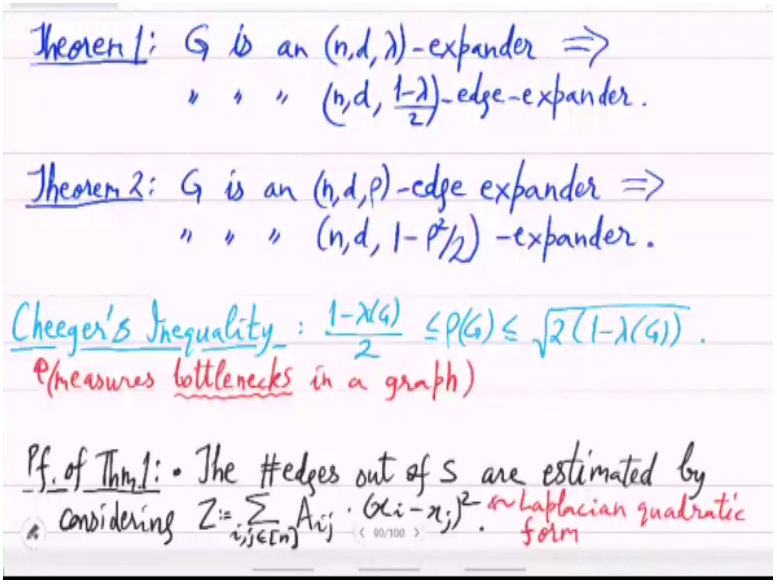
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science & Engineering
Indian Institute of Technology-Kanpur

Lecture - 13
Error Reduction Using Expanders

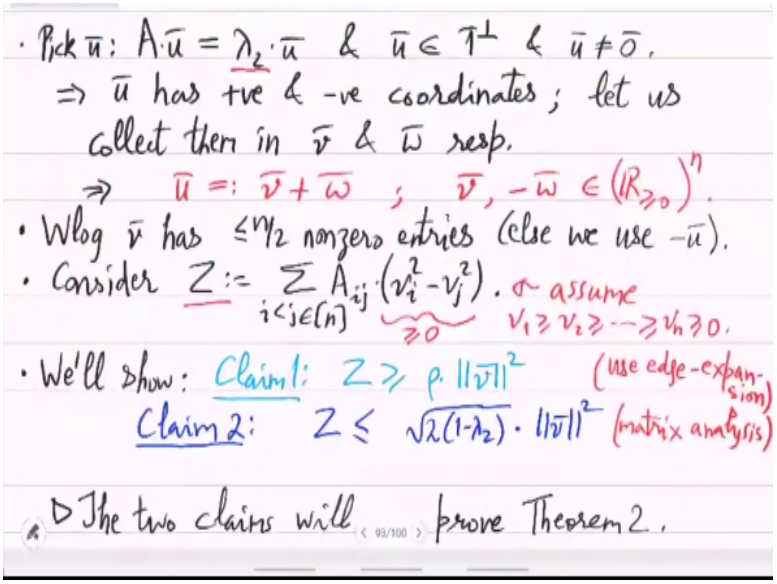
In the last class, we continued the discussion on algebraic expansion versus combinatorial expansion or edge expansion.

(Refer Slide Time: 00:25)



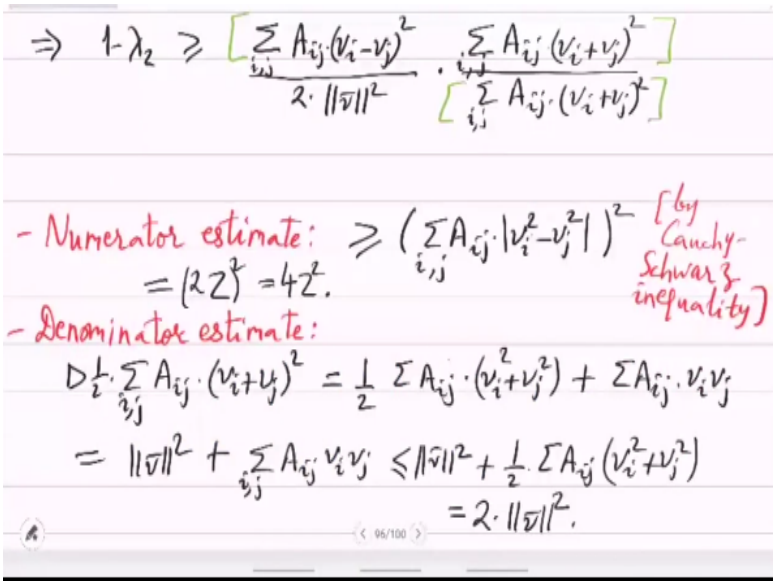
And we showed these two theorems. Theorem 1 saying that algebraic expansion means edge expansion and theorem 2 talking about the converse. There were some mistakes in the constant, which has now been corrected. You can look at these new slides I corrected here.

(Refer Slide Time: 00:45)



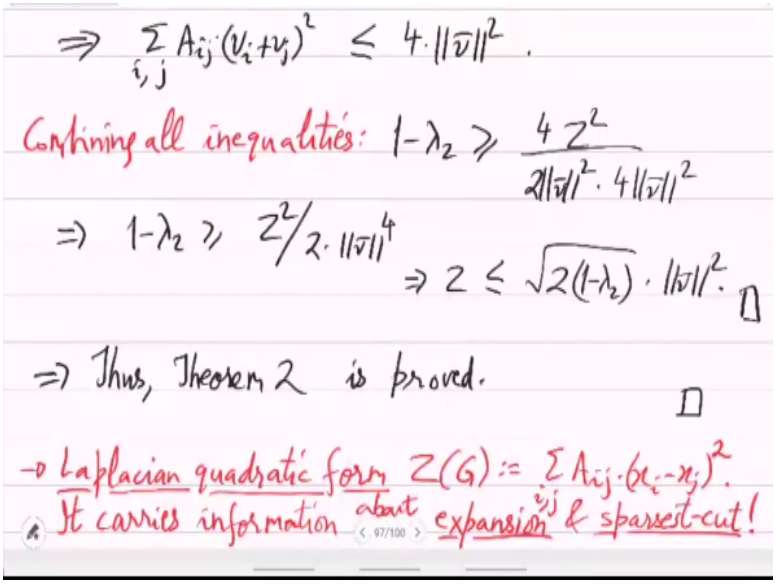
Now to prove theorem 2 there were two claims, claim 2 I have corrected the constant again.

(Refer Slide Time: 01:03)



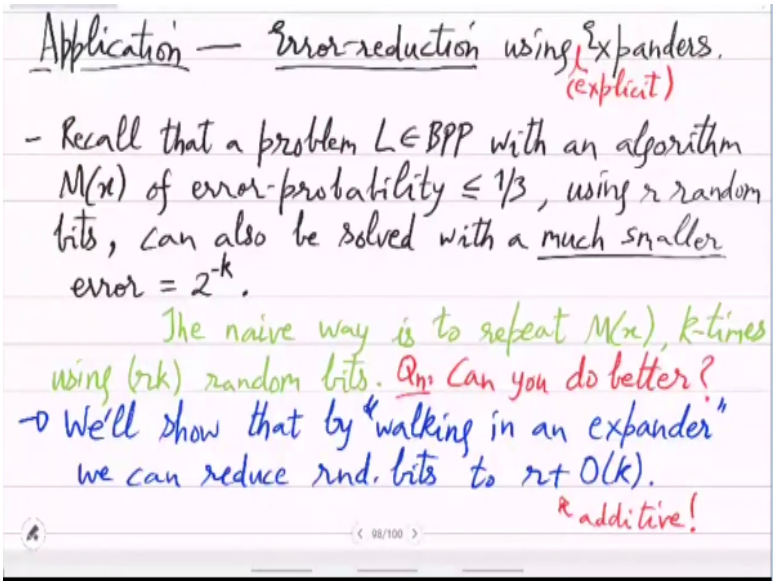
And in this proof, there was some mistake, which is now corrected. We look at this expression $1 - \lambda_2$ greater than equal to numerator divided by the denominator. Numerator estimate is, at least $4Z^2$.

(Refer Slide Time: 01:17)



And the denominator estimate is $8 \cdot ||\, v\,||^2$. The ratio comes out to be $Z^2/2 \cdot ||\, v\,||^4$. So you get this constant $\sqrt{2}$. So this part I have corrected.

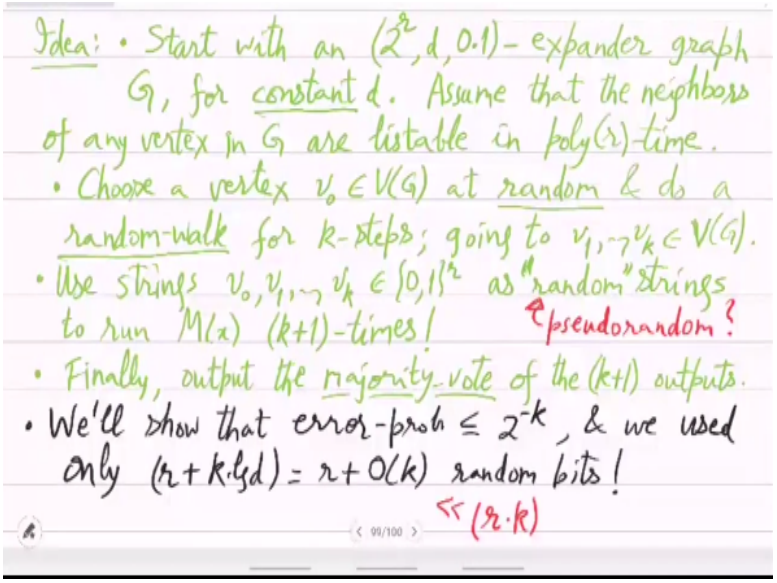
(Refer Slide Time: 01:33)



Then we started an interesting application of explicit expanders, which is that, suppose in a general randomized polynomial time algorithm you were using or the algorithm was using r random bits to get error probability smaller than one third. So how can you make the error probability exponentially small, which is 2^{-k}.

The obvious boosting will require repeating the experiment k times which will give you - r times k random bits. Instead of that we can do it in r + k random bits, the same exponentially small error probability.

(Refer Slide Time: 02:19)

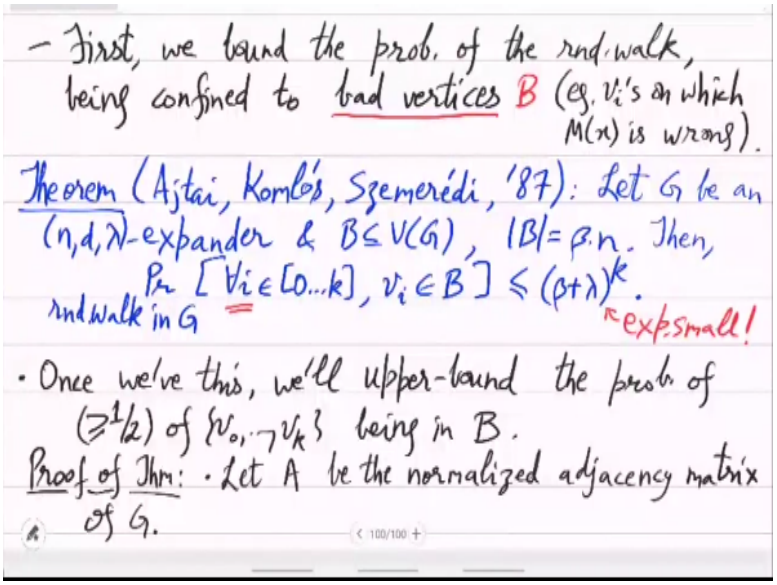


The way this is done is we will start with a very large expander graph, which will be very explicit. For any vertex the vertex label is r bits, which you can think of as the random string supplied to your randomized algorithm. But there are d neighbors only, and think of $d = 3$. There are d neighbors, which means that you can do a random walk easily on this once you know the neighbors.

That was the explicitness condition. The spectral gap you assume is larger than 0.1. Based on this, you do a random walk starting from v_0 for k steps, and that will give you k vertices in the path. So you have kind of k pseudo random strings. This you can use in the randomized algorithm. Or you basically are repeating the algorithm k + 1 times.

You repeat the algorithm k times because now you have k pseudo random strings and then take the majority vote, that was the idea. How many random bits did you use in the random walk? For the first one, r and for the remaining one, it is only $k \cdot \log d$. But $\log d$ is a constant, absolute constant. So this is just like r + k much smaller than $r \cdot k$.

(Refer Slide Time: 03:46)



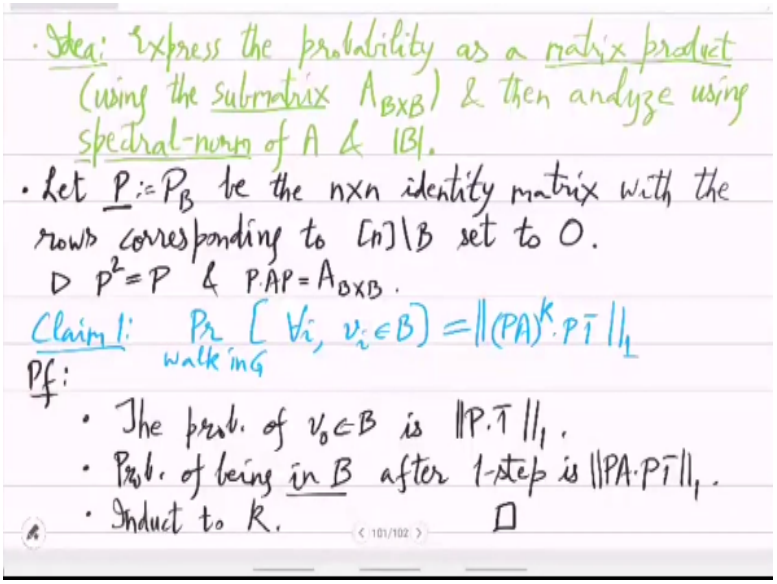
The main thing to calculate is the error probability. Let us do that now. We will be proving Ajtai, Komlos, Szemeredi's theorem, which will basically look at the probability of the random walk being stuck in the bad vertices. Bad vertices, think of them as those strings, labeling vertices that are bad for the algorithm. The algorithm makes mistakes on those choices.

Let us first show that the probability of the random walk being always in this bad subset B is quite small. If you are doing this k step random walk then it is $(\beta + \lambda)^k$. β is the fraction of bad vertices. Out of the total number of vertices n, βn are bad. And λ is the second largest eigenvalue. If $\beta + \lambda$ is smaller than 1, this is an exponential drop in k.

Let us prove this theorem in blue. And once we have a proof of this theorem, then we will show how to get at least half of the vertices outside the bad set B. That calculation will be similar to this and then you can use it in the algorithm. Let A be the normalized adjacency matrix of the graph G.

But now since you only want to consider the case where every vertex goes is in B in the random walk, you will actually only need a part of the adjacency matrix, the part which is $B \times B$.

(Refer Slide Time: 06:27)



We will express the probability as a matrix product using the sub matrix $B \times B$ because you do not even care in this random walk about the vertices outside B. You only care about vertices in B and then walking to B and then analyze using the spectral norm of A and the size of B. This is also important. If B is everything or in the other case, if B is empty right, then the probability will be 0.

But if B is large, then the probability will be high. So probability will really depend on the size of the B. This matrix product sub matrix and spectral norm are the key words in this proof. Let P be the $n \times n$ identity sub matrix with the rows corresponding to $[n]\setminus B$, which is $\overline{B}$ set to 0.

So basically kill these $\overline{B}$ once in the position of $\overline{B}$ and others you keep, that is your matrix P. And you can easily see that $P^2 = P$, because identity square is identity. So a more interesting thing is that the probability on the walk that every vertex v_i is in B in the walk.

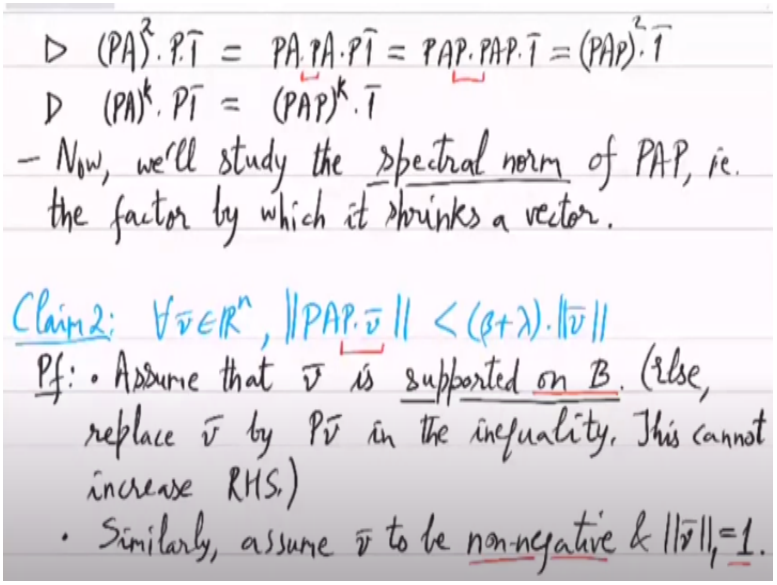
This is exactly equal to the matrix product based on PA, k times. A random walk is very tightly related to the matrix powers of A. But since here, you do not want to go outside B, you always want to remain in B. So you should actually do matrix powering on $P \cdot A$. And so you will get a probability vector and then you take the 1 norm. You sum the probabilities.

The proof is similar to what we have seen before for the general random walk. So for k=0 this is just $P\bar{1}$. And $P\bar{1}$ is just the size of B divided by n. How many vertices there are and you can be in each of them with probability 1 by n. That is the probability of the union. So it is true for k = 0.

Now the probability of being in B after one step, so k=1 case. So probability of being in B after one step is this vector, probability vector $P\bar{1}$ bar for the 0th step and then you multiply by the sub matrix which is PA. And for this probability vector again take the 1 norm.

Also matrix $(PA) = A_{B\times B}$. So when you multiply P with A or 2A, then you get the part of adjacency matrix A of interest. And that is what you multiply when you take a single step. And then generalize this 2k. That is the full proof of claim 1. After k steps, you get this expression $(PA)^k \cdot P\bar{1}$.

(Refer Slide Time: 13:31)



Notice that this $(PA)^k \cdot P\bar{1}$, you can also see this as what is $(PA)^2$. So $(PA)^2$ you can see as $PA \cdot PA \cdot P\bar{1}$ and that is also $PAP \cdot PAP \cdot \bar{1}$. I can repeat P because repeating P will still give you $P^2 = P$. So we get $(PA)^k \cdot P\bar{1} = (PAP)^k \cdot \bar{1}$

You can think of this as restricted to B random walk as matrix powering of PAP. So you multiply on the left and on the right and then you have restricted your both rows

and columns to B. And that is what will be relevant now. We will be looking at the powers of PAP. So henceforth we will study the spectral norm of this matrix.

That is the factor by which it shrinks a vector. What we will show next is claim 2. For every vector $\overline{v}$, $||\, PAP \cdot \overline{v}\,|| < (\beta + \lambda) \cdot ||\, \overline{v}\,||$. This matrix PAP shrinks every vector to something below $\beta + \lambda$. Here β is the density of B amongst all the vertices and λ was the second largest eigenvalue of A. This is a curious expression.

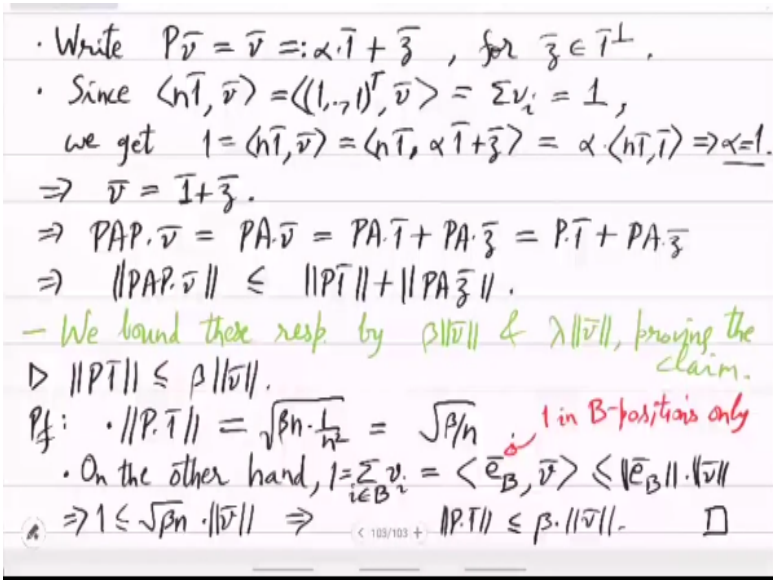
How do you get this expression? We will first club this part $P \cdot \overline{v}$ part. We will say that assume that $\overline{v}$ is supported on B which means that other coordinates in $\overline{v}$ are 0. If not then you just replace $\overline{v}$ by $P \cdot \overline{v}$ in the inequality. And notice that when you replace it by $P \cdot \overline{v}$ this RHS in the inequality cannot increase. It can only reduce, it cannot increase.

Once you have shown that result it will also imply this result, this inequality in blue. So we can assume without loss of generality that $\overline{v}$ has only coordinates in B position not outside it. This cannot increase the RHS and similarly, we can assume the coordinates to be non-negative because if there is any negative coordinate then you can make it positive and prove the above result.

That will again not increase RHS. So you can assume coordinates be non-negative and you can also assume the 1-norm to be 1. Because if it is not 1 then you can normalize the vector $\overline{v}$ by it. We have all these assumptions that we can make. With these assumptions now let us write down. Remember that in this RHS, you want β, and λ to appear. So how will that happen?

Intuitively you will get β because $\overline{v}$ coordinates are only β fraction many. And to get λ somehow you have to relate LHS to the way A acts on $\overline{1}$ orthogonal. Because you know that the action of A on $\overline{1}$ orthogonal, the amount it shrinks, that is upper bounded by λ. So you have to somehow relate with that.

(Refer Slide Time: 20:55)



So $P\bar{v} = \bar{v}$ because $\bar{v}$ is already supported on B and write it as $\bar{v}$ in two components $\bar{1}$ and $\bar{z}$. This is the orthogonal decomposition of $\bar{v}$. So α and $\bar{z}$ will be defined accordingly and uniquely. $< n\bar{1}, \bar{v} >= 1$ because $\Sigma\, v_i = 1$.

Now let us write $\bar{v} = \alpha\bar{1} + \bar{z}$. Here, $\bar{z}$ is orthogonal to $\bar{1}$, so that does not matter. Then you get $1 = \alpha \cdot < n\bar{1}, \bar{1} >$. This means $\alpha = 1$. By the assumptions on $\bar{v}$, what we have deduced is that actually $\alpha = 1$. So $\bar{v} = \bar{1} + \bar{z}$ where $\bar{z}$ is orthogonal to $\bar{1}$.

And this will be useful now to understand the action of PAP on $\bar{v}$. So we again look at $PAP \cdot \bar{v} = PA \cdot \bar{v} = PA \cdot \bar{1} + PA \cdot \bar{z}$. It is an eigenvector of A. So that is $P \cdot \bar{1} + PA \cdot \bar{z}$. Which means that you are interested in the norm of it. So the norm is certainly less than equal to the sum of the norms.

And this sum when we will work it out, you will see that the first summand will give you the promised β and the second summand will give you the promised λ. Because $\bar{z}$ is orthogonal to 1. So you can intuitively see why. And that is how you will get the expression of claim 2, which is $\beta + \lambda$. Let us do that in detail. We bound these respectively by $\beta \cdot \|\bar{v}\|$ and $\lambda \cdot \|\bar{v}\|$ and thus proving the claim.

The first is $\|P\,\bar{1}\| \leq \beta \cdot \|\bar{v}\|$. This you deduce as follows. In $P \cdot \bar{1}$, you are only looking at size of B many places and then summing up with weight 1/ n. You will get

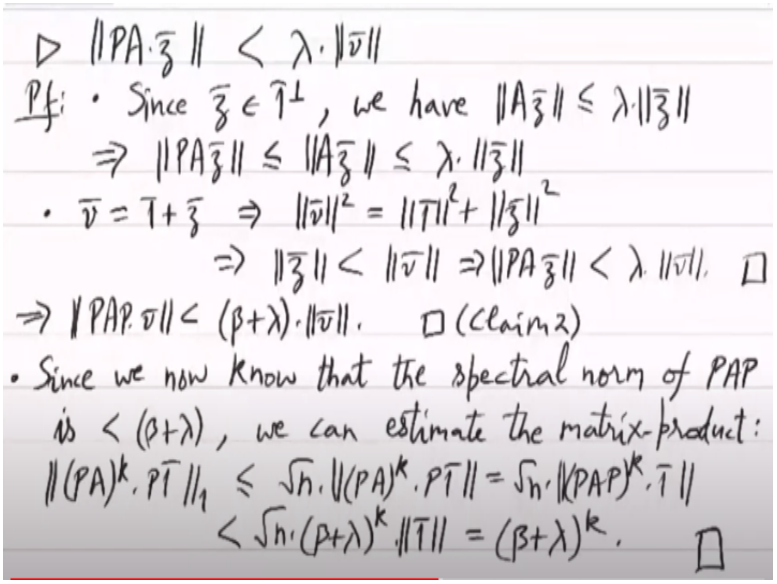
$\sqrt{\beta \cdot n \cdot 1/n^2} = \sqrt{\beta/n}$. That is what $|| P \cdot \bar{1} ||$ is exactly. But how do we relate it to the length of $\bar{v}$?

That we have to calculate separately. If we look at this the sum of the coordinates of v, this is equal to the inner product of this elementary vector or this vector $\bar{e}_B$ where 1 is in the position of B, otherwise it is 0. So $\Sigma v_i = < \bar{e}_B, \bar{v} >$. Now the inner product is less than equal to the product of the lengths.

You also know that $\Sigma v_i = 1$. You get that $1 \leq \sqrt{\beta \cdot n} \cdot || \bar{v} ||$ and v bar length we do not know. But now we have a lower bound. We know that $\bar{v}$ is at least $1/\sqrt{\beta \cdot n}$, which means that $P \cdot \bar{1}$ that we have calculated above, is less than equal to $\beta \cdot || \bar{v} ||$. You can check this. $|| P \cdot \bar{1} ||$ is $\sqrt{\beta/n}$.

And that is the same as saying v is at least $1/\sqrt{\beta \cdot n}$. We have shown that $|| P \bar{1} ||$ is small compared to $\bar{v}$ by a fraction of β. Next thing we have to show is upper bound of $|| PA \cdot \bar{z} ||$.

(Refer Slide Time: 29:05)



This we will show is smaller than $\lambda \cdot || \bar{v} ||$. Recall that $\bar{z}$ is orthogonal to 1. By the definition of λ or by the relationship of λ with the action of A, you get that $|| A\bar{z} ||$ has

to be less than equal to $\lambda \cdot ||\bar{z}||$. You can say that the shrinkage will be more than λ. We mean that $||PA \cdot \bar{z}|| \leq ||A \cdot \bar{z}||$ because P only restricts.

Length at least is not larger than $A\bar{z}$, which is less than equal to $\lambda ||\bar{z}||$. We have shown that $||PA\,\bar{z}|| \leq \lambda \cdot ||\bar{z}||$. But now what is the relationship between $\bar{z}$ and $\bar{v}$. So that is now just one step away. What you use here is that $\bar{v} = \bar{1} + \bar{z}$. Which means that $||\bar{v}||^2 = ||\bar{1}||^2 + ||\bar{z}||^2$, using the sum of square property.

From this you know that $||\bar{z}|| \leq ||\bar{v}||$, from which you know that $||PA\,\bar{z}|| \leq \lambda \cdot ||\bar{v}||$. And from both these small observations, what you have learnt is that $||PAP \cdot \bar{v}|| \leq (\beta + \lambda) \cdot ||\bar{v}||$. This proves claim 2. So what next?

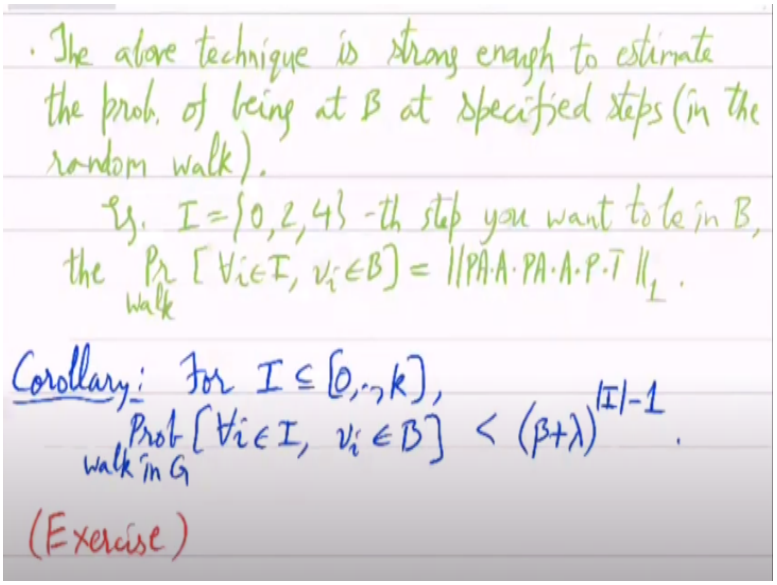
We have shown that the probability of being stuck in these bad vertices is essentially dependent on this $(PAP)^k$. We have shown in claim 2 that PAP's behavior on a vector is shrinking it $(\beta + \lambda)$. Now you can imagine how the subsequent proof will proceed. Since we now know that the spectral norm of PAP is smaller than $(\beta + \lambda)$.

Since the spectral norm is less than $(\beta + \lambda)$, we can estimate the matrix product to get the probability. So $||(PA)^k \cdot P\bar{1}||_1 \leq \sqrt{n} \cdot ||(PA)^k \cdot P\bar{1}||$. That is an exact equality. And now from the above claim, you just repeat it k times.

Every time you will get a shrinkage of $(\beta + \lambda)$. And the length of $\bar{1}$ is $1/\sqrt{n}$. This is exactly equal to $(\beta + \lambda)^k$, so as we had claimed in this theorem. We have shown that the probability of being stuck within the bad vertices is this much when you are doing a random walk.

But we are actually interested in calculating the chance that half of your steps in the random walk or more than half are stuck in bad, maybe not all but more than half. In fact, that also can be proved in a very similar way.

(Refer Slide Time: 35:31)



The above technique is strong enough to estimate the probability of being at B at specified steps in the random walk. For example, let us say you are at 0th step and then second step and maybe fourth step you want to be in B then the probability is, so then you can basically take the matrix product to be, so 1 bar is your initial vector, but 0th one you want to be in B.

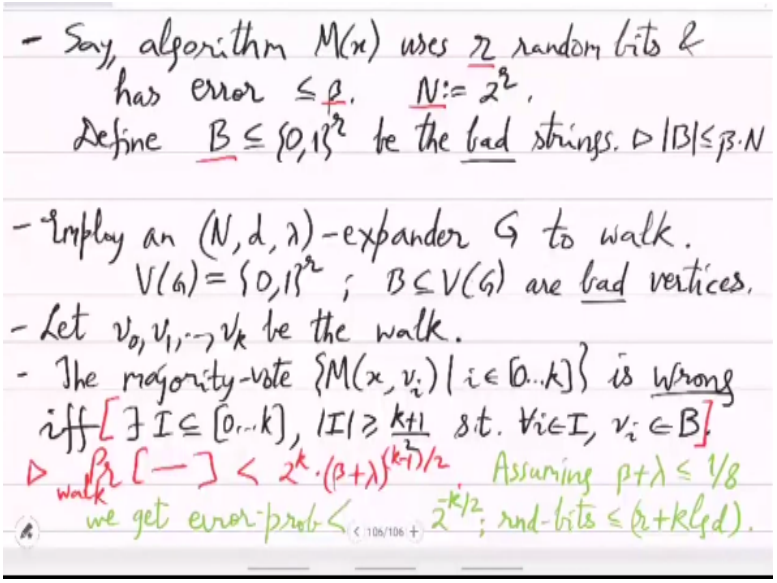
So you can multiply it with P. Then, first step you do not care. So you will multiply by A and then second step, you do care you want to be in B. Then you should multiply with PA. Third step, again you do not care. So it is anything which is A. Fourth step you care so it should be PA. And that gives the matrix product. I am claiming that these kinds of matrix products also can be analyzed in a way similar to the above.

This expression also can be analyzed and the corollary that you get is for any subset of $[0, \cdots, k]$ when you look at the probability over the walk that these vertices v_i, they should be in B. Not all the vertices in the random work, but the ones which are given in the set i, them being in B, this probability is also limited by power of $(\beta + \lambda)$. And the power will be the size of I.

When I is everything 0 to k then we showed $(\beta + \lambda)^k$. Otherwise it is the number of vertices that you want in B. So you can also give a similar proof for this. That I leave as an exercise. This is a very strong result, you can now use it to estimate the chance

that more than half of the vertices in the random walk are bad. Let us complete the application.

(Refer Slide Time: 40:04)



Say your algorithm M uses r random bits and has error less than equal to β. In the space of $\{0,1\}^r$, β fractions are bad. Define this B to be a subset of $\{0,1\}^r$ be the bad strings. And you know that the size of B is less than equal to $\beta \cdot N$, where $N = 2^r$. In the whole space of r bit strings beta of them are bad, that is the set B. What kind of an expander should you employ here looking at the probability estimate?

Employ an (N, d, λ) −expander G to walk where this $(\beta + \lambda)^k$ should be small enough so that the probability estimate works out. We are thinking of the vertices as $\{0,1\}^r$ strings and B are the bad vertices. When you are doing a random walk, you want to hit these vertices at most half or smaller than half the time. Let $v_0, v_1, \cdots, v_k$ be the walk.

Now the majority vote when you run this algorithm M on v_i, this majority vote is wrong if and only if at least half of these were, v_i. So the size of I is at least half, $(k+1)/2$. That is the only case when the majority vote fails. Otherwise the majority vote would have been correct if bad vertices were few.

The probability of this happening in the random walk, so probability of this event that there is an $(k + 1)/2$ size set I such that for all i you are stuck in v_i, is what we have to compute. So possibilities of i is it does not exceed obviously the number of subsets which is 2^k. And once you have fixed I, you use this theorem that we proved. That is $(\beta + \lambda)/^{(k-1)/2}$.

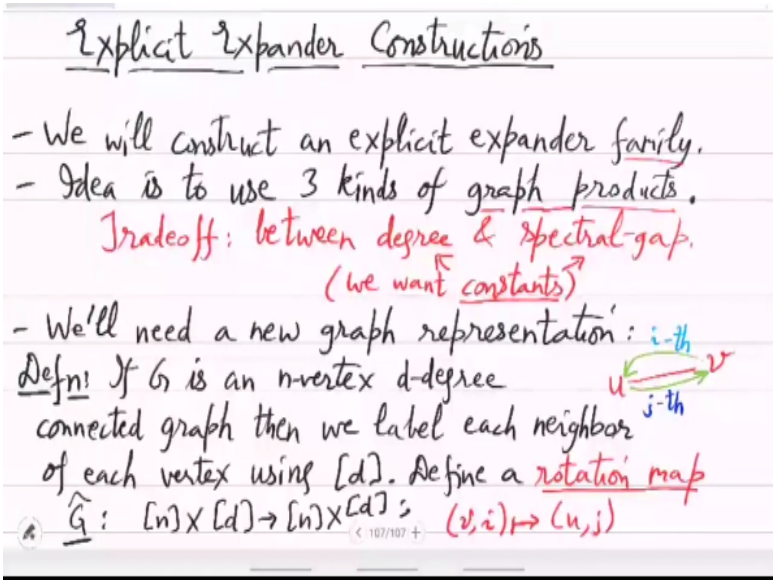
So $(k + 1)/2$ is the size of I, then -1 of that, you get $(k - 1)/2$. That is the error probability that you use the explicit expander, run M x take the majority vote and still you get a wrong answer. That probability is limited by this expression. You just have to take $(\beta + \lambda)$. Assuming $(\beta + \lambda)$ to be let us say less than equal to 1/8. Which you can ensure.

You can easily assume that the β was the error in the algorithm and it was let us say less than 1/16 and λ also is less than 1/16 for the expander graph. You will get this upper bound 1/8. We get error probability less than $2^{-k/2}$ for random bits.

Random bits that you need is $r + k \log d$. So for roughly r + k random bits you are getting an error of $2^{-k/2}$ assuming that you have a good enough expander. That is what we have shown. This finishes some major properties of expansion and how it can be used in general. For general algorithms, you can use expanders and very large expanders as long as they have constant degree and they are very explicit.

For every vertex the neighbor can be found easily. Now how reasonable is it to expect these explicit expanders to exist? How can we construct them? How can we find them? So this is what we will focus on next. The new topic that we begin is still within expanders. But now we will want explicit expanders, constant degree, very large, but very few neighbors and easy to find.

(Refer Slide Time: 48:09)



The nice thing about the methods in this topic, in this chapter will be that ultimately you can start with any undirected graph and once you do these operations in a certain sequence, it will become an expander. It is not only a construction to get big, explicit expanders, but also to convert any graph into an expander. It will have multiple applications.

You can do what we just saw before this success probability boosting, minimizing the number of random bits. But you can also solve connectivity in logspace. We will see that application later. We will construct an explicit expander family. So remember that when we construct an expander, we mean infinitely many expanders. It should be constructable for any size.

It should not be that you construct only one graph. That would have little to no utility. You actually need to have somebody give you a parameter, let us say the size of the graph, and then you have to construct an expander bigger than that. It is an infinite family construction. And there are three main algorithmic operations, which will be used. Idea is to use three kinds of graph products.

These operations, we will call them products, because in some sense, they will be multiplying two graphs. But the definition of multiplication will be different. It may be powering a graph or just matrix multiplication for the adjacency matrix. Second type will be tensor product and third type will be a combination of both, which is called zig-zag product.

And there will be a tradeoff between degree and spectral gap. The tradeoff means that in these operations if you want the spectral gap large, you can achieve it by increasing the degree. But you cannot afford too big a degree because you want the degree to be constant. And you also want the spectral gap to be constant, both of them you want them to be constants.

We have to keep a balance so that both of them remain constant and the spectral gap increases as much as possible. And the degree is minimized as much as possible. We will need a new representation of graphs. You already know two graph representations. One is the vertex edge drawing or the actual diagram of the graph. Second is thinking of it as a 0, 1 matrix, which is an adjacency matrix or normalized adjacency matrix which is a stochastic matrix.

And third is via something called rotation map. So you can think of it as follows. Vertex u and vertex v, these are two vertices and they are neighbors of each other. You can think of v as a neighbor of u and v as a neighbor of u. There is a single red edge, undirected edge, which makes them neighbors of each other because it is an undirected graph.

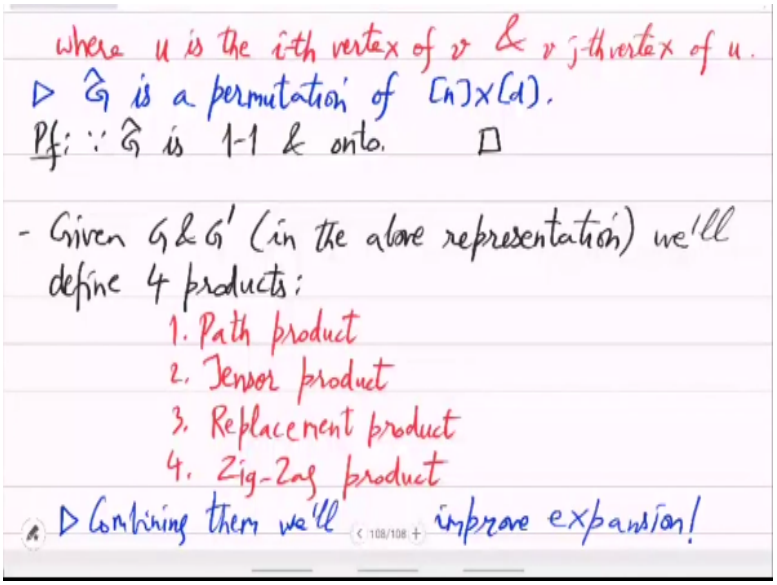
When you think from the point of view of u, v is let us say it is the i-th neighbor of u. And when you think from the point of view of v, u is the j-th neighbor. Basically we want to store, for every vertex u we want to store the neighbors in a certain order. And since you will do this for all the vertices, u, i will take you to v and then from v, j will take you to u. That is the representation.

If G is an n vertex d-degree connected graph. We will look at connected graphs only, and then connected graphs or operations will ultimately make it an expander. That is the plan. Let G be a connected graph n-vertices d-degree. Then we label each neighbor of each vertex using number 1 to d. And then define a rotation map.

We call this $[n] \times [d]$ map rotation map $\hat{G}$. It will send $(v, i) \rightarrow (u, j)$. So you should read it as u is the i-th and v is the j-th neighbor.

This is the natural information associated with an edge. All the four things will be related like this and we will think of $(v, i) \to (u, j)$ as a map $[n] \times [d]$ on the domain $[n] \times [d]$.

(Refer Slide Time: 57:44)



Where u is the i-th vertex of v and v j-th vertex of u. Now note that $\hat{G}$ is a permutation on $[n] \times [d]$. Why is that? Why is it a permutation? It is a permutation simply because the image of (u, i) is exactly one pair. And if you look at the (v, i)'s image is unique and the preimage of (u, j) is also unique, simply because of this picture. It is obvious from the diagram.

So that is why it is 1-1 and onto-map. It is onto because of connectivity. Some weak form of connectivity is needed. Because of that, it is both 1-1 and Onto. So given G and G' in the above representation, we will define four kinds of products.

The first will be the matrix product or path product. Second will be the Tensor product. The third one will be the Replacement product. And related to the replacement product, we will also have a fourth Zig-Zag product. We will define these things next time. And then we will use these 4 products to improve expansion. Combining them we will improve expansion. They will have different strengths and weaknesses and in combination they will work very well.