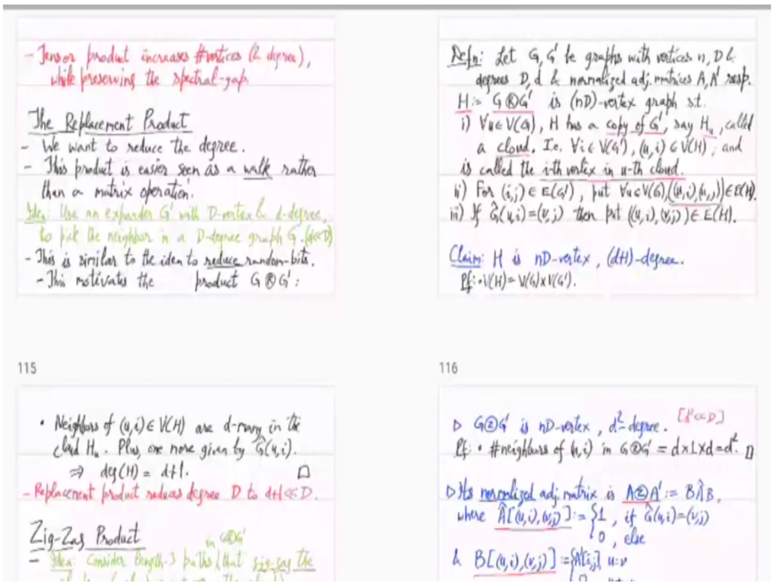
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology - Kanpur

Lecture – 15
Explicit Expander Construction

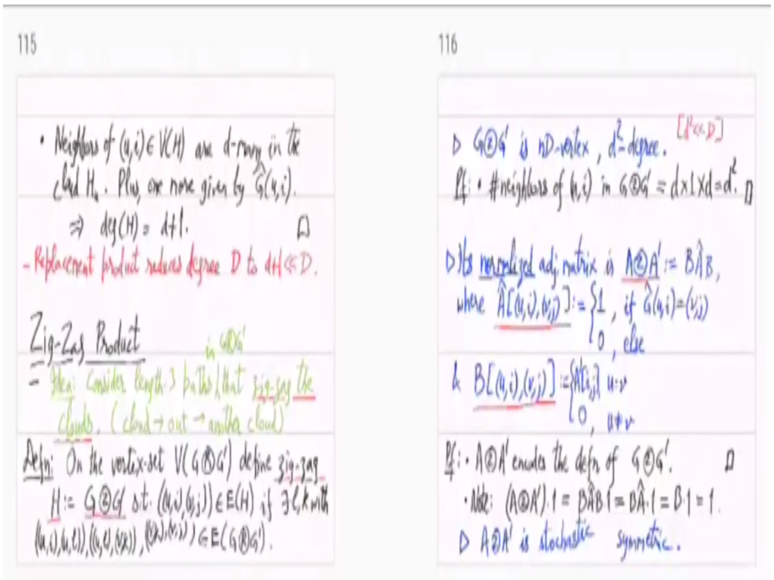
(Refer Slide Time: 00:18)



Last time we started the zig-zag product. For that let us recall the replacement product which is used to reduce the degree that has grown because of the earlier products. What you do to reduce the degree is you take this replacement product with a graph G' where in G' the number of vertices is equal to the degree of G and the degree of G' is much smaller. So we used big D and small d.

And then what you do is whenever you have to walk or pick a neighbor in the graph G, you make that decision by walking in GG', that was the main intuition and then there was this extended definition and all. That gave us the replacement product. Basically, its size grows to $n \times D$ and the degree becomes originally it was D, now it becomes d+1. So, the degree falls drastically.

(Refer Slide Time: 01:33)

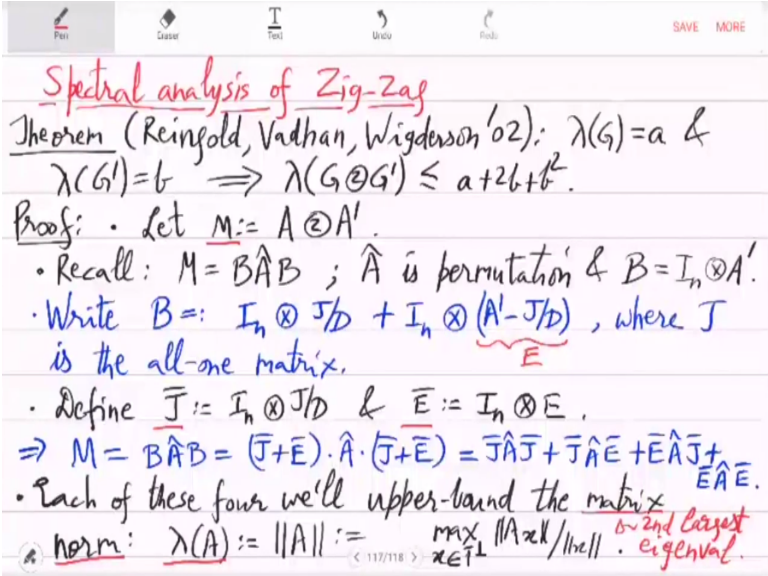


Based on that now remember that we could not do or we did not do the spectral analysis for the replacement product. We will do it for a different product which is called the zig-zag product. So, in the zig-zag product you basically look at length-3 paths, it is kind of cubing the replacement product, but you did this in a zig-zag way. You zig in the cloud, then zag out of the cloud and again zig in the cloud.

It is a kind of zig-zag zig, 3 steps that gives you the zig-zag product which is denoted as GⓩG'. Here the number of vertices is the same as $n \times D$ and degree grows slightly to d^2 but d will be much smaller than D. So, this is still much smaller than D and we had these formulas. Now the normalized adjacency matrix of the zig-zag product this A ⓩ A' is $B\hat{A}$.

Where $\hat{A}$ is actually the step you take outside the cloud so that is dependent on A and B is the step you take inside the cloud so that is dependent on A'. It is just so $\hat{A}$ and B are just blown up versions of A and A'. Then you multiply the 3 so you get $B\hat{A}B$, that is the new adjacency matrix normalized. So, A ⓩ A' is both stochastic and symmetric that we had observed last time.

(Refer Slide Time: 03:45)



Now we will do the spectral analysis. The major theorem is due to Reingold, Vadhan, Wigderson. This theorem says that if the second largest eigenvalue of G is let us say equal to a and that of G' is b, then $\lambda(G \circledZ G') \leq a + 2b + b^2$. To understand the meaning, think of a and b as small and also remember that they are both fractions below 1.

So, b^2 is quite small so this is like a + 2b. Think of this as something additive. So, if you started with small a and small b, then this spectral norm is small, it remains small and the advantage of the zig-zag product is that degree also has fallen. The earlier products were increasing the size of the graph, this product decreases the degree. Moreover, it keeps the spectral gap intact so that is the import of this theorem. Let us prove it now.

Let us call the product A ⓩ A' as M. That is the zig-zag product and we will now study the spectral norm of this matrix. Recall that $M = B\hat{A}B$ where $\hat{A}$ is a permutation matrix dependent on G for A and B is dependent on A'. Furthermore, B is a tensor product. It is just a blown up version of A'. It helps you move in the cloud, one step in the cloud and then go out by $\hat{A}$ and then again move in the cloud by B.

Let us do the following. We will write M in a different way and that will start by rewriting the B matrix. You first break up A' by this all-one matrix J. So J by D we take out and what remains is $A' - J/D$ where J is an all-one matrix and this $A' - J/D$ we will call E. So why did we do this? We are doing this so that we separate B into two kinds of matrices.

The first matrix is expected to have a larger spectral norm and the second one smaller, which is what we want. We want to break it up into large, non-small norms and then we will bound the small norm separately. Define $\bar{J}$ to be the first part and $\bar{E}$ to be the second part. We have broken B into $\bar{J}$ and $\bar{E}$. As I said, the idea is that $\bar{J}$ is kind of the main term and $\bar{E}$ is the error term in terms of spectral norms and then we will try to bound the error.

What this means is

$M = (\bar{J} + \bar{E}) \cdot \hat{A} \cdot (\bar{J} + \bar{E}) = \bar{J}\,\hat{A}\,\bar{J} + \bar{J}\,\hat{A}\,\bar{E} + \bar{E}\,\hat{A}\,\bar{J} + \bar{E}\,\hat{A}\,\bar{E}$.

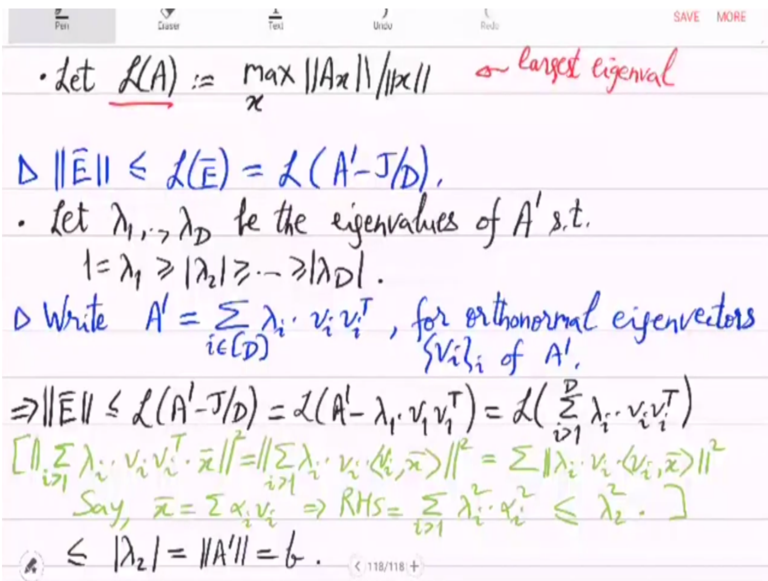
There are these 4 terms when you multiply out. Remember that this multiplication is possibly noncommutative. These four terms could be different. The last three we will consider as the error, first as the main.

Each of these four will upper bound a matrix norm first matrix norm or the spectral norm which is $\lambda(A)$ for a matrix, we will denote it by $||\, A \,||$ which is further defined

$$||A|| = \max_{x \in \bar{1}^{\perp}} ||Ax|| \,/\, ||x||.$$

or in words what is the maximum shrinkage that is what this norm measures of action of A on unit vectors orthogonal to 1. And we will also use another norm. This is to do with the second largest eigenvalue.

(Refer Slide Time: 13:11)



There is another norm. Let $L(A) = \max_{x} ||\, Ax \,|| \,/\, ||\, x \,||$. So, the shrinkage of unit vectors by A over all the vectors, this is the norm, this is another norm, this will be bigger than the previous one. Now this is related to the largest eigenvalue and the previous one was the second largest eigenvalue of A. These are the two norms of interest. We are interested in the gap between the two.

Observe that for this error matrix $\overline{E}$ matrix norm or spectral norm is at most the L-norm by definition and $\overline{E} = A' - J/D$, we want to upper bound this. Let, $\lambda_1, \cdots, \lambda_D$ be the eigenvalues of A' such that they are ordered. A' comes from G' and that was a graph with a number of vertices D. So, it has D-many eigenvalues starting from 1.

We are interested in how small lambda 2 is the magnitude. Since these are eigenvalues you can write A prime in terms of them and eigenvectors. So, write A prime as lambda i v i v i transpose i 1 to D or orthonormal eigenvectors of A prime. So, this I leave an exercise. This is a standard fact from linear algebra that that A prime has well it has real eigenvalues, then it has real eigenvectors.

And in fact, it has eigenvectors which are unit and orthogonal which means that they form an orthonormal basis. With respect to that you can actually write A prime like this sigma lambda i v i v i transpose. Aet us use this expression to bound norm of E bar. So, A prime – J by D. Now notice that v 1 is just all coordinates 1 over square root d, right. So v i v i transpose is actually J by D and lambda 1 is 1.

So, this is equal to the L norm of what remains which is lambda i v i v i transpose for i 2 to D. And this one how do you upper bound this? So, notice that sigma lambda i v i v i transpose when you act it on a unit vector x, so what does it become? So, it becomes lambda i times v i times this number real number v i, x bar and you are interested in the norm of this. And remember that v i's are orthogonal. So, this is in fact equal to norm square.

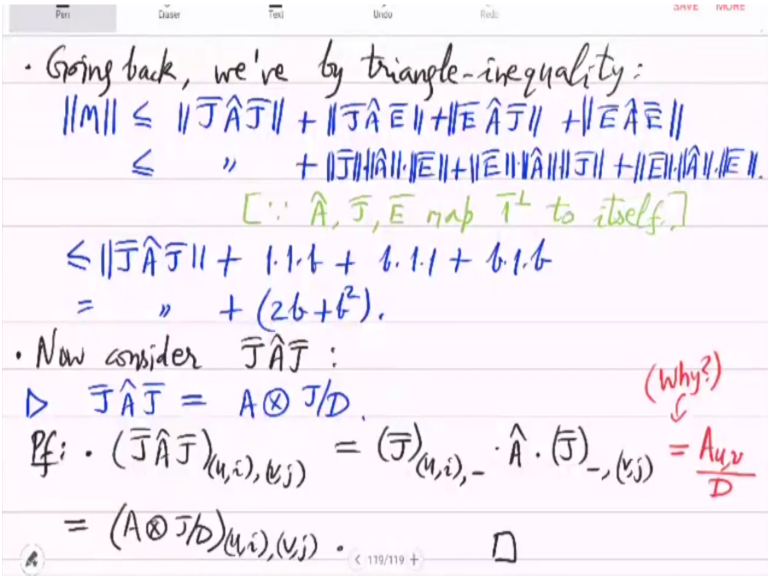
So, what we will use here is that x bar you can write as a combination of this orthonormal basis. So, say x bar is alpha i v i and when you do this, then the above thing is the same as v i, x bar. So v i, x bar will only extract alpha i. So, you will actually get this that is the point. So, you actually when you take inner product of x bar with v i all the summons inner product vanishes except that of v i which gives you alpha i and also remember that v i is a unit vector.

So, you are left with actually sigma lambda i square alpha i square and sigma alpha i square is actually 1. So, this expression is at most the highest lambda i which is lambda 2. So that is what we get, we get that this sigma lambda i v i v i transpose to the action of A prime on x bar this norm is at most value of lambda 2. So, we go back, we know that that is at most the value of lambda 2 which is the matrix norm of A prime which is equal to b.

So, we have upper bounded the matrix norm of E bar in this way. So, let us now see where we are. We broke up M by breaking up B and then multiplying. So, we had these 4 terms $||\bar{J}\hat{A}\bar{J}|| + ||\bar{J}\hat{A}\bar{E}|| + ||\bar{E}\hat{A}\bar{J}|| + ||\bar{E}\hat{A}\bar{E}||$.

We have an upper bound on the matrix norm of $||\bar{E}||$. Let us apply this to the sum.

(Refer Slide Time: 21:58)



Going back we have by triangle inequality the matrix norm of M which is defined by the norm of a vector the action of M on x. On a vector norm you can apply triangle inequality, hence you can apply it on this matrix norm also. You will get that this is less than equal to the norm of the first term which is

$$||M|| \leq ||\bar{J}\hat{A}\bar{J}|| + ||\bar{J}\hat{A}\bar{E}|| + ||\bar{E}\hat{A}\bar{J}|| + ||\bar{E}\hat{A}\bar{E}||.$$

And then you use the multiplicative property of norm which is that the norm of product is less than equal to product of norms. So, you will get, first one I do not change I keep it the same, but second one I change to that plus similar things.

$$||M|| \leq ||\bar{J}\hat{A}\bar{J}|| + ||\bar{J}|| \cdot ||\hat{A}|| \cdot ||\bar{E}|| + ||\bar{E}|| \cdot ||\hat{A}|| \cdot ||\bar{J}|| + ||\bar{E}|| \cdot ||\hat{A}|| \cdot ||\bar{E}||$$

So, first the main term and then the error term I have written as sum of products. And I just point out that this we can do since $\overline{E}, \hat{A}, \overline{J}$ they map $\overline{1}$ 1 bar orthogonal to itself.

They preserve this space orthogonal to 1. Hence by the definition of matrix norm you can easily see that this multiplicative property is there. And now I can write this as again upper bound, the main term I do not analyze now, I am postponing that, but these error terms I will use the trivial bound. So, for $\overline{J}$ matrix norm it cannot exceed 1, same for $\hat{A}$ norm, $\overline{E}$ I have shown to be b at most. Let us use this approximation or upper bound.

So, I get

$$||M|| \le ||\overline{J}\hat{A}\overline{J}|| + 1 \cdot 1 \cdot b + b \cdot 1 \cdot 1 + b \cdot 1 \cdot b = ||\overline{J}\hat{A}\overline{J}|| + \left(2b + b^2\right).$$

Let us now estimate the main term. So, now consider $||\overline{J}\hat{A}\overline{J}||$. Let us first consider this matrix. What we will show is that this matrix is, of course, coming from A, but the exact relationship is a tensor product.

Remember $\overline{J}$ is tensor product identity with J/D and $\hat{A}$ is the permutation coming from A, coming from graph G. It is not very surprising that when you multiply $\hat{A}$ with $\overline{J}$ on left and right, then you again get a tensor product. Let us just check that.Let us check the u, i-th v, j-th entry of this matrix. You have to recall how the rows and the columns of $\hat{A}$ were indexed.

They were indexed by using this rotation map of the graph G. Let us look at row u, i-th and general column v, j-th. This entry is following matrix multiplication

$$\left(\overline{J}\hat{A}\overline{J}\right)_{(u,i),(v,j)} = (\overline{J})_{(u,i),-} \cdot \hat{A} \cdot (\overline{J})_{-,(v,j)} = \frac{A_{u,v}}{D}.$$

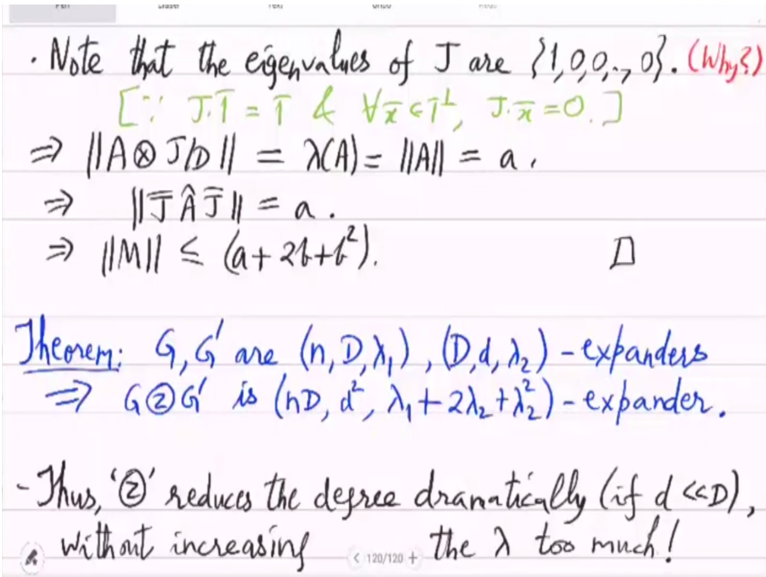
These entries are just 1/D, I mean the kind of the block diagonal in the block matrix representation it is 1/ D in the block diagonal locations, other locations it is 0.

What you will get when you multiply these two things to $\hat{A}$ is that you will get actually the u, v-th entry of $\hat{A}$, so there is a 1/D factor which gives you $1/D^2$, but then there will be a multiple of D also. You will actually get 1/D. You will get the u, v-th entry of the matrix A divided by D just check this. It follows from the definition, just have to go back and compare the definitions.

And $A_{u,v}/D$is actually the RHS matrix entry. If you look at the u, i-th; v, j-th entry of this tensor product A with J / D, then you will see that it is just $A_{u,v}/D$, this factor 1 by D. Remember that in this tensor product what is happening is the u, v is actually indexing A and the i, j is indexing J. So this i, j goes away, you just get $A_{u,v}$. So, you can check this and that shows that $\bar{J}\hat{A}\bar{J}$ is actually quite simple to begin with.

It is just A tensor all-one matrix and normalized by D. We want to understand the spectral gap of this matrix $A \otimes J/D$ and it will boil down to the spectral norm or matrix norm of A so that is how you will get small a. Let us just finish that part.

(Refer Slide Time: 31:19)



Note that the eigenvalues of J are the largest one is of course 1, everything else is 0. This is a small exercise that you can do and of course this 0 is repeated. So, overall the number of eigenvalues has to be the dimension of this J. You know the eigenvalues of J and you know the eigenvalues of A, so you know the eigenvalues of the tensor product. So, the J times all-one vector is all-one vector and for every vector orthogonal to 1, $J \cdot \bar{1} = 0$.

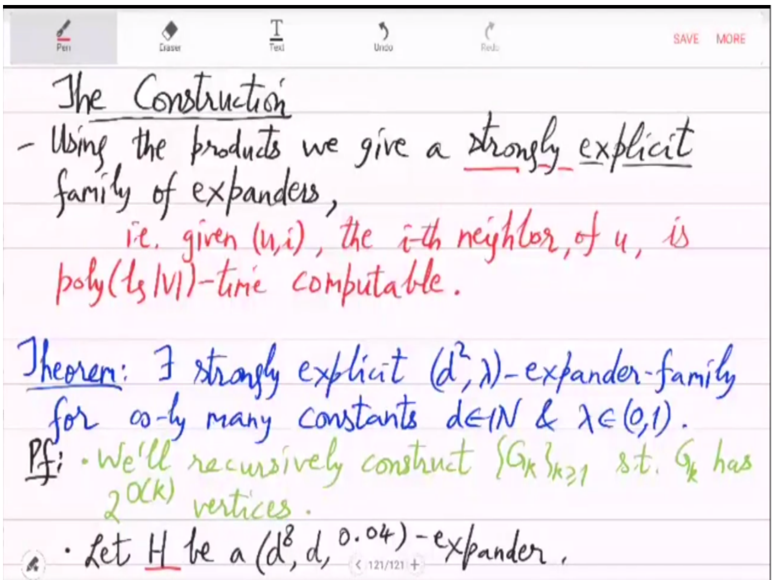
So, 1 is an eigenvalue and then the whole space orthogonal to 1 is an eigenspace. Which means that you understand the overall eigenspace of J and the eigenvalues are then just this 1 and then all the zeros repeated. This means that $|| A \otimes J/D ||$ is at most whatever the second largest eigenvalue was of A that will remain unchanged because it is only being multiplied by 1 or 0.

Hence overall the second largest eigenvalue will remain that of A which is equal to a. And so we know now that $||\bar{J}\hat{A}\bar{J}||$ norm is a. This means that going back the matrix norm or spectral norm of M is at most $(a + 2b + b^2)$. This finishes the proof of the theorem by Reingold, Vadhan, Wigderson and it gives you a very good understanding of the spectral gap in the case of the zig-zag product. Let us write that down in terms of algebraic expansion.

So G, G' are (n, D, λ_1), (D, d, λ_2) $-$expanders. Then for the zig-zag product as the previous theorem showed the algebraic expansion will be there. And the other parameters so the number of vertices is multiplicative nD. The degree is d^2 and the spectral norm is $\lambda_1 + 2\lambda_2 + \lambda_2^2$.

So, this λ_2 we have to pick small enough so that $\lambda_2 + \lambda_2^2$ is not that much. Thus, this zig-zag product reduces the degree dramatically, assuming that d is much smaller than D without worsening the expansion, so without increasing the λ too much. That is the point of a zig-zag product. This last product together with the earlier products gives us all the tools to construct expanders that are explicit and the expansion or spectral norm is constant. Let us finish that.

(Refer Slide Time: 37:13)



Using these products we give a strongly explicit construction or family of expanders. Now what is this strongly explicit business? The i-th neighbor of u is computable in poly log $|V|$ time. The bits needed to describe u is $\log n$ or log $|V|$ and the bits needed to describe i is they are only d neighbor so that is log d which is constant and in terms of this input size u, i, this is polynomial time algorithm.

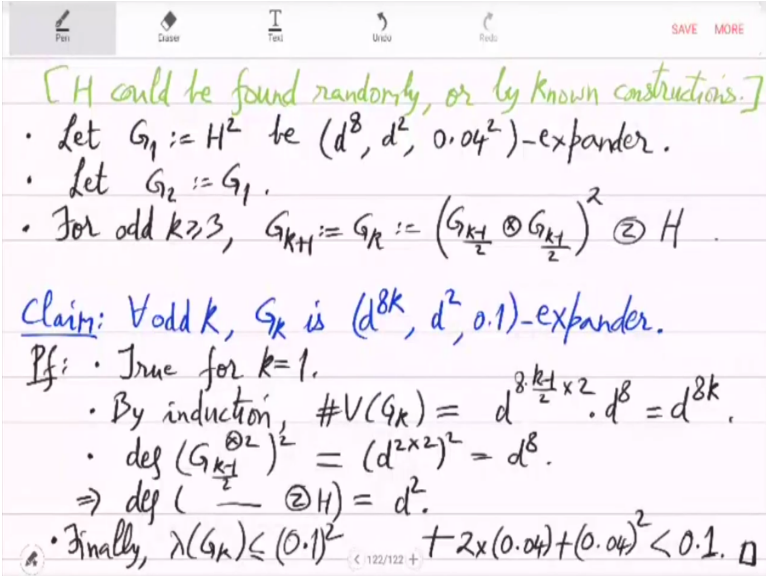
So that is strongly explicit. In terms of this time actually the graph size of V can be exponential. So, it is a very large expander, it is a growing family, degree is constant, expansion spectral norm is constant and it is very explicit, polytime explicit. So, there exists strongly explicit d square lambda expander family for infinitely many constants d and lambda. So, d being natural numbers and lambda being a positive fraction less than 1.

Now further away it is from 1 the better expansion. So, now we just have to give you the sequence of products to do. In the base case you just start with some constant sized expander and I mean even if you just pick a random connected graph it will be of that type, it will be a constant sized constant spectral norm and starting with that to make it bigger you do tensor product and to improve expansion you do path product.

And by that point, the degree would have grown, so to reduce the degree you do zig-zag product that is the sequence. So, we will recursively construct expanders G_k such that G_k has 2^k vertices. So, remember that graph will be expanded will be very large. It will have these 2^k many vertices but everything the complexity, the time complexity will be polynomial in k.

So, it is basically a recursive construction, the k steps you will get this G_k and k steps kind of. So, what are these steps exactly? So, let H be a d^8 sized, d degree, 0.04 spectral norm expander.

(Refer Slide Time: 43:04)



So, this you find by each could be found either randomly or by known constructions. This is not the hard part, this is all easy to do. You can find H because it is constant sized constant expander. Then you start the process the k steps that will give you G_k. Let G_1 be H^2 that is $H \times H$ path product, this you know is vertices will not change so d^8 many degrees will square, but the expansion will also improve drastically so that will also be squaring, right.

So this G_1 is a very good expander but the problem is that it has constant size, we have to grow that. Also, for technical reasons we will keep $G_1 = G_2$ equal and then G_3, G_4 equal and so on. So let G_2 remain G_1. For odd k, the definition is G_{k+1} is the same as G_k will do the construction recursively. So, you actually go down to k–1 by 2 and take tensor product with itself.

So when you go to $G_{(k-1)/2}$, obviously by recursion or by induction that was already constructed so you have access to it although it is a very big graph but you have access to the algorithm that produces it. So that algorithm you can actually do tensor product and get a different algorithm to get this tensor product graph or expander. Do that, then you do the path product, square it. Now the degree has blown up so reduce that by zig-zag with H.

So H has degree d, so this thing in the end will have degree little d^2 that is how you do it. So, the claim is that for all odd k, G_k is d^{8k}, d^2, 0.1 expander. So, d remember is a constant, so d raised to 8k is like $2^{O(k)}$. It's exponential in k sized graph, degree is d^2 so it is constant and expansion is 0.1 spectral norm and this is true for every odd k.

Every other one is equal to the prior one, but overall as k grows this is giving you a bigger and bigger expander, still constant degree, degree the same, expansion the same upper bound. So, in the base case we have shown this it is true for k = 1 by definition that was H^2. We have shown d^8 , d^2, $(0.04)^2$ so it is even better than 0.1. Let us now do induction. So, what is the number of vertices in G_k?

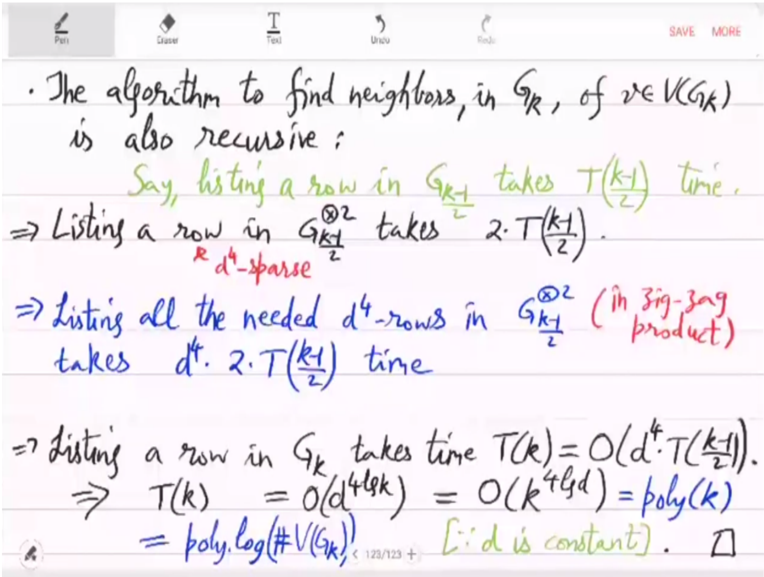
So, you have to recall the three products; tensor product, path product, zigzag product and how the size of the vertex set grows. So, d raised to 8 K–1 by 2 is the by induction hypothesis on G K–1 by 2. Tensor product will square it, path product will not change it and zig-zag product will multiply it with d raised to 8 vertex set of each which is also d raised to 8. So that is d raised to 8K. So, this checks out this is as promised. What is the degree now?

So, degree of this part this tensor square and then a square let us start with the degree of $G_{(k-1)/2}$ by induction it is d square. Then tensor product will make again square and the path product will make it another square so that will become d^8. So this much squaring tensor squaring and then path squaring or matrix squaring this is giving you degree d^8, so it has blown up, you want to bring it back to d^2 so you take the zig-zag product.

This implies that the degree of this zig-zag with H that will be down to d square. So that also checks out finally the expansion. So, lambda G_k is at most, so what is the lambda of this tensor square squared? Tensor square does not change lambda so that is the same as 0.1, but when you do path product it improves so that is $(0.1)^2$. That is the first thing plus you had a square + 2b + b square.

So now a square is done, 2b is 2 times lambda of H, what is that? Lambda of H was 0.04 plus the square of it. So, this you can see is just below 0.1 so that also checks out. So, G_k as claimed is d^{8k} vertex, d^2 degree, 0.1 expander. So, now we have an infinite family of expanders, expansion is constant or 0.1 and degree is constant and the size of the expander is exponential in k. But now we have to check strong explicitness, how explicit is it?

(Refer Slide Time: 51:50)



So, the algorithm to find neighbors in G_k of a vertex V is recursive that is not surprising because the definition of G_k itself was recursive, so just follow that recursion algorithmically. So, say listing a row in $G_{(k-1)/2}$, now this row may be very long because $G_{(k-1)/2}$ is an exponential sized graph, but remember that any row has very few ones, everything else is 0, so non-zero entries are very few because that is by the degree of G_{k-1}.

And the degree of $G_{(k-1)/2}$ is just d^2 so there is only d^2 which is constant many non-zero entries. So, listing those entries those positions take let us say $T((k-1)/2)$ times and using this subroutine you can get the expression or the even the algorithm recursive algorithm to list a row in G_k, $G_{(k-1)/2}$ first we see tensor with itself listing a row there takes which will again be a sparse row, just it will have how many?

It will have at most d^4 non-zero entries. So, listing a row there will take twice the time because so to check this you have to actually think about how tensor product is defined. A row will be defined by a pair of indices one for the first factor and second for the second factor and so you basically then find first, I mean row in the first component, row in the second component and then combine to get the full row.

So, it will be you just have to call this subroutine two times. It is $2 \cdot T((k-1)/2)$. This means that listing all the rows in, so now this is for the zig-zag product with H, to take the zig-zag product you have to, so let me remind you that this row has d^4 sparsity, only d^4 entries are non-zero. When you are taking zig-zag product with H remember you have to zig inside the first cloud zag and then zig inside the second cloud.

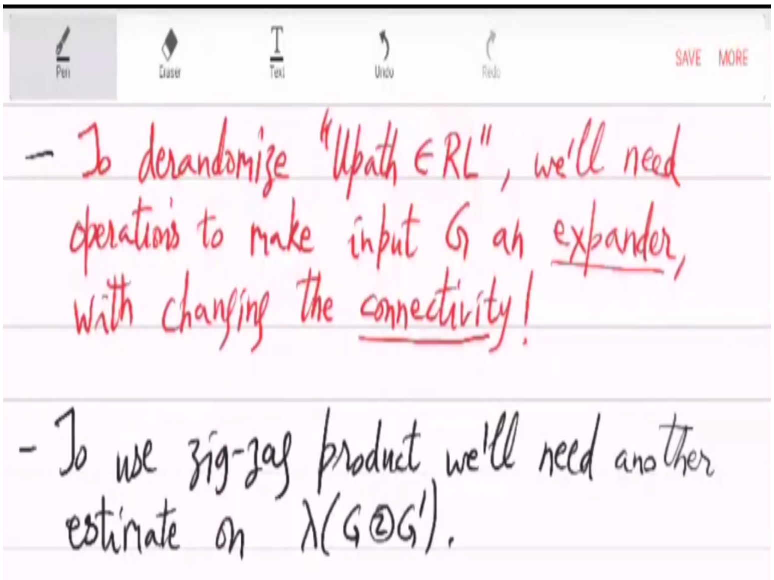
So, you have to actually do this thing twice, you have to compute then all the rows which are required. There are d^4 rows which will be required in the other square in the zig-zag product. This will take $d^4 \cdot 2 \cdot T((k - 1)/2)$ time. This is what it takes to list a row in G_k. So listing a row in G_k takes time $T(k)$ which we have just shown is $G_k O(d^4 \cdot T((k - 1)/2))$.

In big O we are using absolute constant, basically this 2 so this is correct in terms of d as a function of d that much time it takes to list a row in G_k. Now in the RHS you are halving k, so how many times can you half k? Log k many times right. So, d^4 will be multiplied log k many times. You will get that $G_k T(k) = O\left(d^{4\log k}\right)$ that is the time complexity which is the same as $G_k k^{4\log d}$.

So, the point is that since d is a constant this is only polynomial in k which is polynomial in $\log\ 2^k$ since d is constant. In terms of number of vertices this is polylog. The time complexity is polylog and you can list a row that you want and row will have few entries. This is how it is proved. Yes, so that finishes the strongly explicit construction of expanders with constant spectral norm.

The last application that we will do important result that we want to show is given a general graph in the input can you make it an expander by doing certain operations and without changing the connectivity? Somehow remembering the connectivity.

(Refer Slide Time: 01:00:45)



To de-randomize undirected path connectivity this result that we had randomized algorithm, so to derandomize this we will need operations to make input graph G an expander without changing connectivity. You want an expander but you also want to remember something about the connectivity of S and T, source and target vertices in G. To use this zig-zag product, we will need another estimate on lambda G zig-zag G'.

The previous estimate will not work because there both a and b were small but if your G was given in the input, you do not know how small it is. It can only be very close to 1. So, we will use a different estimate. We will prove and use a different estimate next time.