p}$. If you call the first probability vector to be \bar{e}^s be the elementary vector with 1 at the s-th coordinate and others 0, then you can write that after l steps of the random walk the probability vector is $A^l \cdot \bar{e}^s$.

So just multiplication or powering of A, l times. The question that remains now is what is the t-th entry of this? Remember because our original goal is to understand the randomized algorithm we gave. That after l steps of the random walk starting from s will you reach t, with what success probability. That is what we want to answer.

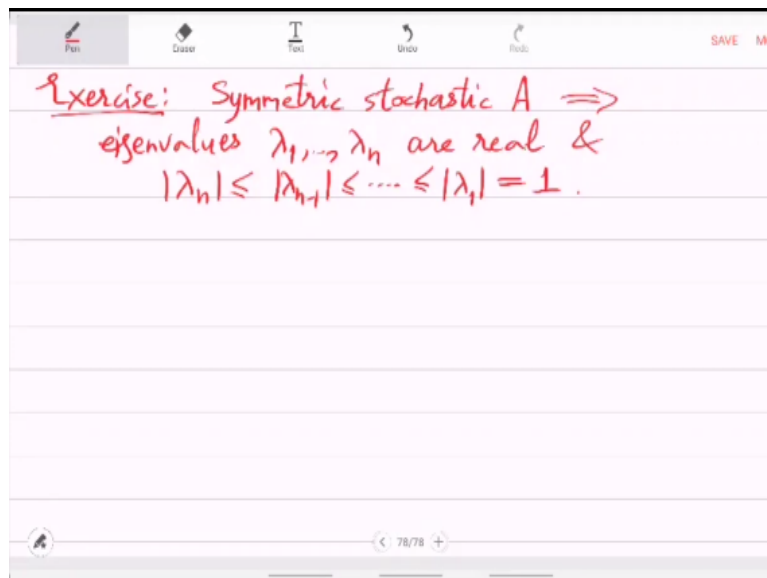
What is the probability of reaching t in l random steps. Is this a decent probability? So for example, if this probability was 60% it is a good chance of reaching t. The question is, is this probability 60%, 80% or is it just a very small probability, exponentially low probability?

Anything can happen because the number of paths is too high. The number of paths is at least 2^n . Your graph was a general graph. So the probability can also be $1/2^n$. Is it that small or is it much bigger? So this is what we will analyze. We will not be able to finish it in this lecture. But next time we will finish this analysis. The main tool now will be to understand this A^l . And we have to move to its eigenvalues.

And then everything will become a study of eigenvalues of the matrix A and hence A^t . To study the magnitude by using the eigenvalues of A as the main tool. Eigenvalues, and when you study eigenvalues obviously eigenvectors as well appear. So what is the meaning of this? Recall that an eigenvalue is λ and the eigenvector is v . If the action of A on v is simple it is just scaling by λ .

This is the meaning of eigenvalue and eigenvector. It basically reduces the study from a matrix to a number. λ is simply a complex number. In this case, what happens in our case because of the symmetric nature of the matrix A , the eigenvalues are all real. So I will leave this as an exercise.

(Refer Slide Time: 59:40)



So symmetric stochastic A implies that eigenvalues are real. Eigenvalues λ_1 to λ_n there are n eigenvalues possibly with multiplicity are real and if you order them by their unsigned value, so smallest magnitude is λ_n , then it is λ_{n-1} , and so on till λ_1 . The largest value is exactly 1. So prove this as an exercise. The eigenvalues are actually real and the maximum is 1. Based on this now we will analyze A^t 's action.