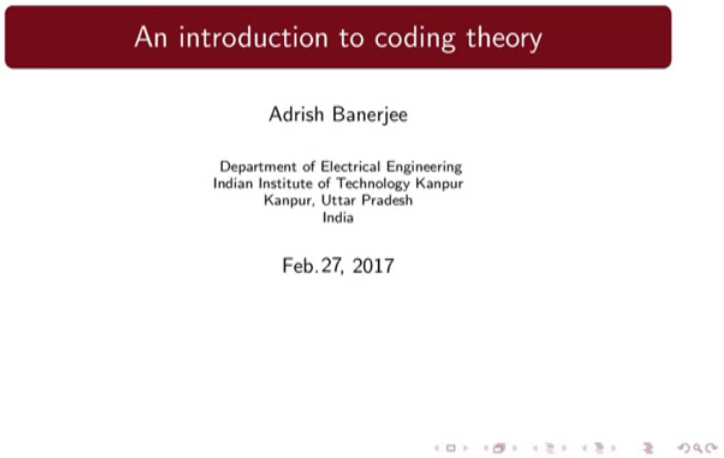
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 06
Lecture Number 24
Low Density parity check codes

(Refer Slide Time 00:13)

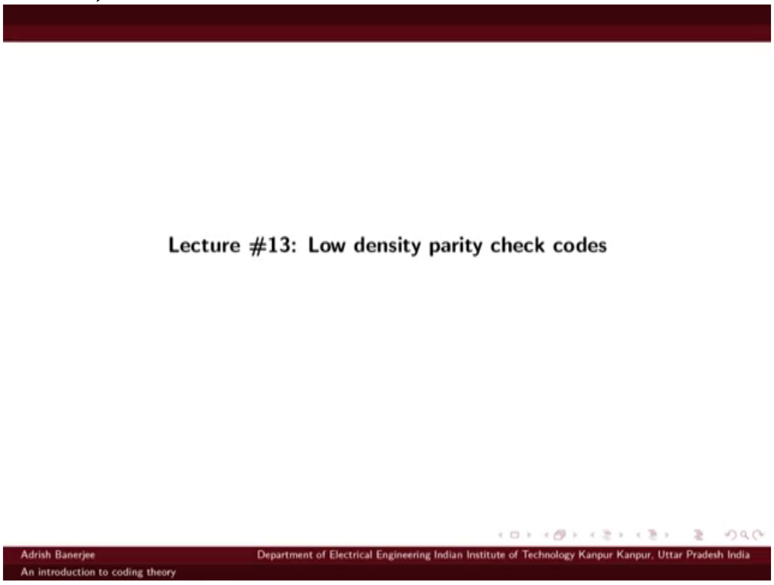


Today we are going to give a brief

(Refer Slide Time 00:16)

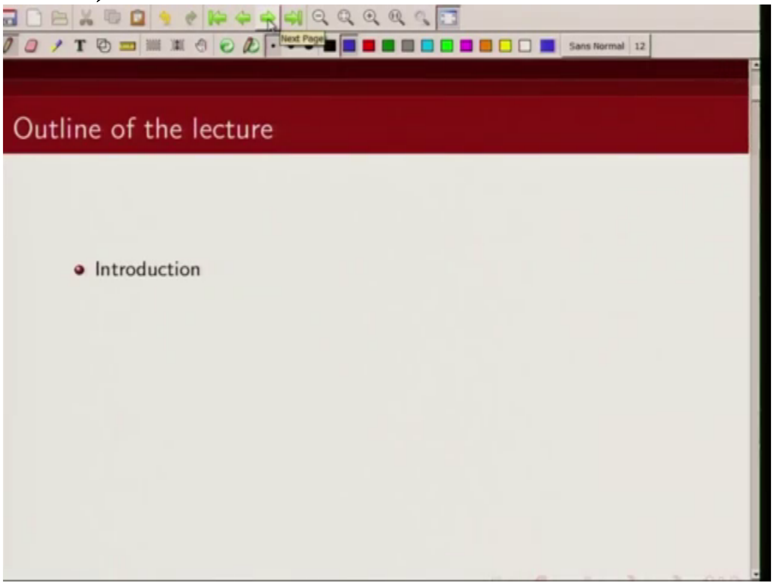
introduction to low density parity check codes.

(Refer Slide Time 00:19)



So we will start off with

(Refer Slide Time 00:22)

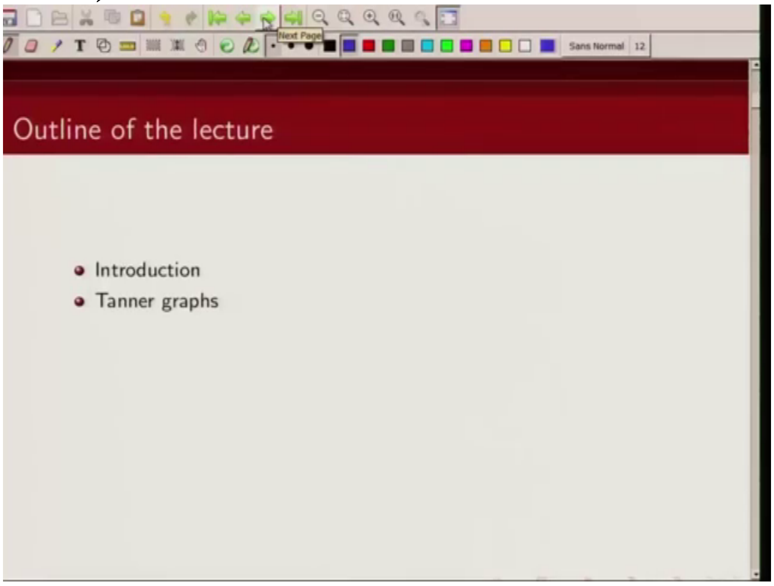


a very basic definition of what do we mean by a low density parity check

(Refer Slide Time 00:27)

matrix, what do we mean by low density and then

(Refer Slide Time 00:31)

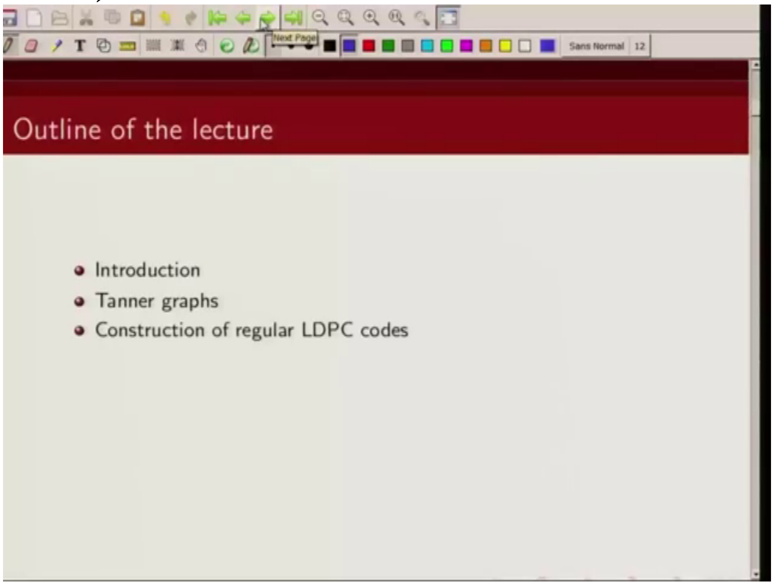


we will show how we can write the parity check matrix using a bipartite graph

(Refer Slide Time 00:38)

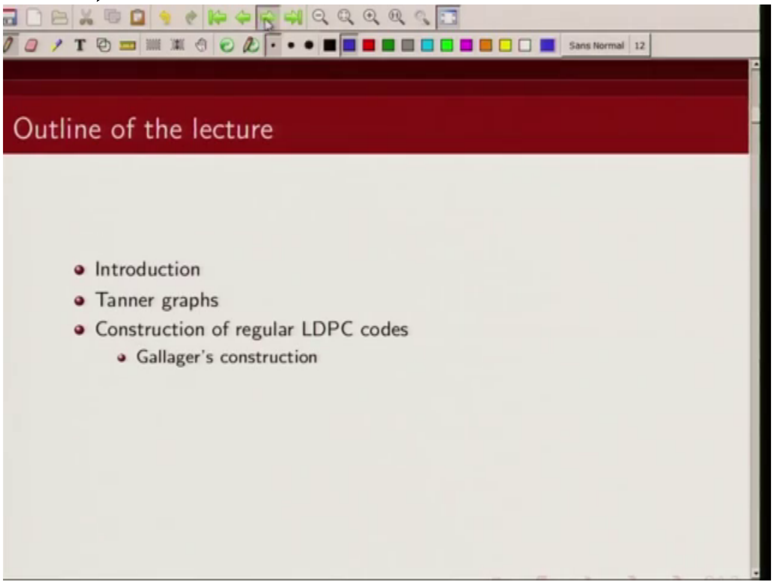
which is known as Tanner graph.

(Refer Slide Time 00:42)

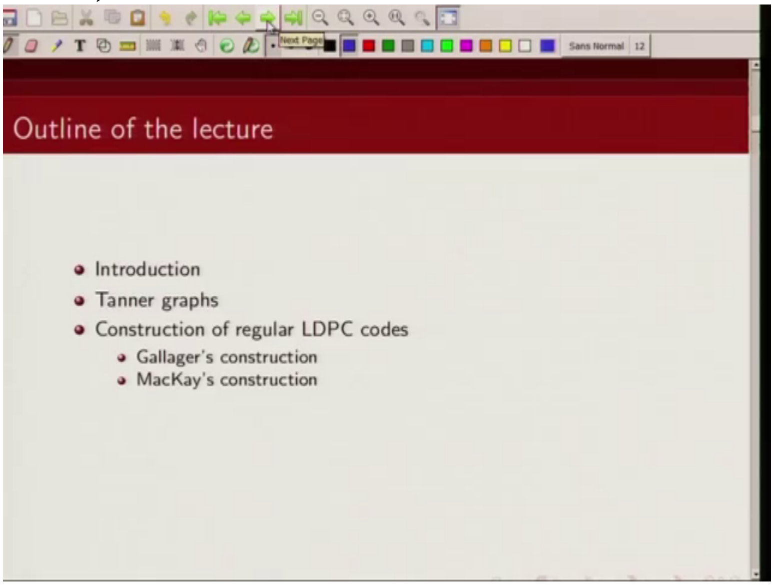


Then we will talk about what is a regular L D P C code and we will give some few

(Refer Slide Time 00:50)

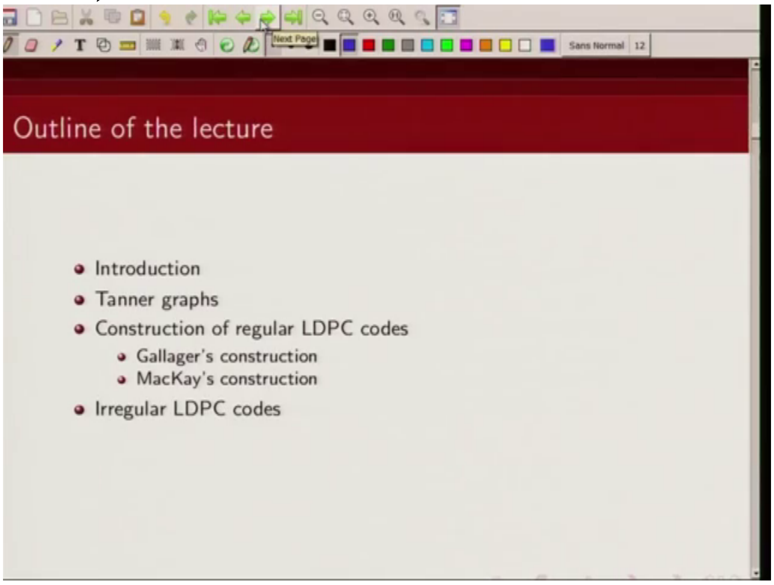


(Refer Slide Time 00:50)



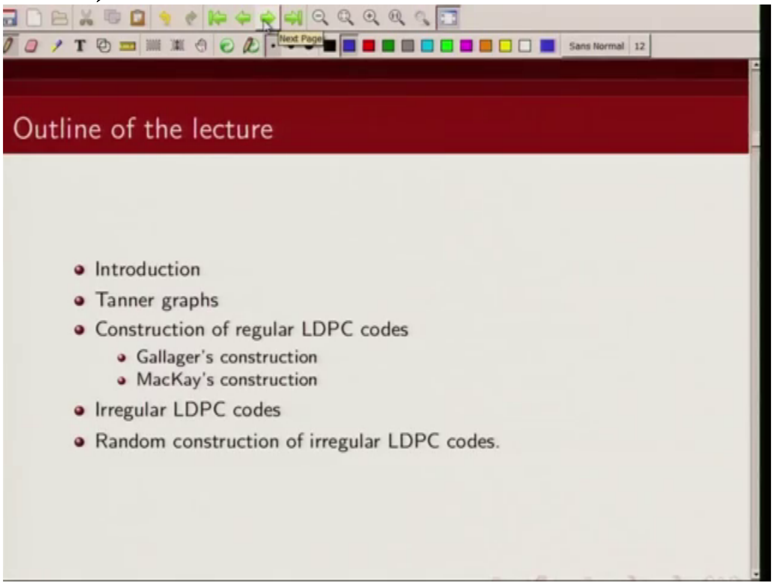
simple constructions of regular L D P C code. Then we will

(Refer Slide Time 00:55)



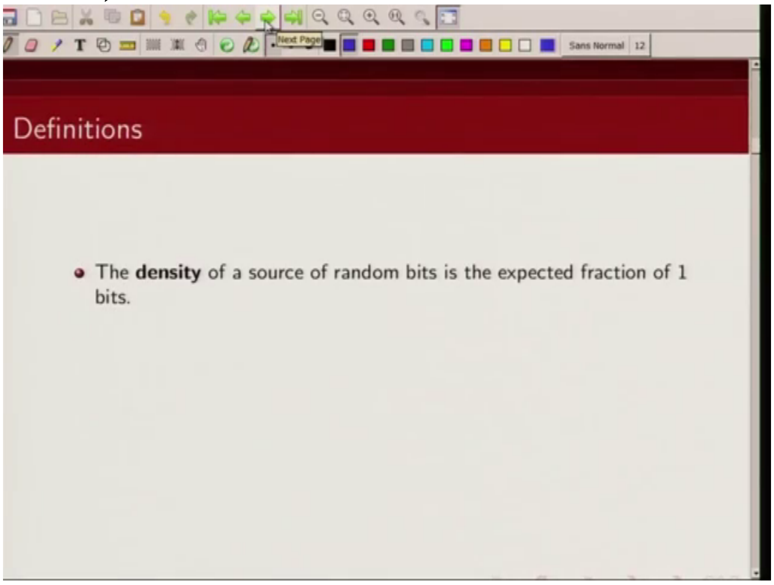
talk about irregular L D P C code

(Refer Slide Time 00:57)



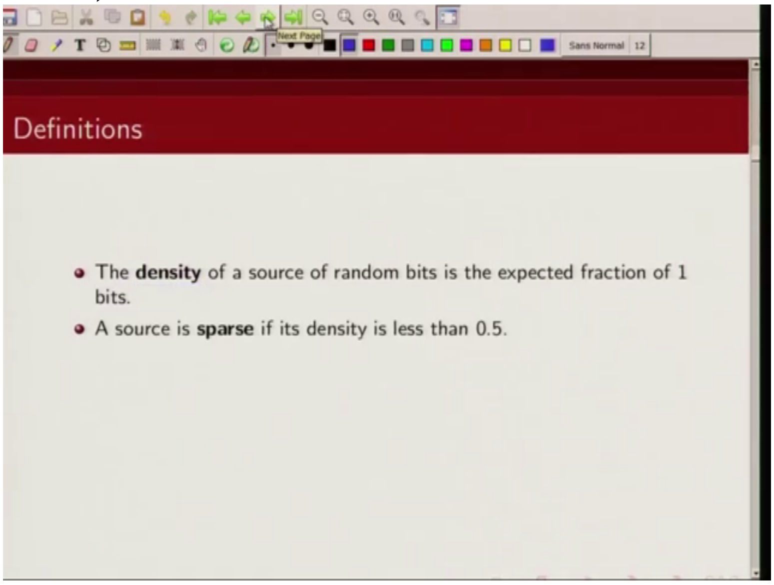
and then again we will give some very simple construction of irregular L D P C codes.

(Refer Slide Time 01:04)



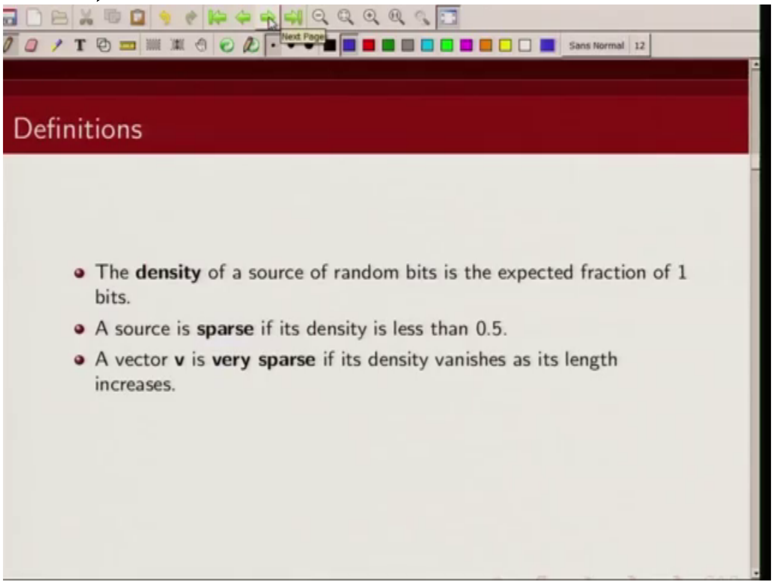
So what do we mean by low density? So we will first define what do we mean by density. So a density of a source is basically the expected number of 1s in the source. Now when is it a low density?

(Refer Slide Time 01:20)



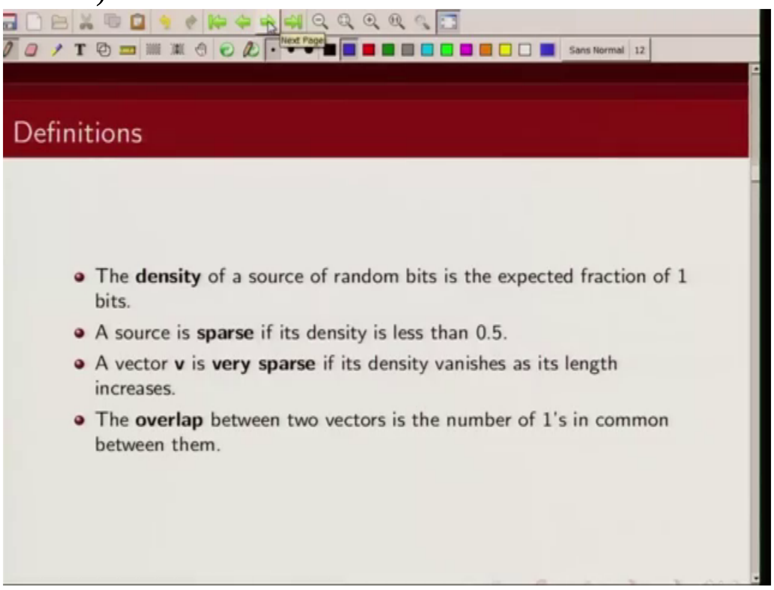
Now a source is low density or sparse if the density of 1 is less than point 5.

(Refer Slide Time 01:31)



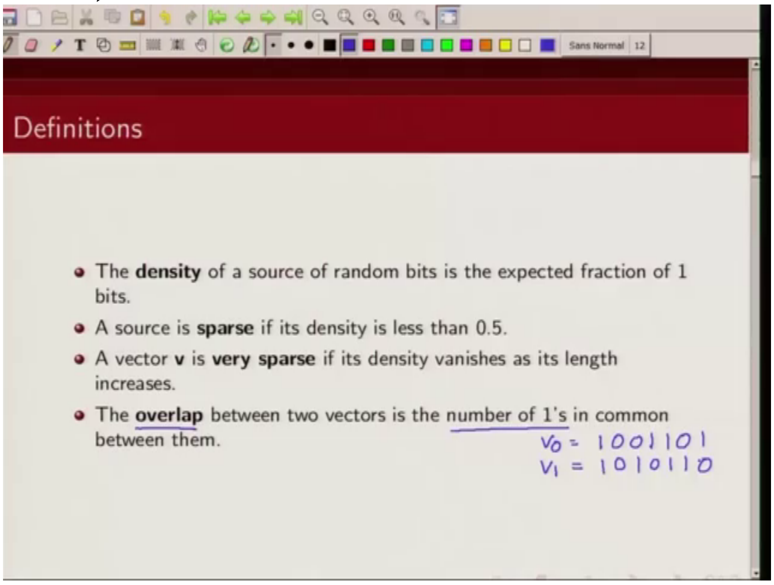
And we say the vector is very low density or it is low density if the density vanishes as the length of the vector increases. In other words number of 1s are fixed even if we increase the length of the vector. In that case the density will vanish as length increases.

(Refer Slide Time 01:55)



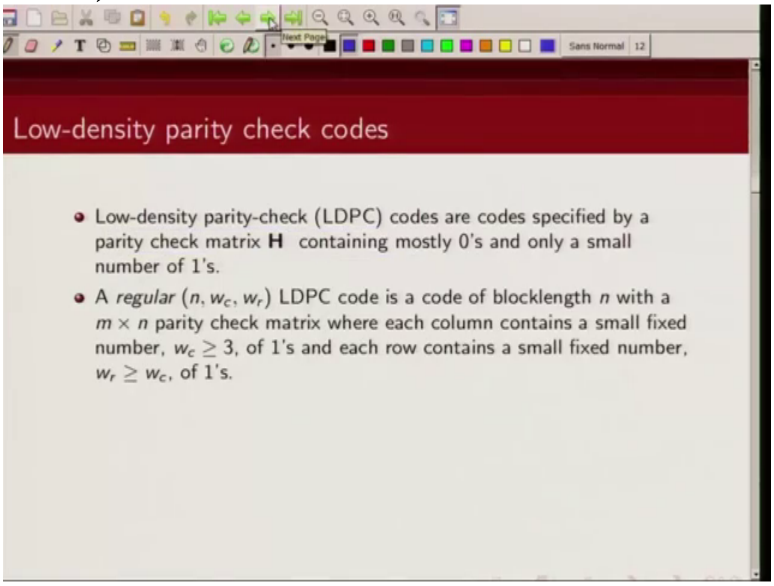
We will also define a term which is called an overlap. So if you have two n tuples, we call an overlap between 2 vectors as the number of positions in which the 1s are common. So for example, if you have a vector, let us call it v 0 which is 1 0 0 1 1 0 1 and you have a vector v 1 which is 1 0 1 0 1 1 0, then we can see

(Refer Slide Time 02:30)



there is an overlap here in 1 location, 2 location, so there is an overlap of 2. So what is a low density parity check code? As the name suggests, a low density parity check code are specified by a parity check matrix which is of low density. And what do we mean by low density? So the number of 1s in this parity check is very small, is less than a half. So an L D P C codes are specified by a parity check matrix which consist of mostly 0's and very few 1s.

(Refer Slide Time 03:17)

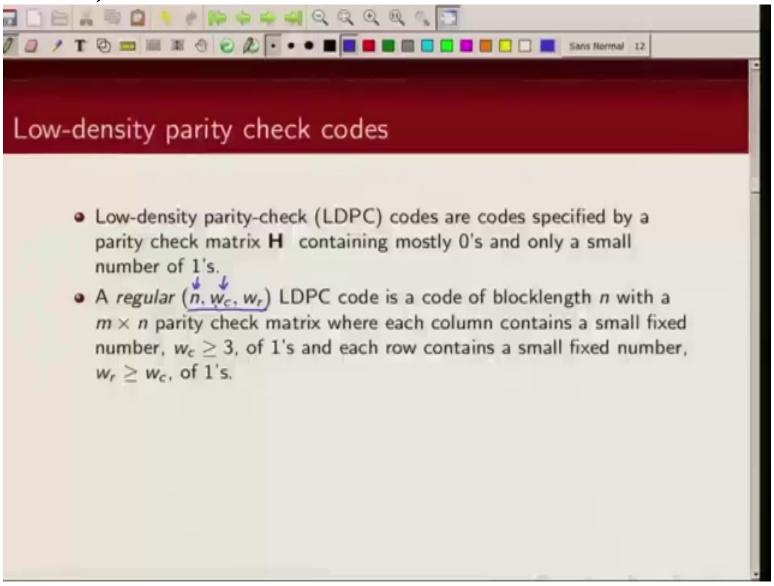


Now what is a regular L D P C code? A regular L D P C code is defined by these 3 parameters. This is the code length, this is number of 1s in the columns of parity check matrix so regular L D P C code has same number of 1s

(Refer Slide Time 03:39)

in each of the columns in parity check matrix and that number

(Refer Slide Time 03:44)

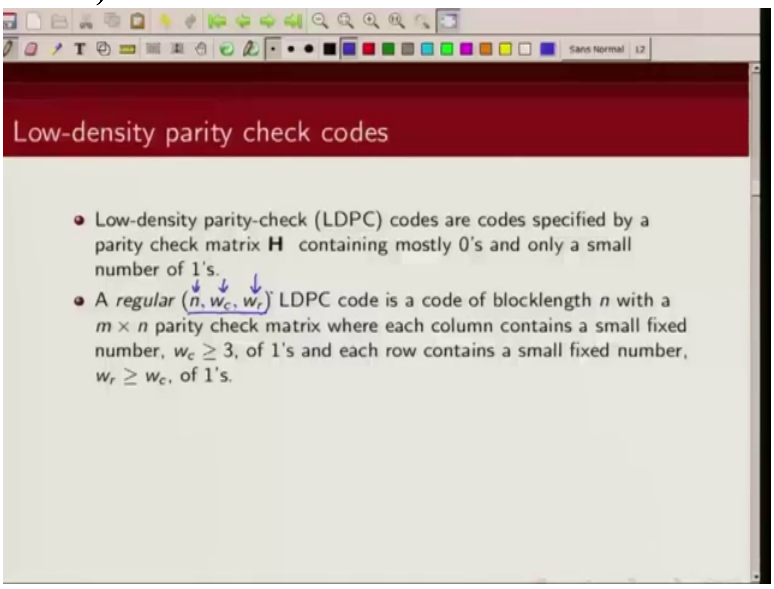


is given by w subscript c. Similarly w subscript r gives us the number of 1s in

(Refer Slide Time 03:53)

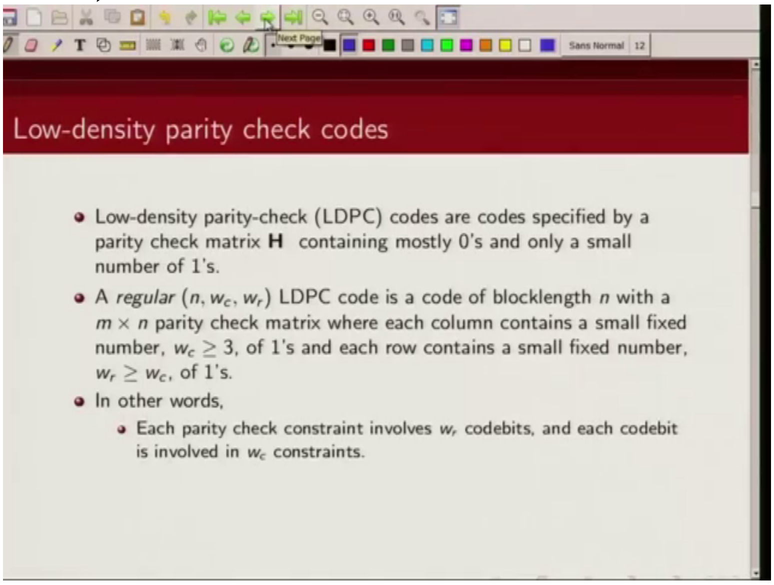
each of the row in parity check matrix. Again for a regular L D P C code,

(Refer Slide Time 04:01)



the number of 1s in each row is same. So a regular L D P C code is specified by this block length m and number of 1s in each of the columns and number of 1s in each of the rows. So we can describe it by a low density parity check matrix of m cross n where each column has a fixed number of 1s and that is w 3 c and that has to be greater than 3. This has to do with distance properties of L D P C codes and each row has w r number of 1s. w r is greater than equal to w c. In other words, now

(Refer Slide Time 04:47)

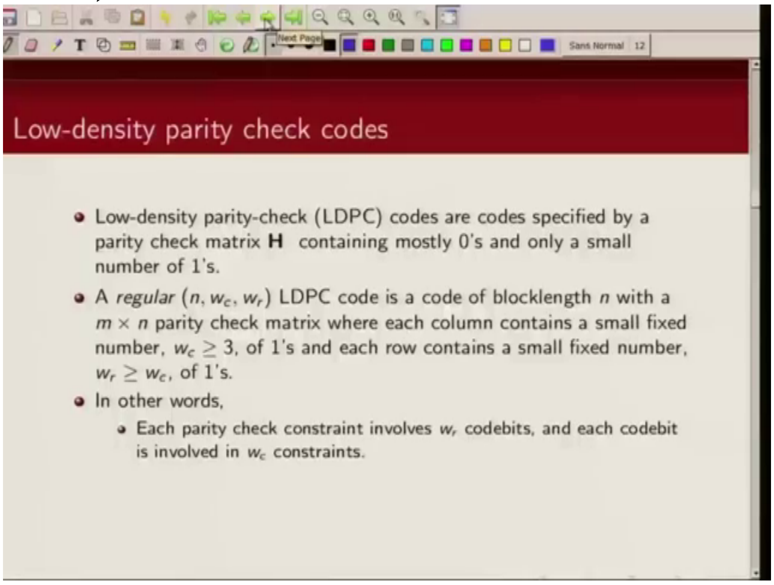


what do the rows in the parity check matrix specify? Now if there

(Refer Slide Time 04:54)

are w r 1s in rows of the parity check matrix, it specifies that w r bits are participating in a parity check equation and in all the parity check equations the same number of bits are participating. And what

(Refer Slide Time 05:13)

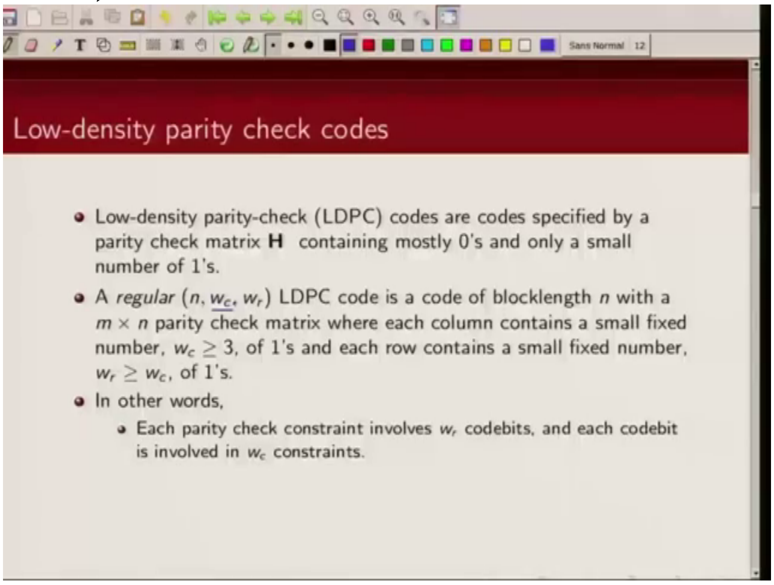


is the implications of w c 1s in each column? It means each bit appears in w sub c parity

(Refer Slide Time 05:24)

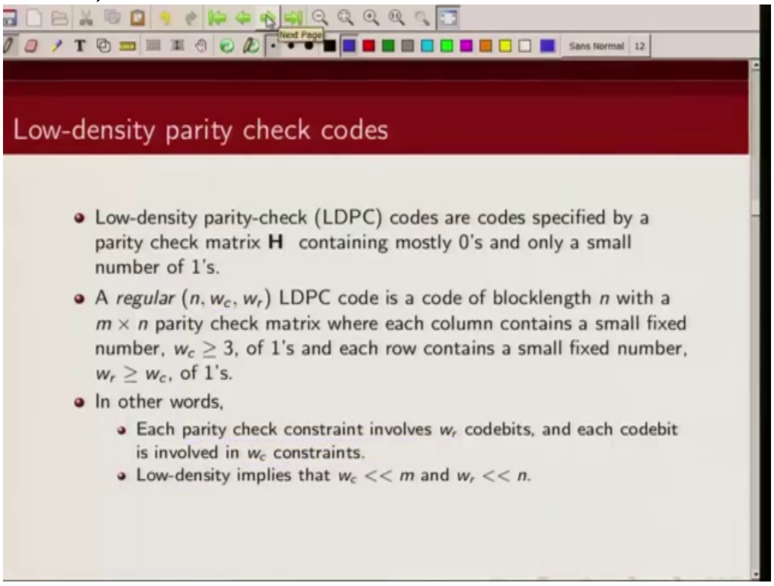
check equations. So each bit participates in w c parity check equations. So that's what I am saying

(Refer Slide Time 05:32)



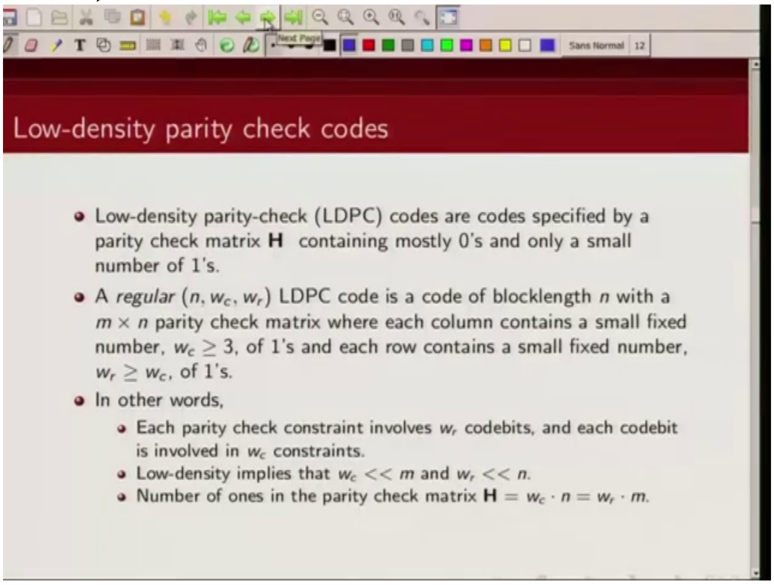
here. So each parity check constraint in an regular L D P C code will have w r code bits and each code bit appears in w c parity check constraints.

(Refer Slide Time 05:51)



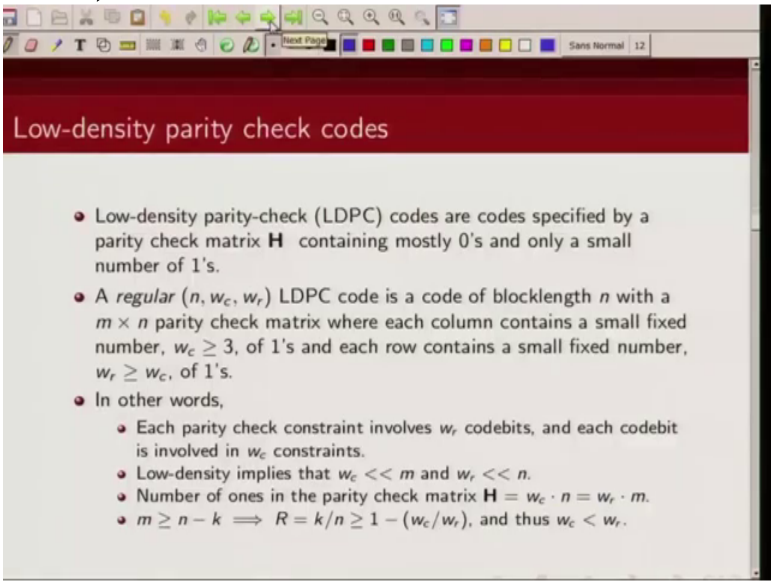
Now typically the number of 1s because it is a low density parity check matrix so numbers of 1s are much less than the dimension of these matrix and w r is also much less than n.

(Refer Slide Time 06:07)



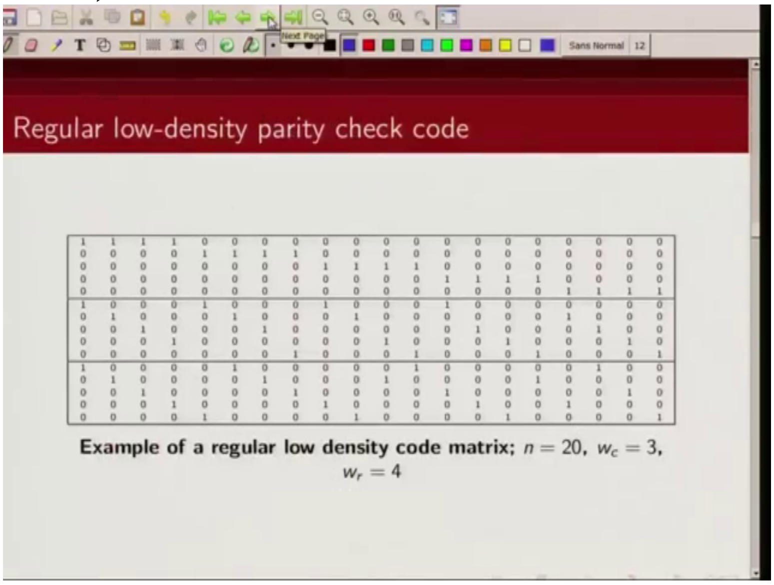
Number of 1s we can count it column wise, there are n columns and there are w c 1s in each column, that number should be equal to number of rows multiplied by number of 1s in each row.

(Refer Slide Time 06:24)



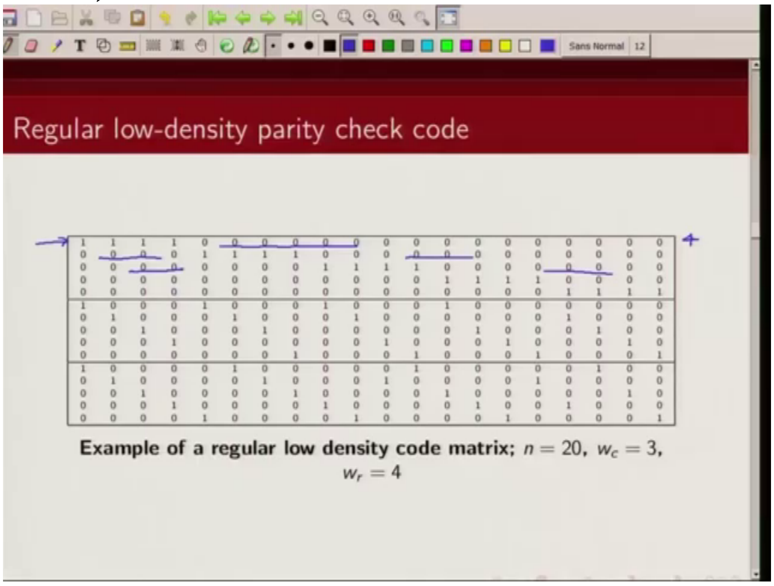
And number of parity check equations is at least equal to n minus k. So the rate is at least 1 minus w c by w r. Sometimes we do have some redundant parity check equations in the L D P C parity check matrix.

(Refer Slide Time 06:48)



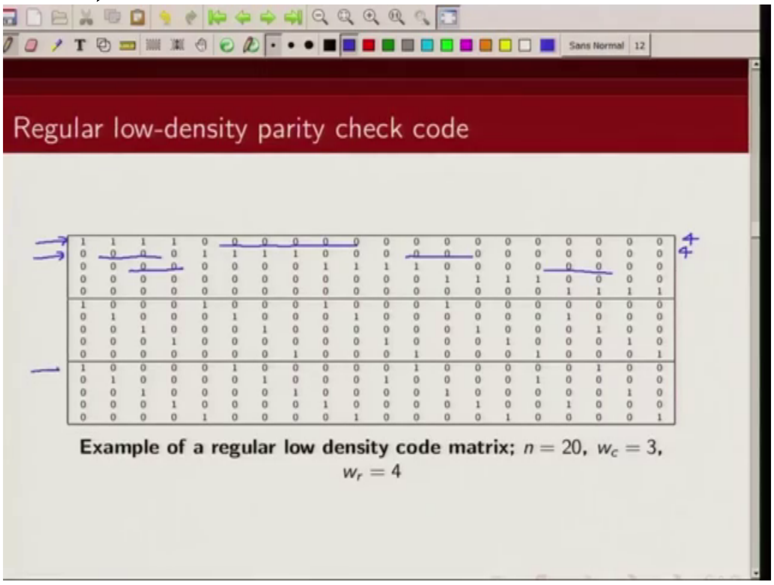
This is one example of the low density parity check code. You can see in this matrix most of the entries are zeroes, these are all zeroes, these are zeroes, you can see most of the entries in this matrix are 0s very few are 1's.And you can see that each row, let's look at row number 1; row number 1 has 4 1s.

(Refer Slide Time 07:17)



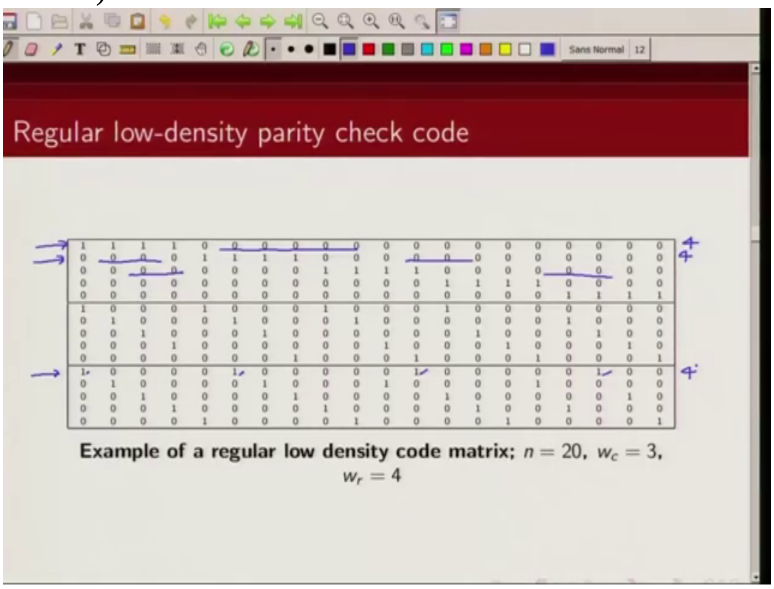
Row number 2 has 4 1's. You can check any row. You can check, let's say this row.

(Refer Slide Time 07:23)



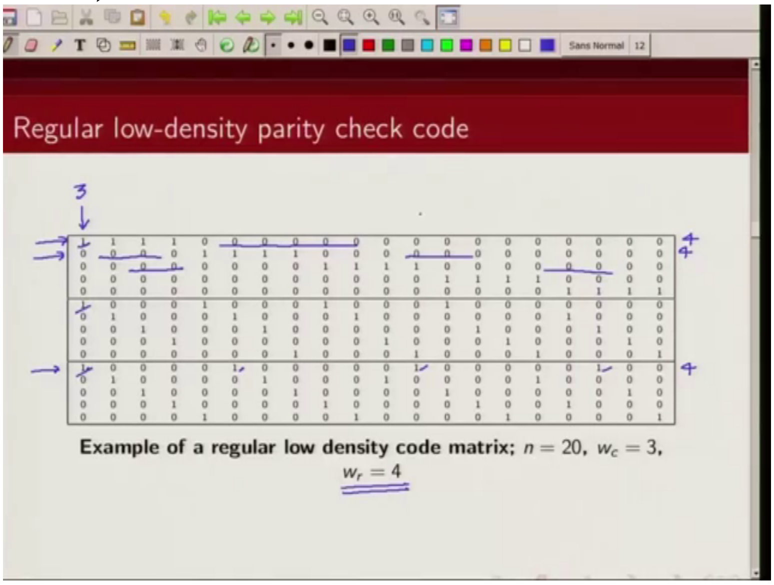
This has 1, 2, 3, 4. There are 4. So each row of

(Refer Slide Time 07:31)



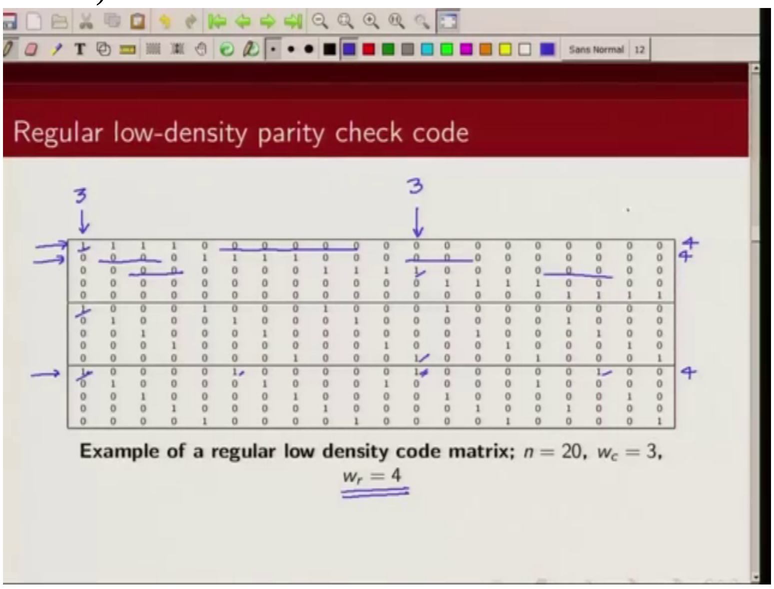
this low density parity check matrix has 4 number of 1s. So w r in this case is 4. And each column, let's take column 1, there is a 1 here, there is a 1 here and there is a 1 here. So column weight is 3. You can check any

(Refer Slide Time 07:53)



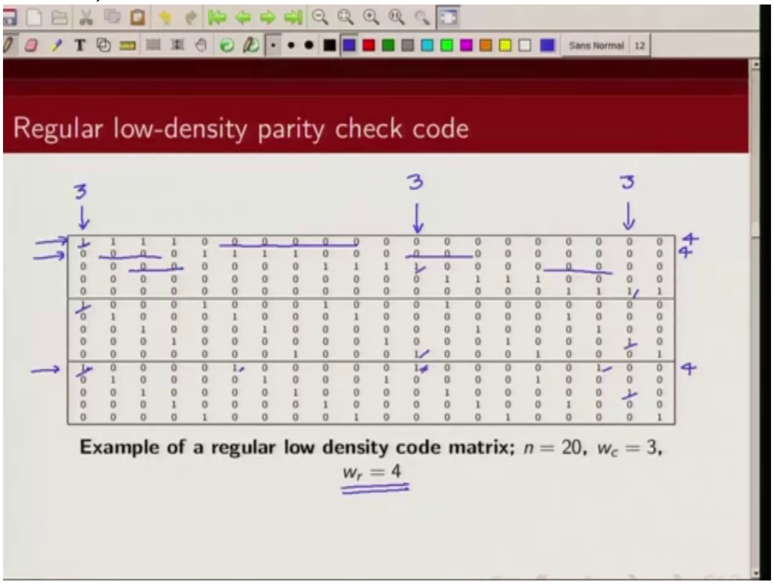
column. Look at this column. 1 here, 1 here and 1 here, column weight is 3. Take this column.

(Refer Slide Time 08:02)



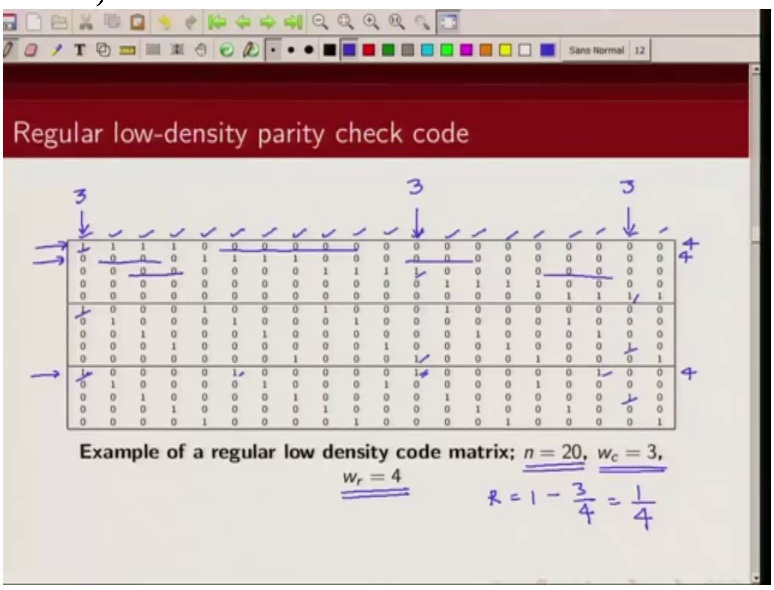
There is a 1 here, there is a 1 here and there is a 1 here. So the column weight is

(Refer Slide Time 08:10)



3. So w c is 3, n is 20 you can see, there is 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20. So the block size is 20. And this is full rank. So the rate here is 1 minus 3 by 4 which is a rate 1 4

(Refer Slide Time 08:39)

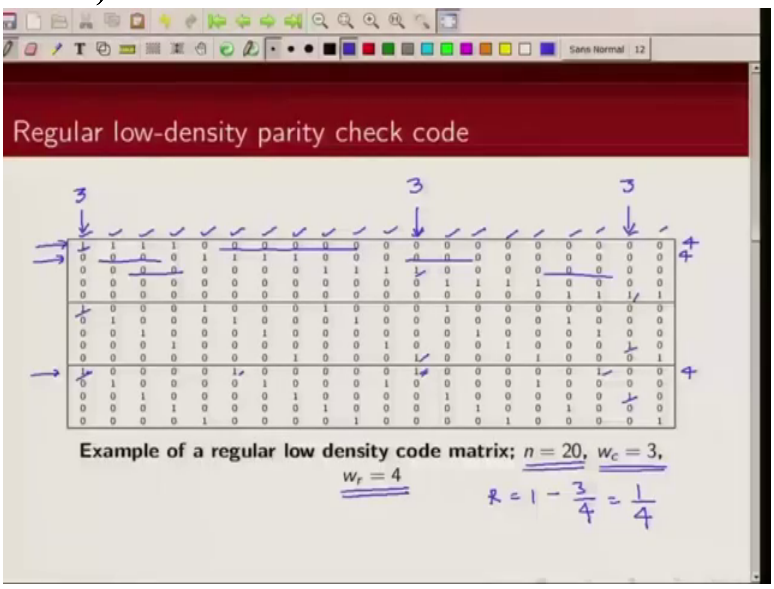


code. So this is an example of low density parity check

(Refer Slide Time 08:45)

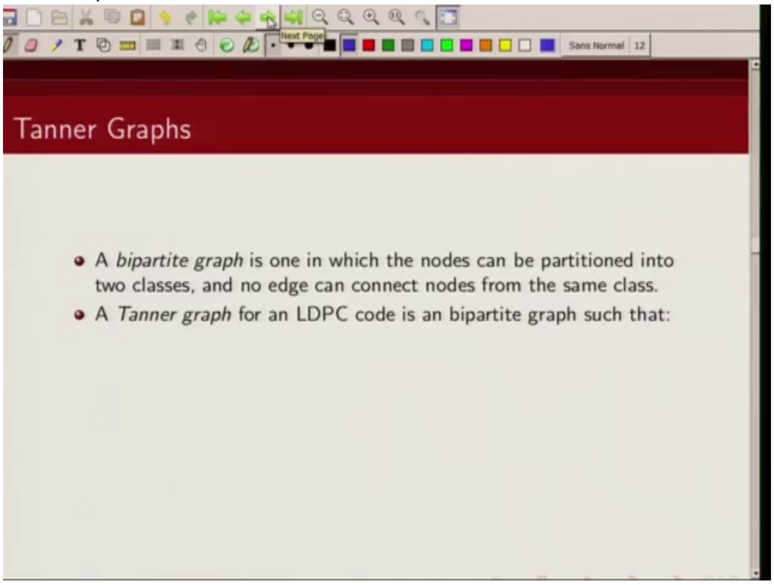
code. You can see the fraction of 1s is much smaller than

(Refer Slide Time 08:50)



number of 0's. Now

(Refer Slide Time 08:55)

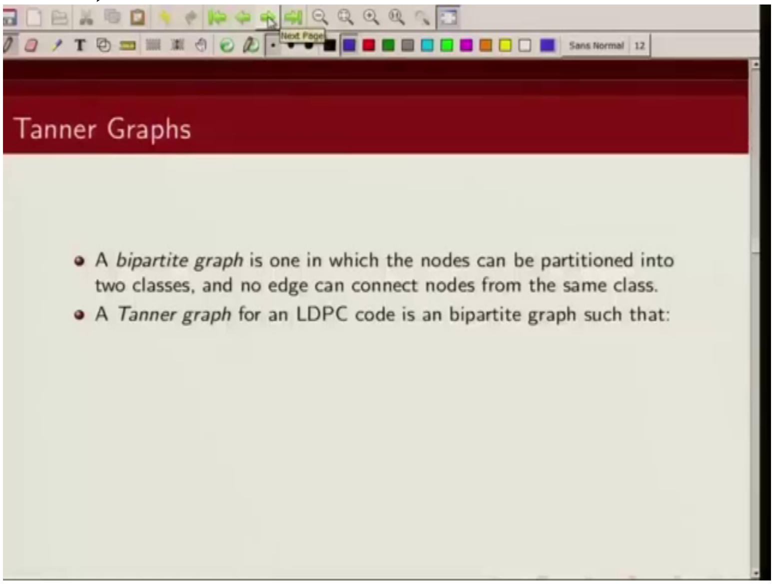


we represent this

(Refer Slide Time 08:58)

parity check matrix using a bipartite graph and this bipartite graph representation of parity check matrix of a linear block code is known as Tanner graph named after

(Refer Slide Time 09:12)

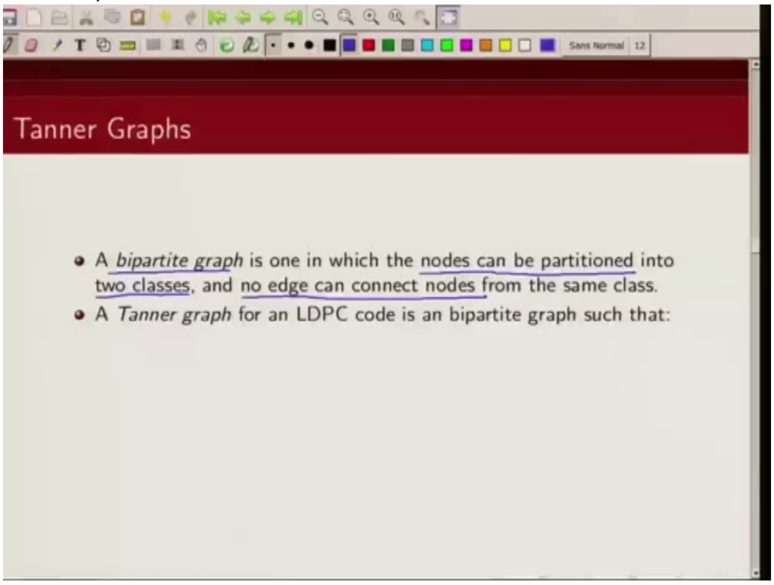


Michael Tanner. So what is a bipartite graph? In a bipartite graph the nodes can be partitioned into 2 classes. Now what are those 2 classes? What is the property? That no edge can connect nodes from the same class. So when we partition the nodes of this graph in 2

(Refer Slide Time 09:37)

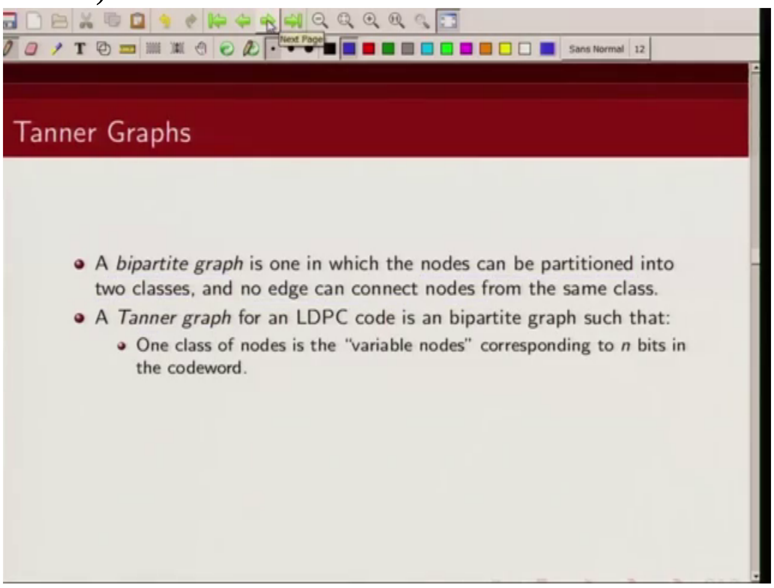
classes there is 2 connection between nodes within a class. So if you want to reach another node within a class you at least have to have traversed twice, Ok. So a bipartite graph is one where I can separate out the nodes into 2 classes such that

(Refer Slide Time 09:58)



there is no edge connecting nodes in the same class. Now we can draw a bipartite graph for an L D P C code parity check matrix. So a Tanner graph and that's basically known as Tanner graph; so Tanner graph on an L D P C code is a bi partite graph

(Refer Slide Time 10:23)

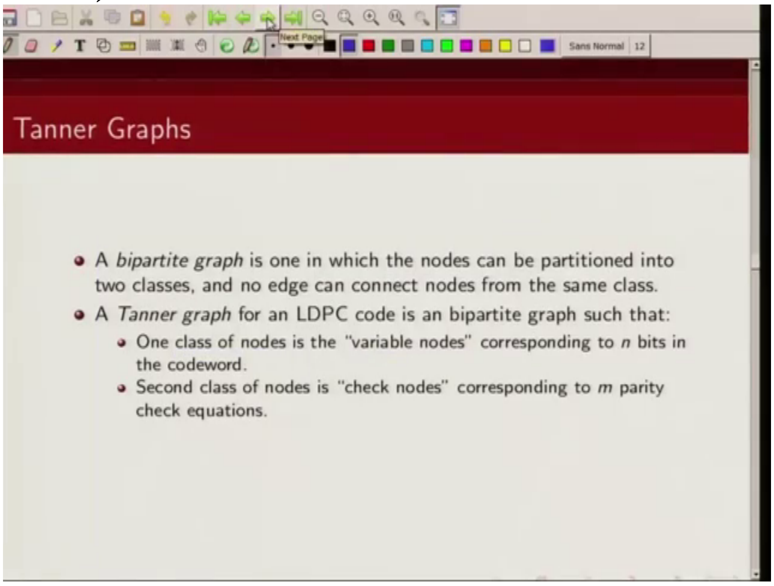


which has the property that there are 2 sets of class of nodes. One class of nodes which we call variable nodes, they represent the n bits of the

(Refer Slide Time 10:36)

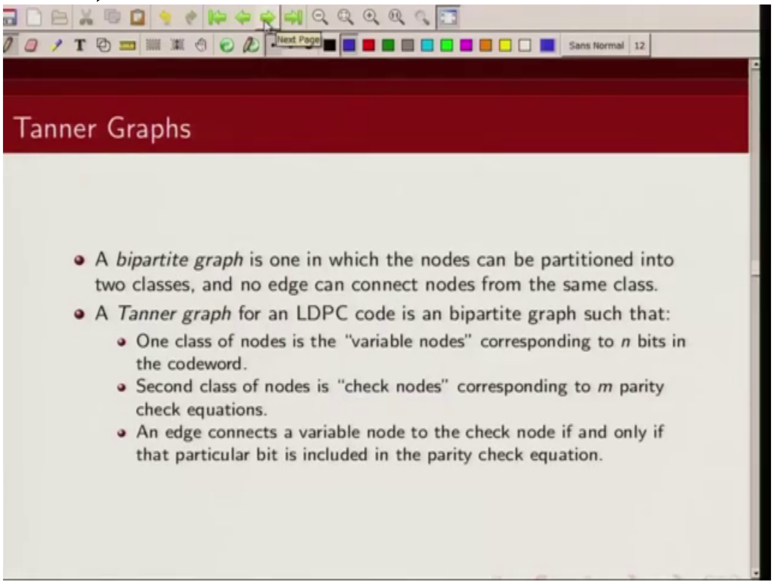
codeword. And the other class

(Refer Slide Time 10:38)



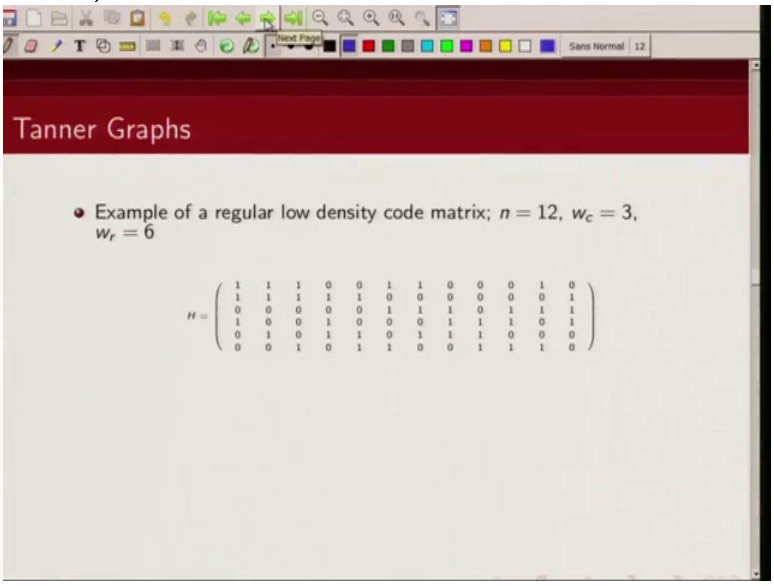
is what is known as check nodes, they represent these m parity check equations.

(Refer Slide Time 10:49)



And how do we connect an edge? An edge connects a variable node to the check node if and only if that particular bit participates in the parity check equation. So let us

(Refer Slide Time 11:07)

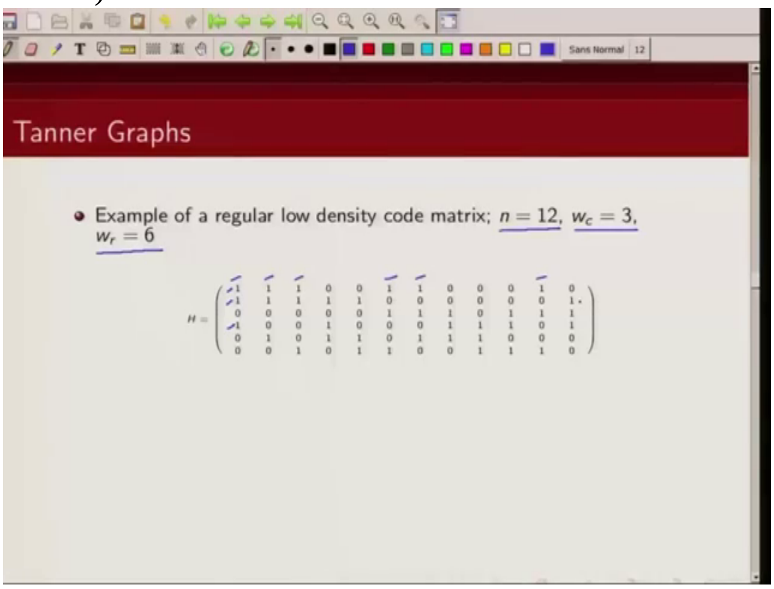


take an example to illustrate how we can draw a Tanner graph of an L D P C code. So this is an L D P C code, block n is 12, number of 1's in each column you can see 1, 2, 3 that w c is 3. And number of 1's in each row is 6. You can check.1, 2, 3, 4, 5, 6. Each rows has six 1s. Each column has three 1s. Now how do we draw the Tanner graph of this? So as I have said there are 2 class of nodes.

(Refer Slide Time 11:49)

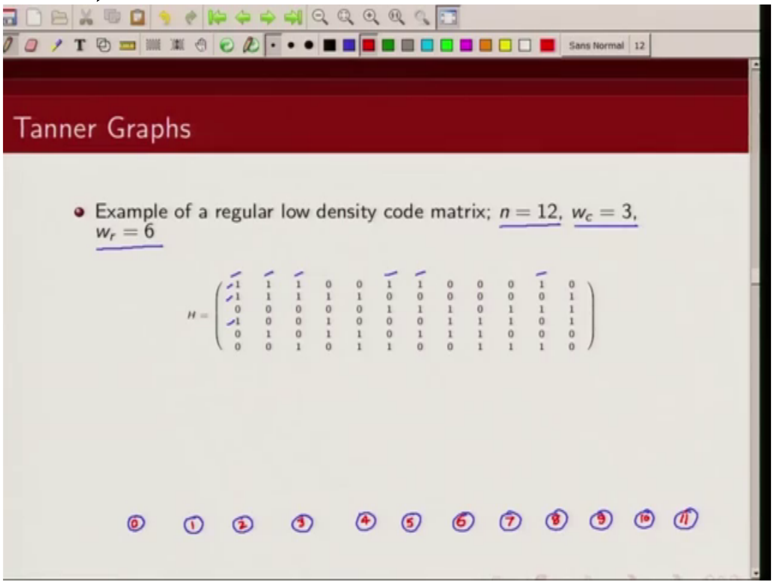
One class of nodes for the variable nodes.

(Refer Slide Time 11:52)



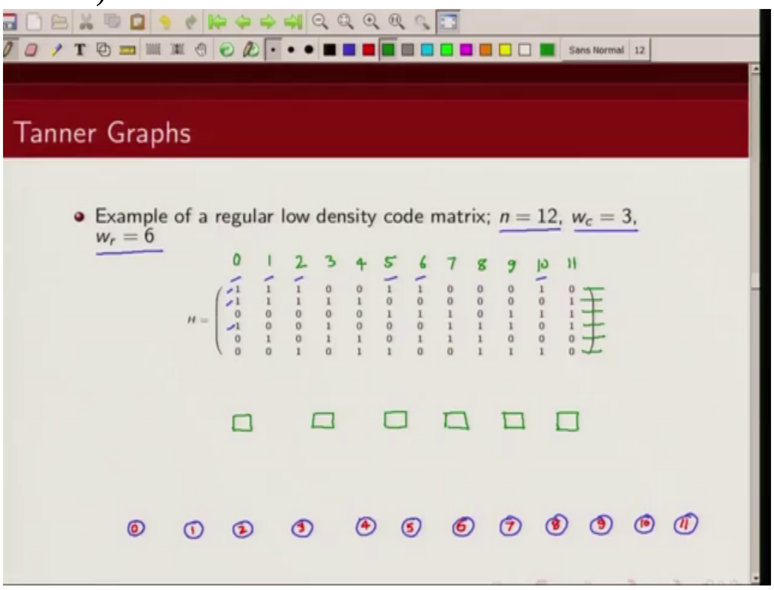
And how many variable nodes we have? We have 12 variable nodes. So let us just draw 12 variable nodes; 1, this 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. Let's just label them. Let's just label them as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11.

(Refer Slide Time 12:35)



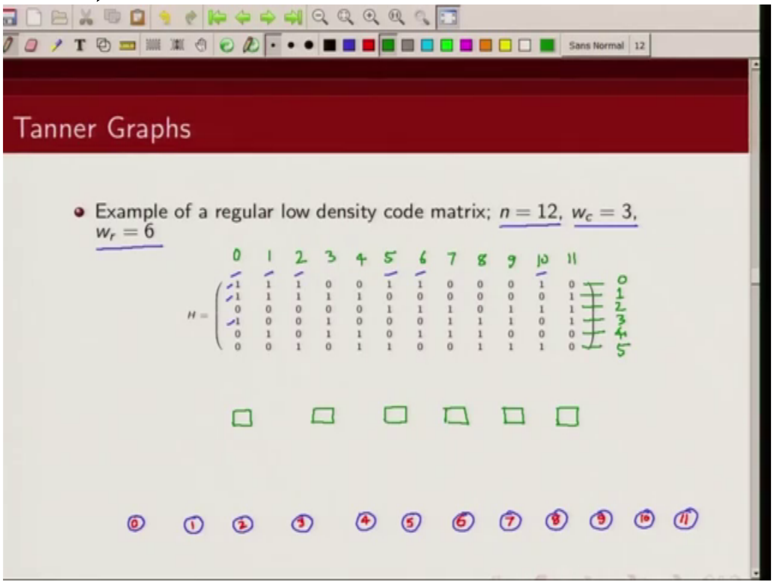
And then you have how many parity check equations? 1, 2, 3, 4, 5, 6. So we will have the next set of nodes, will be for parity check equations. And they are 6 of them, 4, 5, 6 Ok. Let's just label these bits. That will be easier for, this is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, Ok.

(Refer Slide Time 13:13)



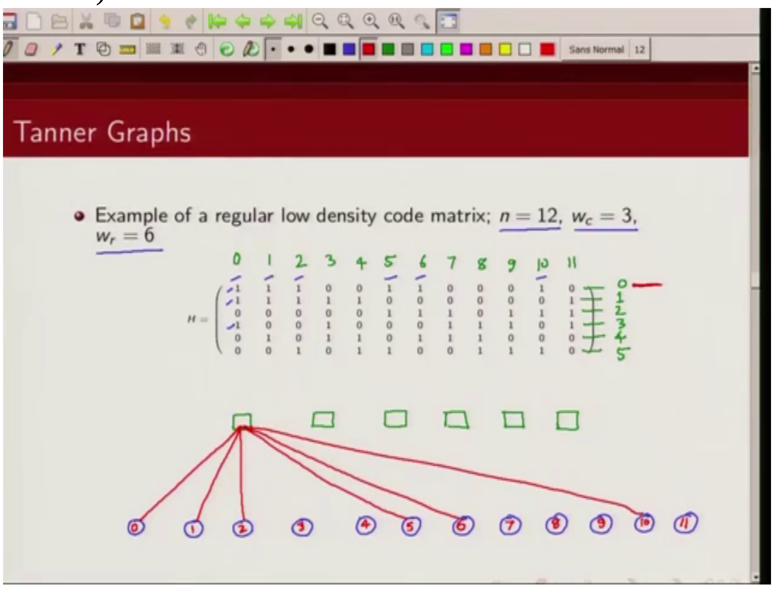
Now let's and similarly label these parity check equations. This is let's say 0 th parity check equation, 1, 2, 3, 4, 5.

(Refer Slide Time 13:22)



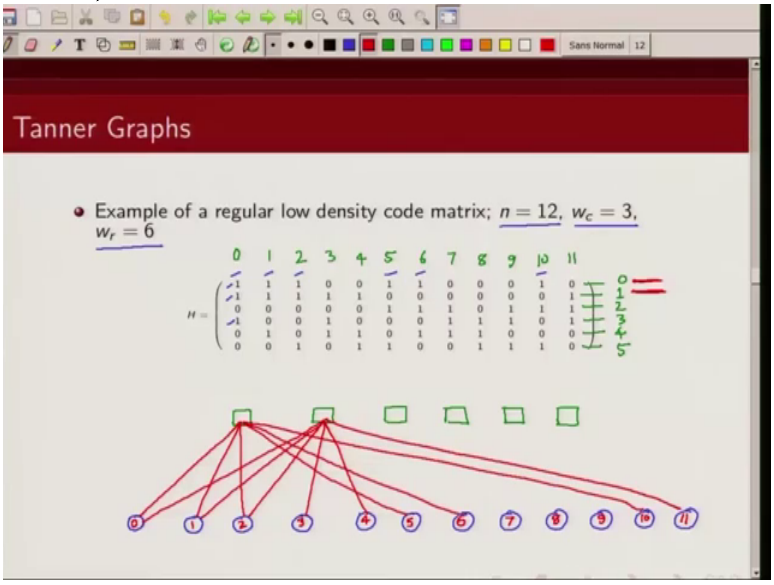
So let's look at this one, this 0th parity check equation. Now which are the bits that are participating in the parity check constraint? Bit number 0, so we will draw an edge from bit number 0 to this parity check constraint, bit number 1 that's this, bit number 2 that's this, bit number 5 that's this, bit number 6, that is this. Bit number 10, Ok. So this is my

(Refer Slide Time 14:05)



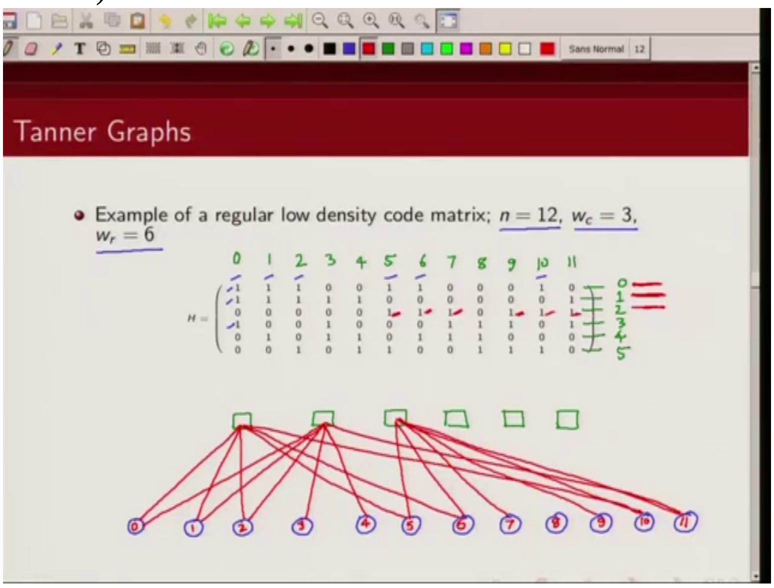
first parity check constraint. Let us look at now this one. Which are the bits participating? Bit number 0, bit number 0, bit number 1, bit number 1, bit number 2, so there is an edge from bit number 2 to this parity check constraint; bit number 3, that's this, bit number 4 that is this, and then you have bit number 11. So you have this one,

(Refer Slide Time 14:38)



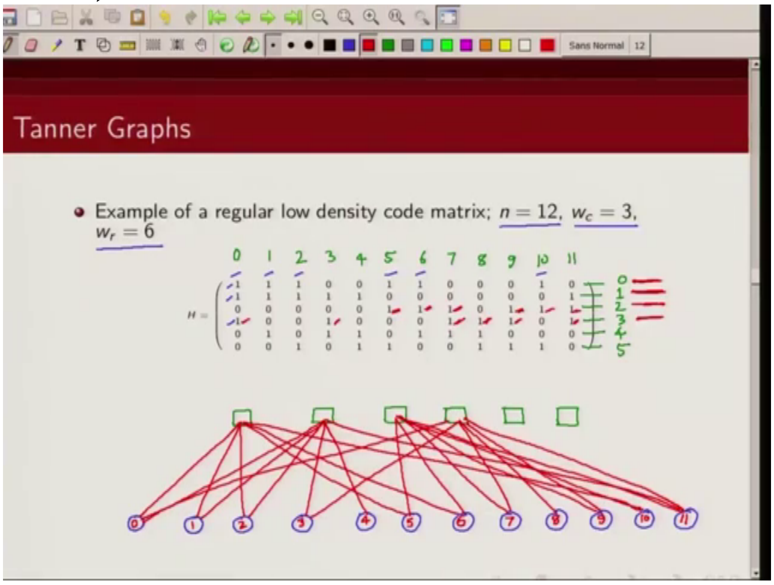
Ok. Now look at this parity check constraint. Now which are the bits that are participating? This is 5. So that's here. 6, that's this one. 7, that's this one. 9 that is this one. Then 10, that is this one and then 11, that is this one,

(Refer Slide Time 15:10)



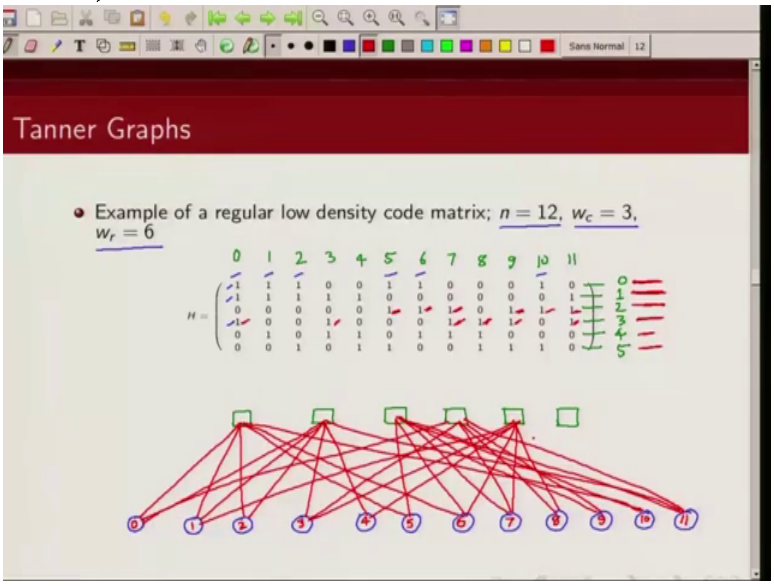
Ok. Similarly for parity check constraint, bit number 0 is participating so you have edge from here to here; bit number 3 is participating so there's an edge from here to here. Bit number 7 is participating so there is a edge from here to here, bit number 8 is participating so there is an edge from here, bit number 9 is participating so there is an edge from here to here and bit number 11 is participating so there is edge from here to here.

(Refer Slide Time 15:46)



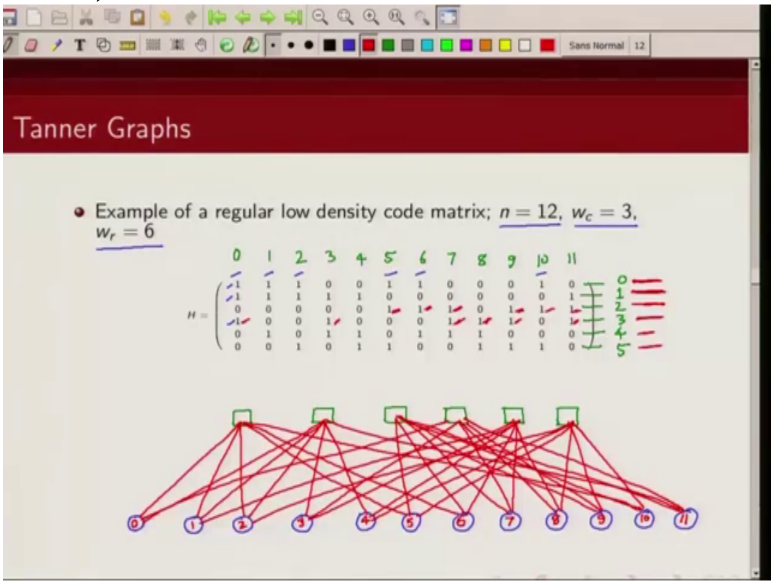
And similarly we can do for this, I will just do it. This is 1, 3, 4, 6, 7, 8, that's it. And this finally this parity check constraint, bit number 2,

(Refer Slide Time 16:14)



bit number 4, bit number 5, bit number 8, bit number 9, bit number 10.

(Refer Slide Time 16:29)

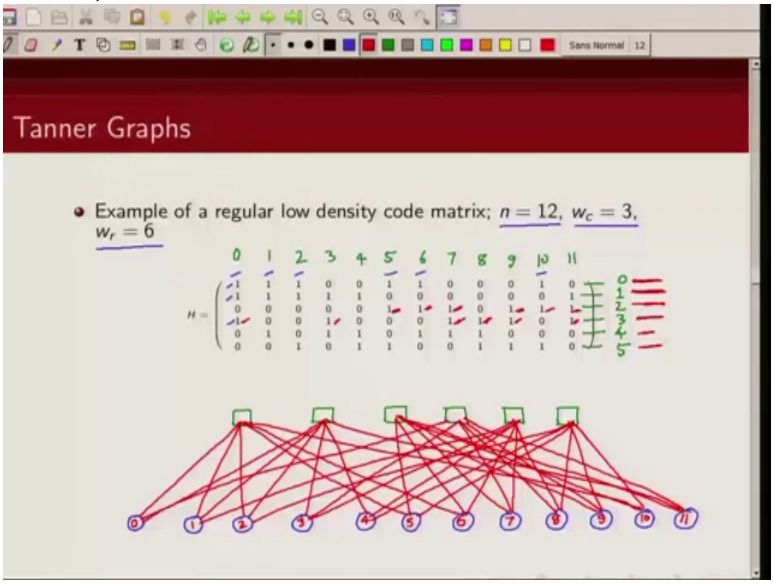


So this is the Tanner graph representation

(Refer Slide Time 16:34)

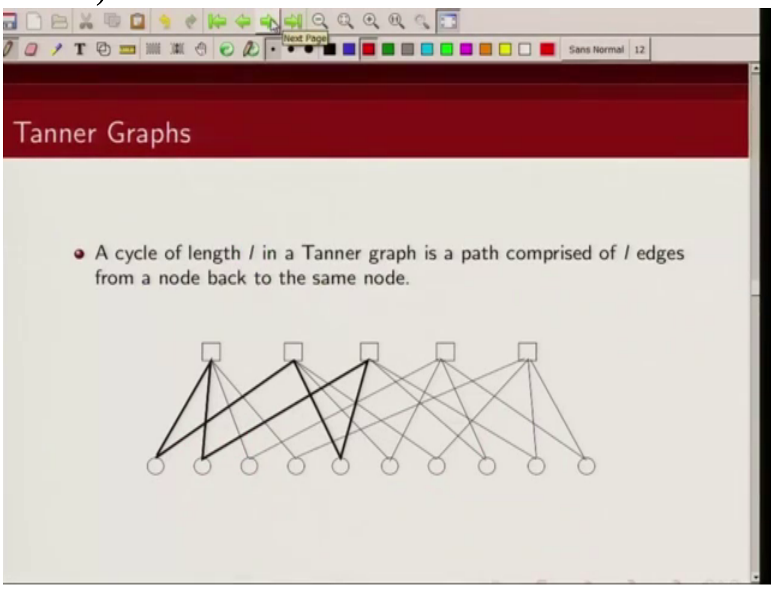
of this low density parity check code,

(Refer Slide Time 16:39)



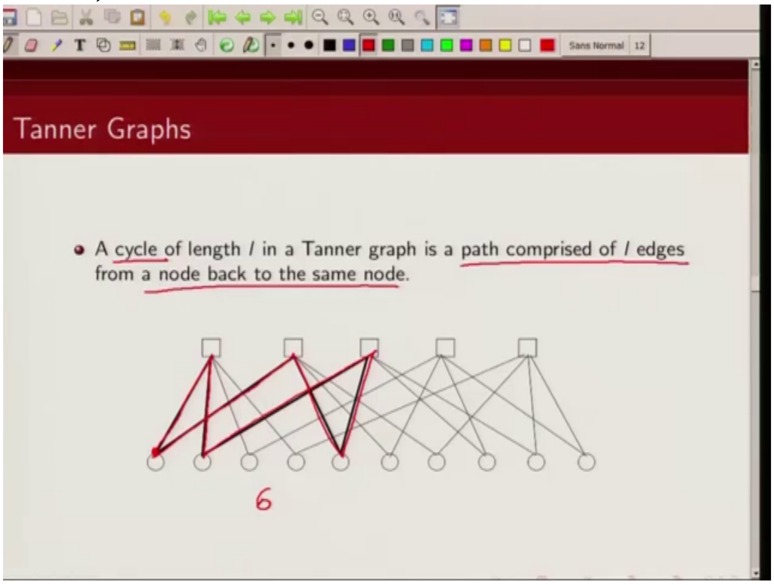
Ok. Now

(Refer Slide Time 16:42)



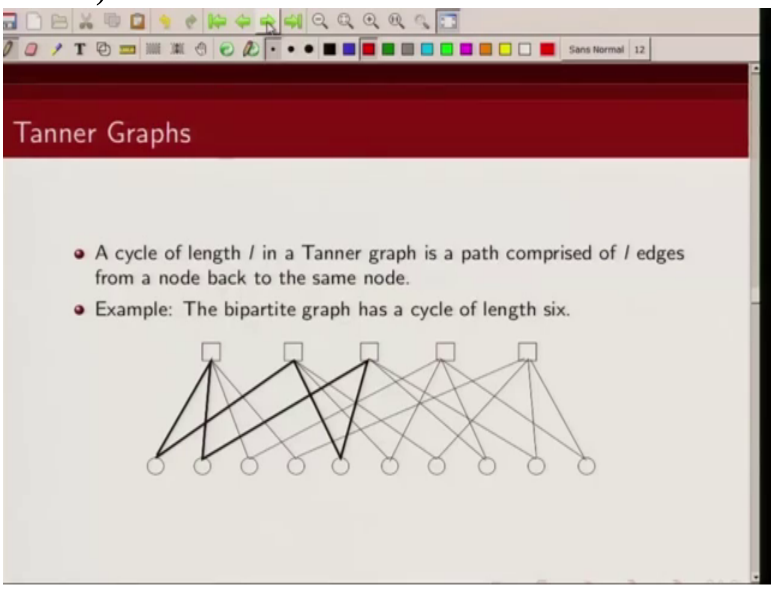
let us define what do we mean by cycle in this Tanner graph. So cycle is defined as a path consisting of length l which will start from a node and come back to the same node. So what is a cycle? So it is a path, cycle of length l is, starts from a particular node and comes back to same node. So let's look at this, let's say this node. So if you start from this node, this edge 1, 2, 3 4, 5, 6 so this is a cycle of length 6. Now note

(Refer Slide Time 17:24)



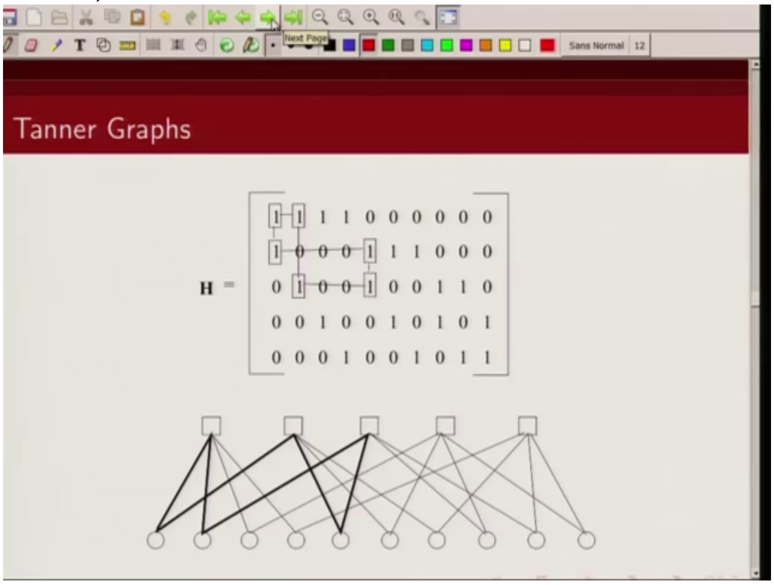
that it is a bipartite graph so it will only have even length cycles because there is no connection between nodes of the same class

(Refer Slide Time 17:36)



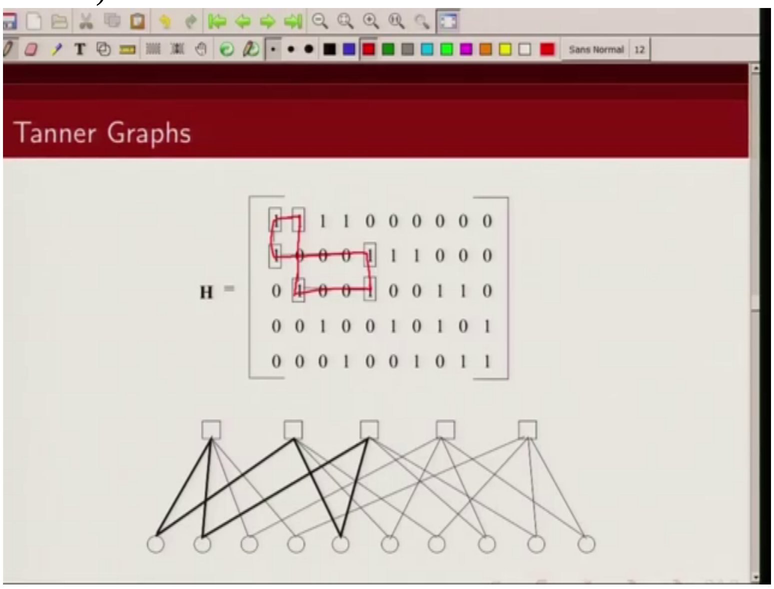
as I said in this particular example this has cycle 6, 1, 2, 3 4, 5, 6. And this can be viewed from

(Refer Slide Time 17:47)



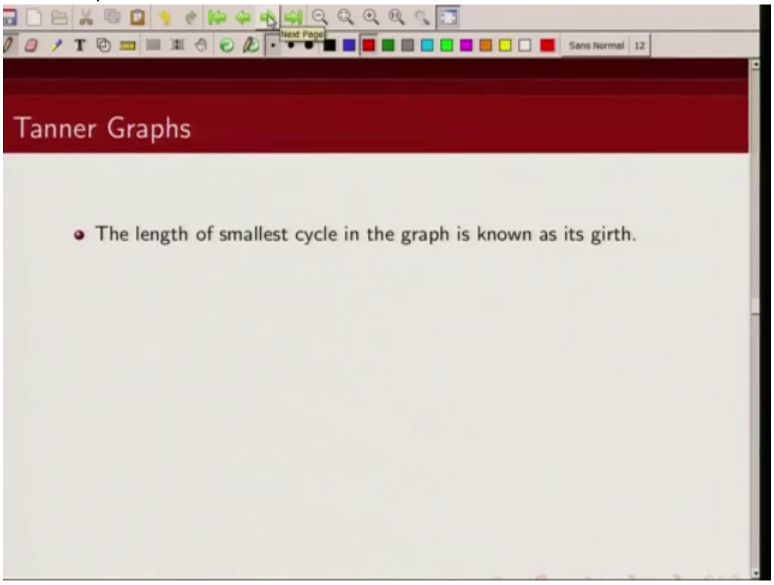
the parity check matrix also so 1, 2, 3 4, 5, 6

(Refer Slide Time 17:54)



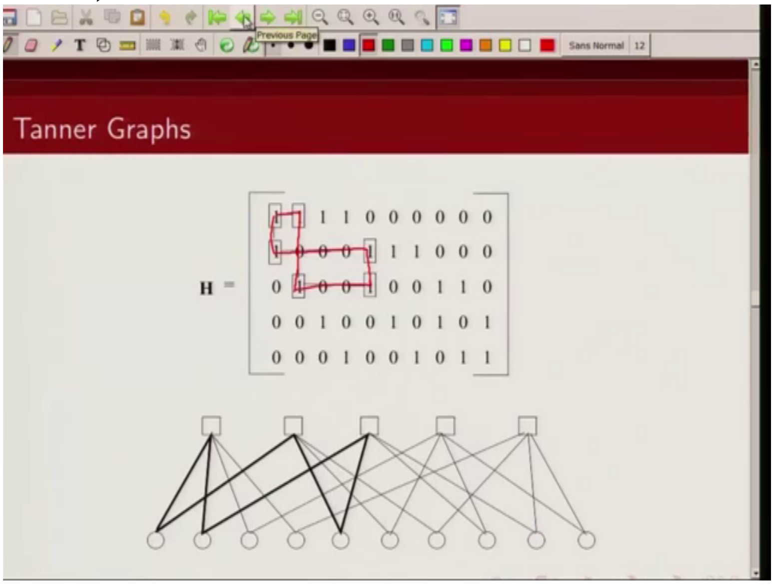
Ok.

(Refer Slide Time 17:59)



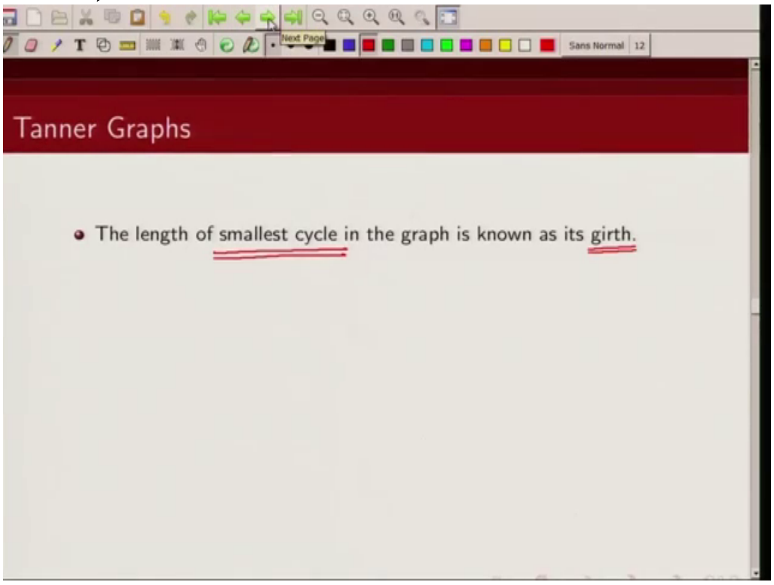
Now we will define what is known as girth. Girth is the length of the smallest cycle in this graph. So girth is defined as the length of the smallest cycle in this graph. In this particular example,

(Refer Slide Time 18:19)

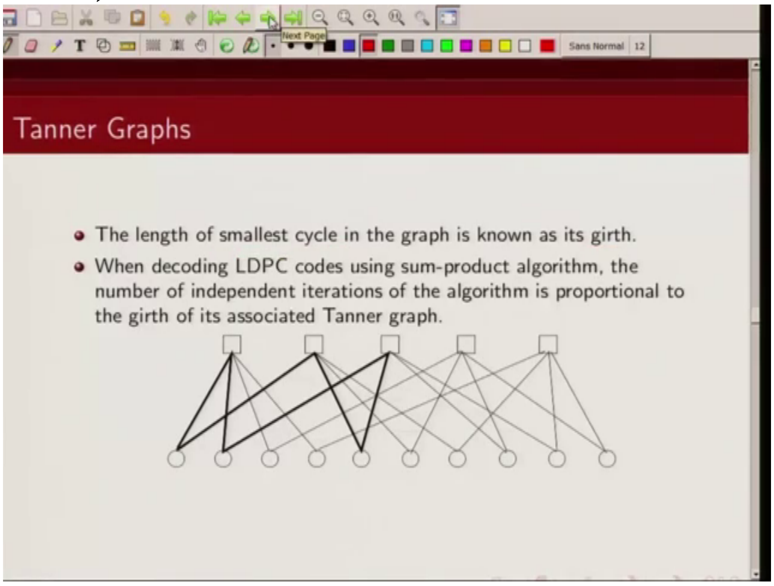


the smallest cycle is 6. You can see there is no cycle of length 2 or 4.

(Refer Slide Time 18:27)



(Refer Slide Time 18:30)

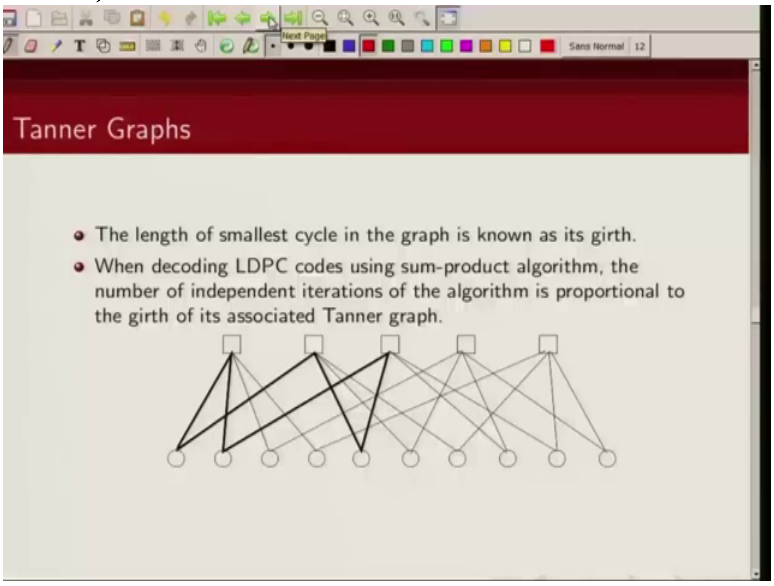


Now when we are decoding L D P C codes we would like

(Refer Slide Time 18:33)

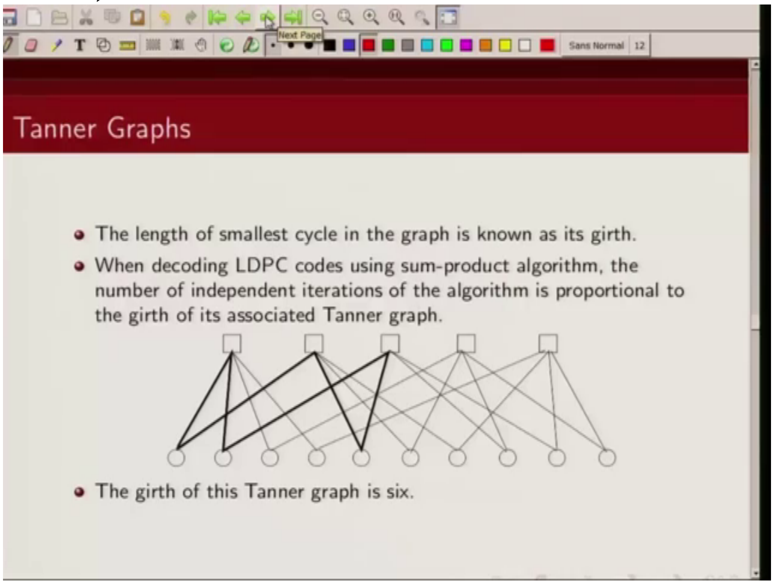
the girth to be very large because number of

(Refer Slide Time 18:38)



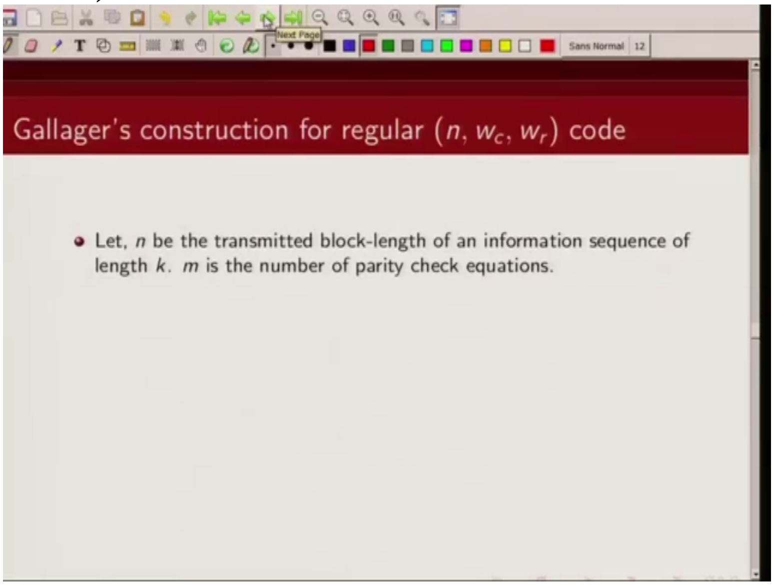
independent iterations that we can get is proportional to the girth of the corresponding Tanner graph

(Refer Slide Time 18:46)



of the L D P C code.

(Refer Slide Time 18:48)

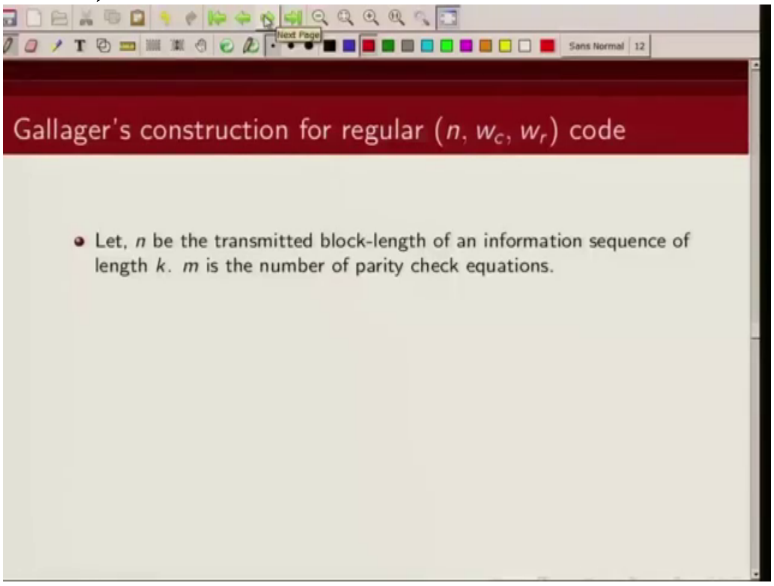


Now that we have defined what is a regular L D P C code, let's talk about how we can construct these

(Refer Slide Time 18:58)

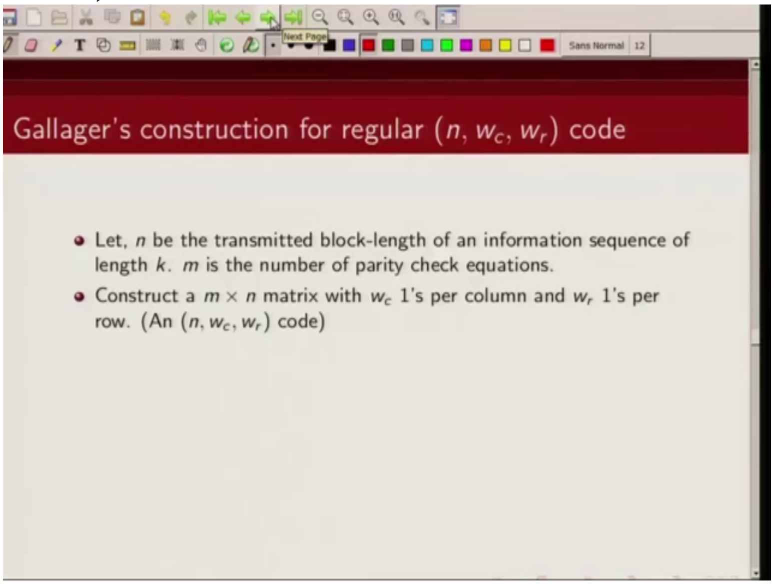
L D P C codes. So we will first start with random construction of L D P C codes given by Gallager.

(Refer Slide Time 19:05)



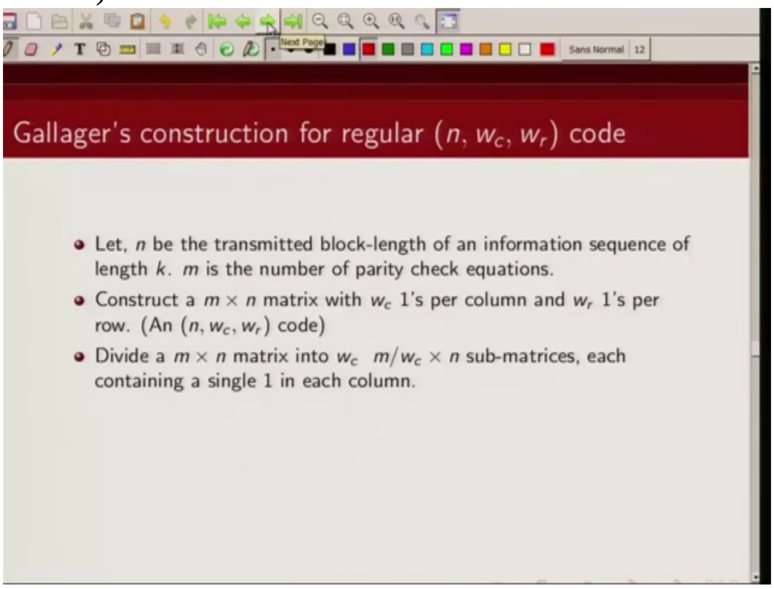
So if l is the block length and m is the number of parity check equations

(Refer Slide Time 19:15)



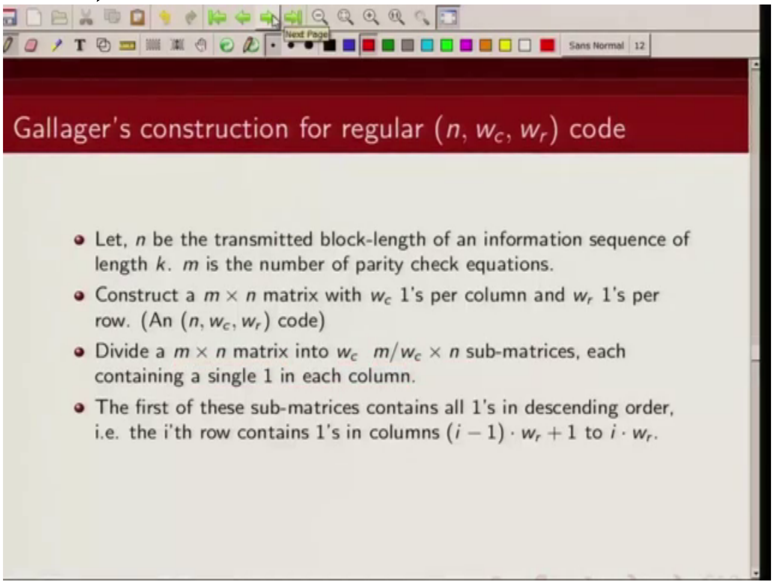
and if you are asked to design a regular L D P C code with w c 1s per column and w r 1s per row you can follow this procedure.

(Refer Slide Time 19:28)



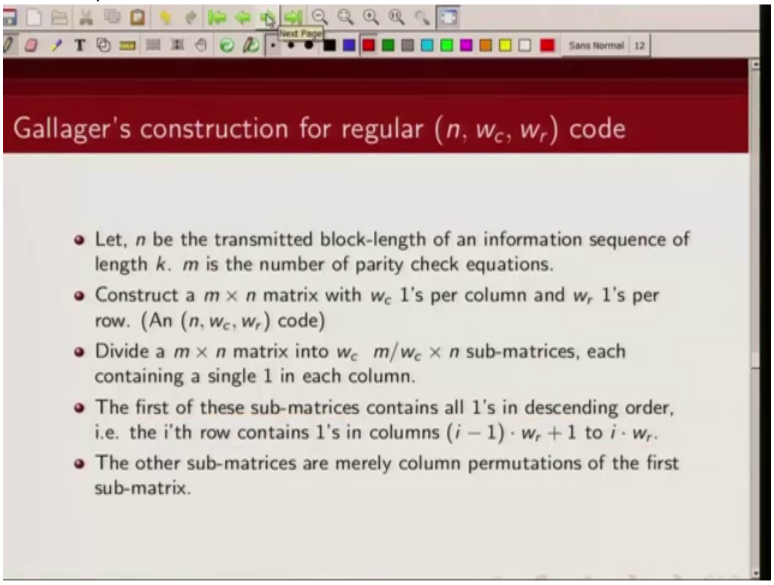
So divide m cross n matrix which is your parity check matrix. You divide them into w c m by w c into n sub matrices such that in each of the sub matrices your column will only have a single 1.

(Refer Slide Time 19:53)



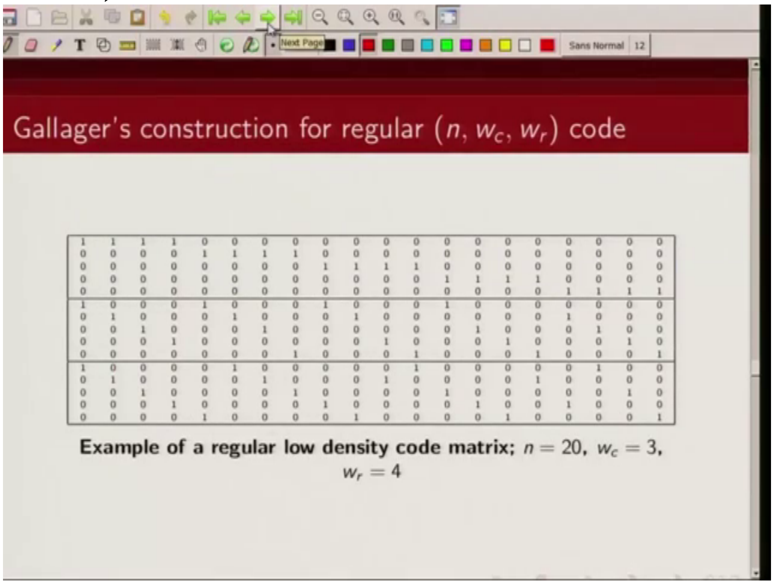
Next you start writing; so in each of the sub matrices, what you do is you write once in the descending order. So the ith row will have 1s from location i minus 1 w r plus 1 to i into w r. So what you do is, I will just illustrate

(Refer Slide Time 20:15)



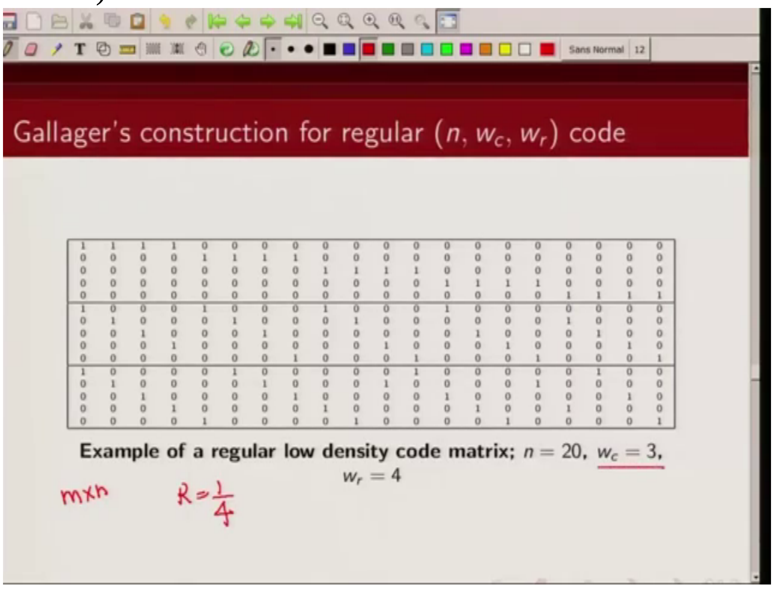
basically,

(Refer Slide Time 20:16)



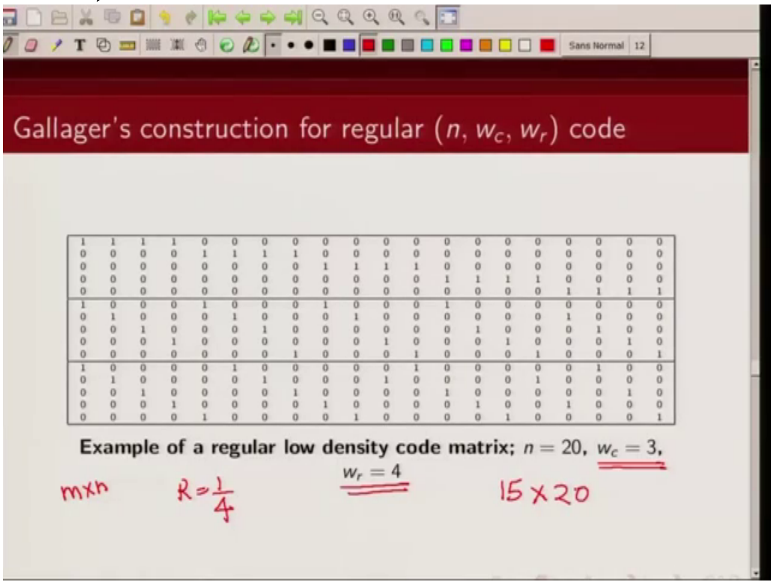
so what you do is first you have a m cross n matrix. Now this was a rate half code, remember rate 1 by 4 code remember

(Refer Slide Time 20:26)



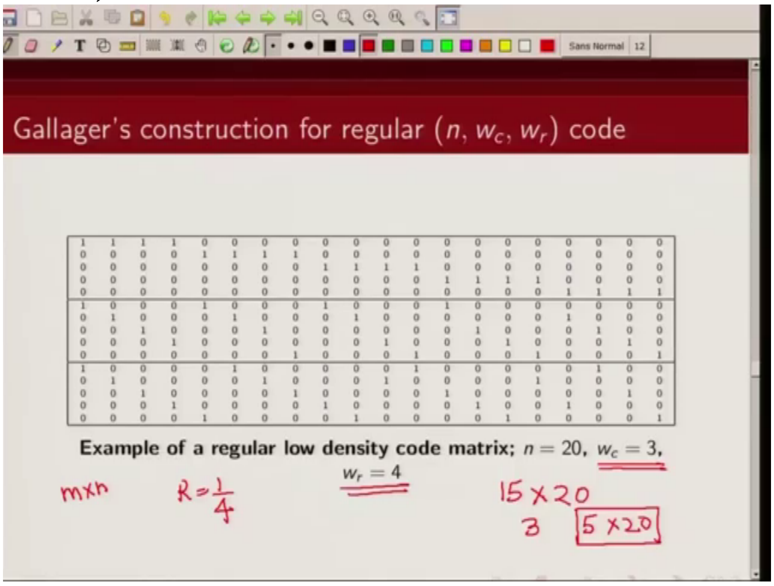
w c is 3 w r is 4, so this is 15 cross 20 matrix.

(Refer Slide Time 20:35)



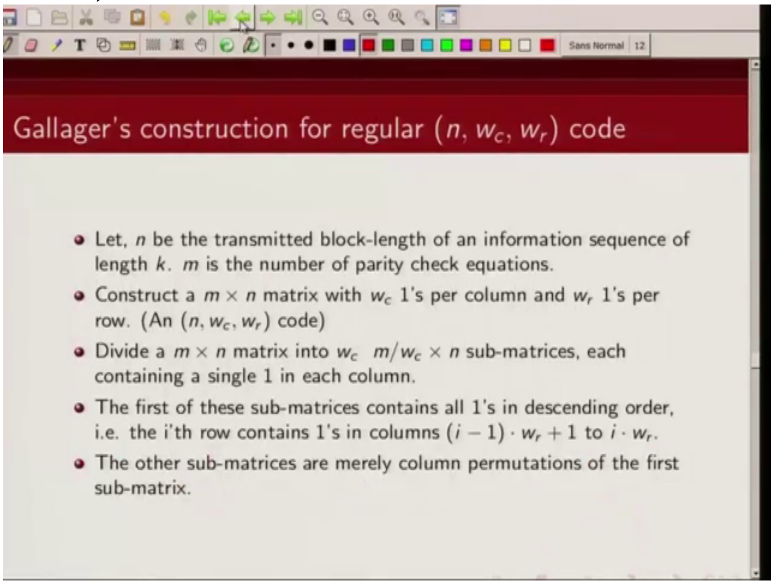
Now what you do is you divide this 15 cross 20 matrix into three 5 cross 20 matrices, sub matrices, right. That was the

(Refer Slide Time 20:49)



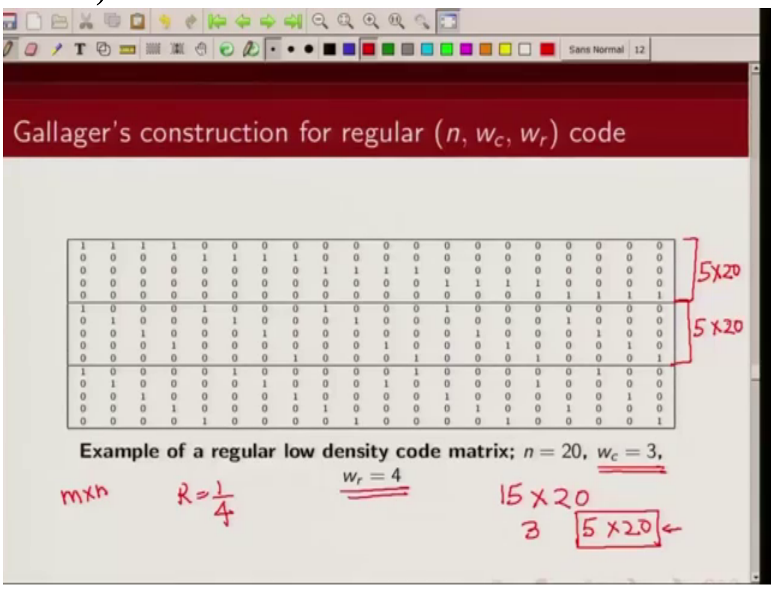
first step.

(Refer Slide Time 20:51)



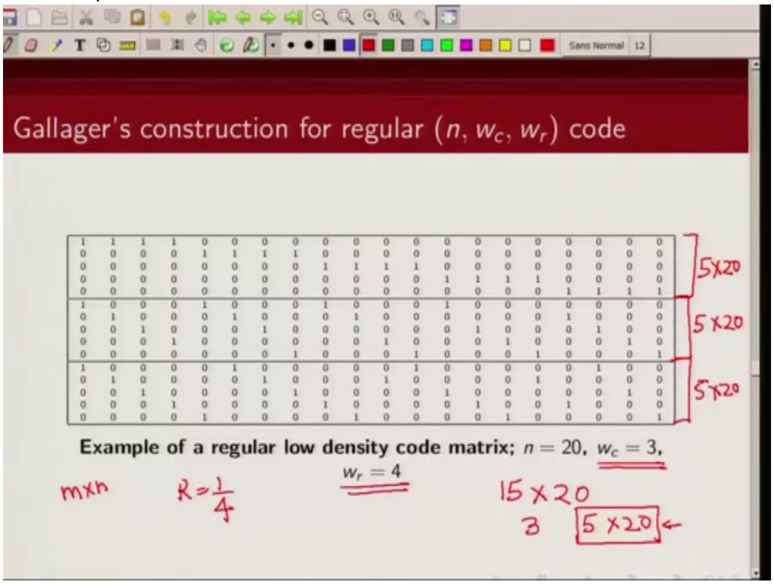
Divide an m cross n matrix into w c m by w c into n sub matrices. So once you divide, so this is one 5 cross 20 sub matrix. This is another 5 cross 20 sub matrix.

(Refer Slide Time 21:12)



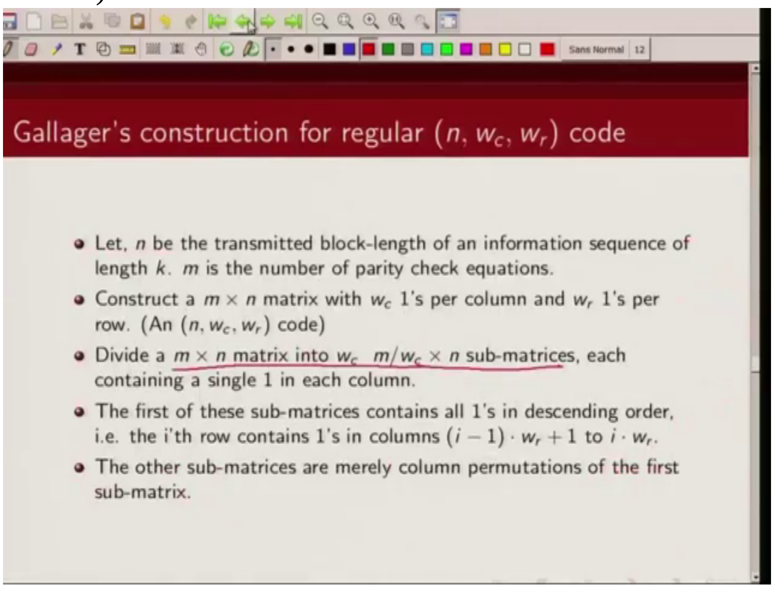
This is another 5 cross 20 sub matrix.

(Refer Slide Time 21:16)



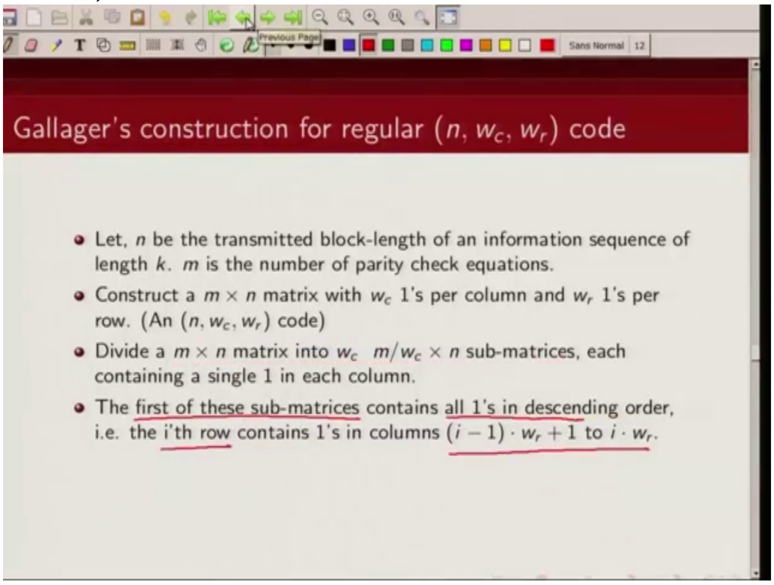
And each of these sub matrices should have only 1s and what you do, you start writing 1s in the descending, so you write, start writing 1 here from location 0, now w r in this case is 4, so in the 0th row you start from column 0 to 3, next row you start writing from 4 to 7, like that you start writing 1s. So that's what I have written here.

(Refer Slide Time 21:45)



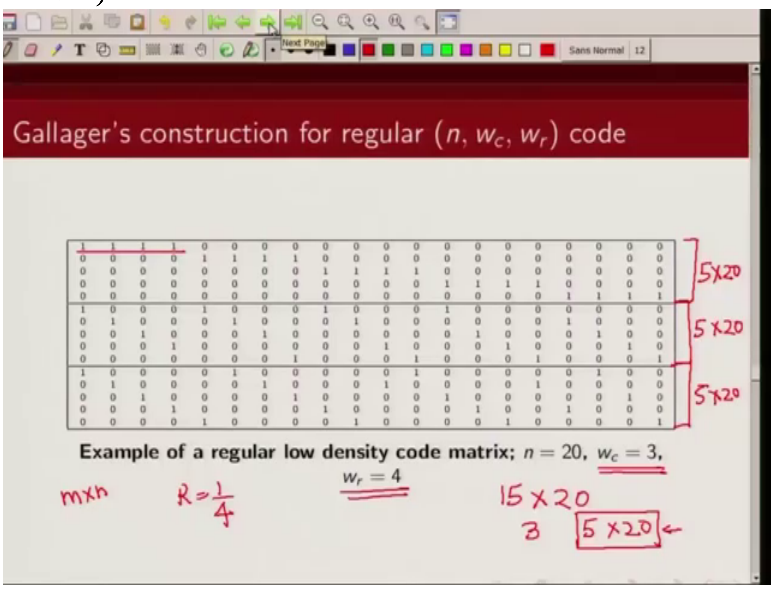
That first of these sub matrices contains all 1s in the descending order such that ith row contains 1s from location i minus 1 into w r plus 1 to i times w r. So

(Refer Slide Time 22:07)



once you have constructed this sub matrix,

(Refer Slide Time 22:10)

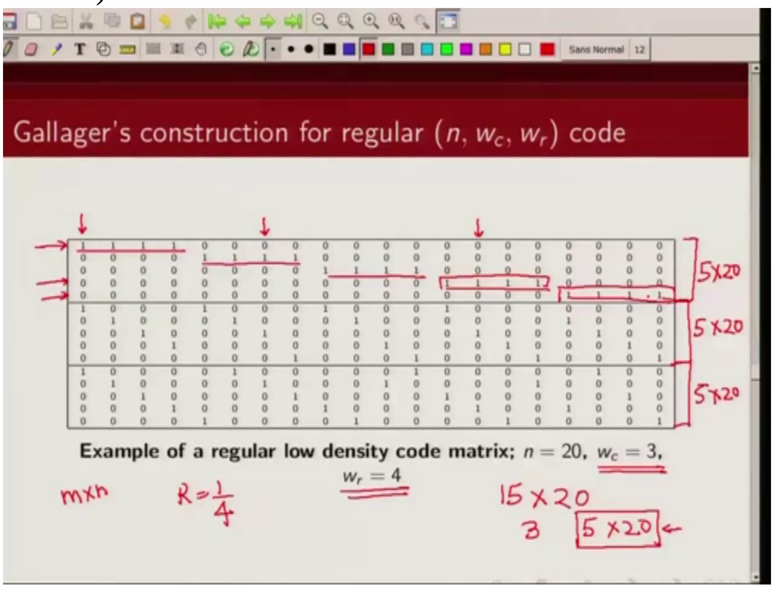


this 5 cross 20 sub matrix by putting 1s like this in descending, like this. Once you have constructed this, rest all are 0s. Now note that each of the columns here have one 1s.

(Refer Slide Time 22:28)

And what about, you can check any column of these sub matrix. It has only one 1. And note what is the row weight. Now each of the row here has w r which is four number of 1s. You can check this is four 1s. You can check here, this is four 1s here; these are the four 1s here. You can check these, these are four 1s. So what you have created is you have created

(Refer Slide Time 22:58)

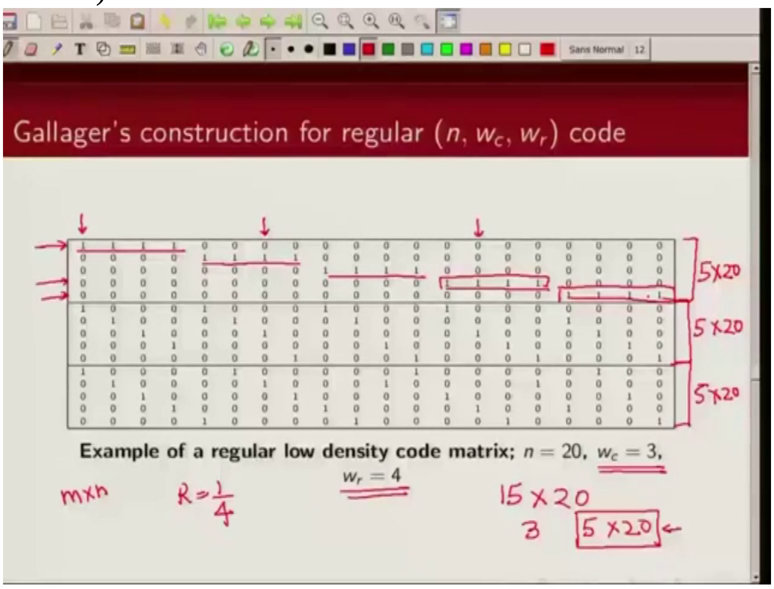


a sub matrix which has column weight

(Refer Slide Time 23:01)

1 and row weight w r. Now next

(Refer Slide Time 23:07)

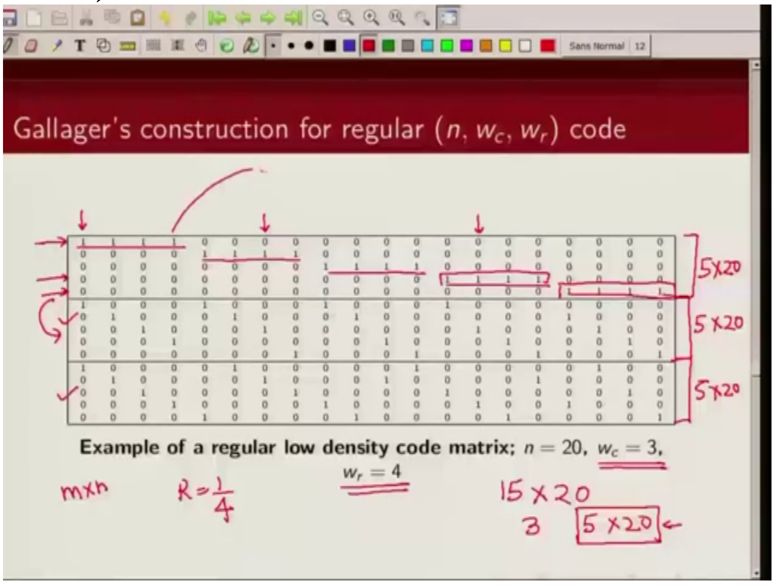


to get w c column weight what do I need? I need to design a similar sub matrix here which will have one 1 in each of the column and w r 1s in each row. Now how do I get this sub matrix and this sub matrix? So what I can do is I can do column

(Refer Slide Time 23:35)

permutation. So for example I can move

(Refer Slide Time 23:38)

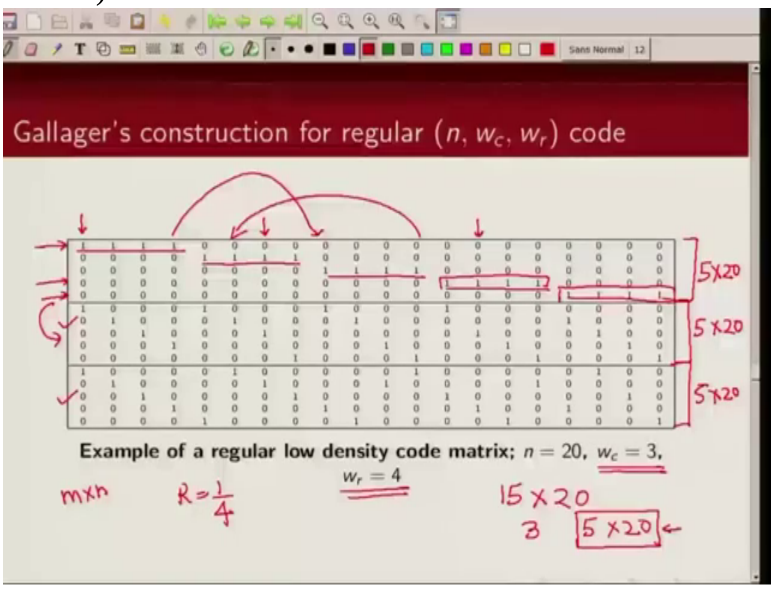


this column here, I can move this column here, I can do column permutation. Now when I do column permutation I do not change the weight of the column. It is still 1. And I do not change

(Refer Slide Time 23:53)

the weight of the rows. It is still 4. So

(Refer Slide Time 23:57)

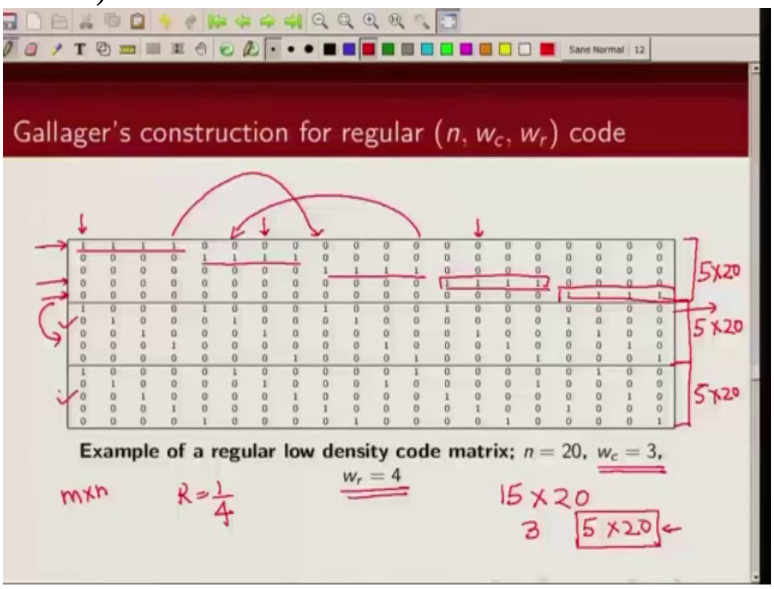


by doing column permutation I will get this sub matrix which will again have one 1 in each column and four 1s in each row.

(Refer Slide Time 24:10)

The same thing I will do

(Refer Slide Time 24:12)

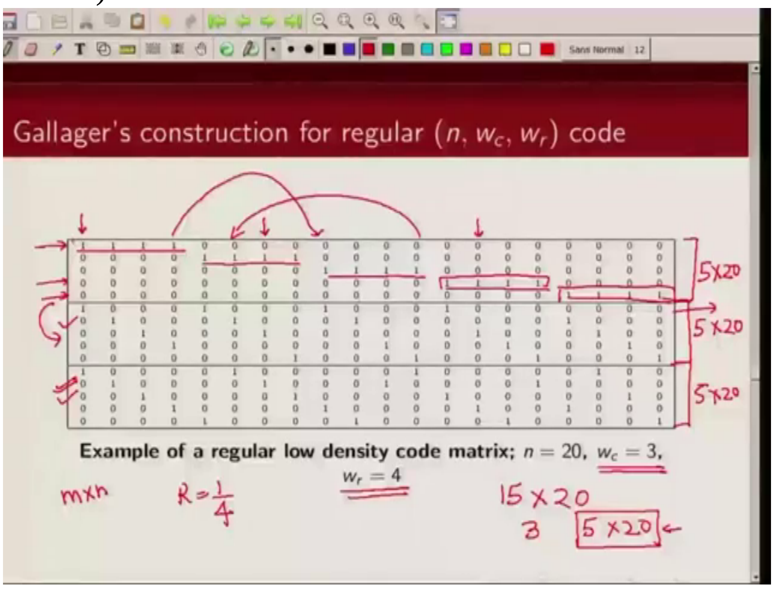


for each sub matrix. I will again do column permutation of this matrix and I will get this sub matrix. So the subsequent sub matrices are obtained by doing column permutation and by doing column permutation I am not disturbing the weight, the row weight or the column weight. The column weight of this sub matrix is still 1; the row weight is still 4. Now once I do this then I am able to design these three 5 cross 20 sub matrices each of them have

(Refer Slide Time 24:47)

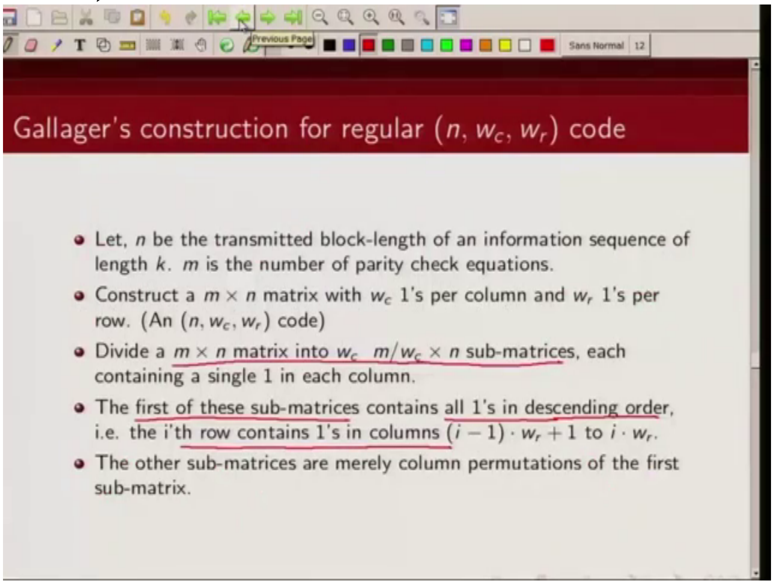
column weight 1. So overall column

(Refer Slide Time 24:49)



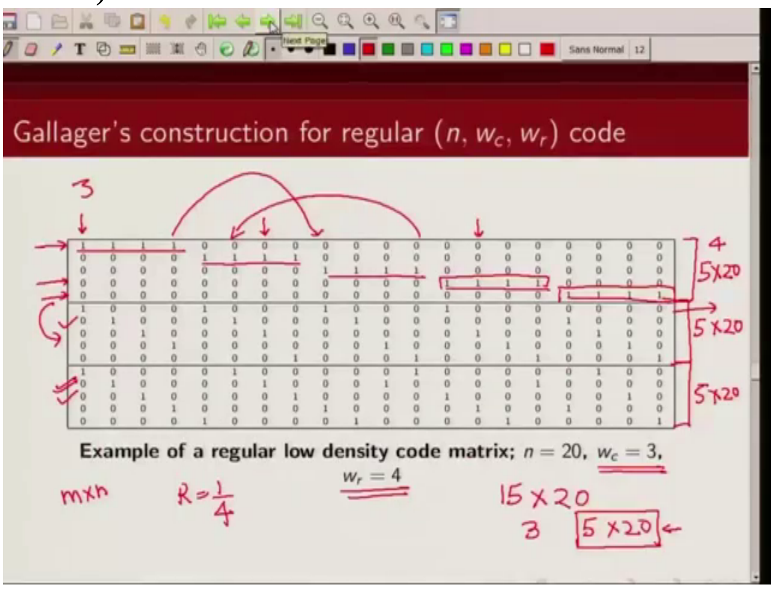
weight will be 3 and overall row weight is still 4. So I am able to design a low density parity check matrix which has three 1s in each column and four 1s in each row.

(Refer Slide Time 25:08)



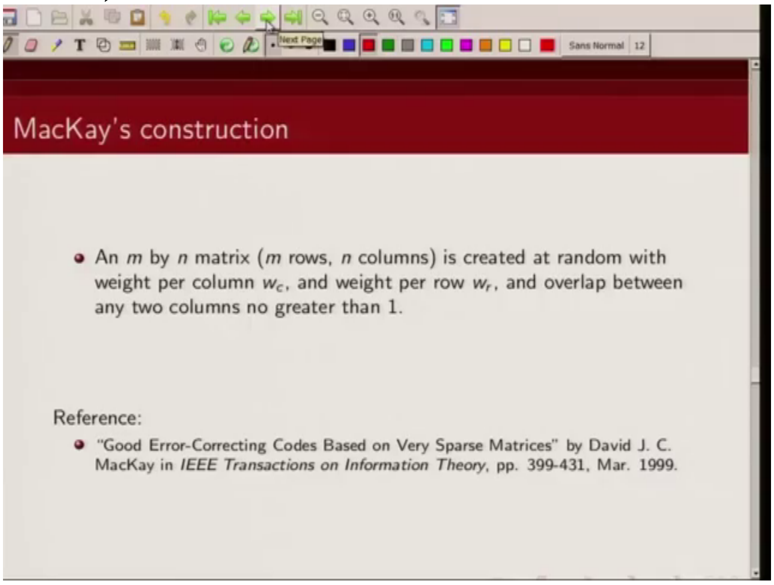
So that's what I said that other sub matrices are merely column permutations of the first sub matrix. So this is

(Refer Slide Time 25:19)



Gallager's construction of a regular L D P C code which has w c 1s in each column and w r 1s in each row.

(Refer Slide Time 25:35)

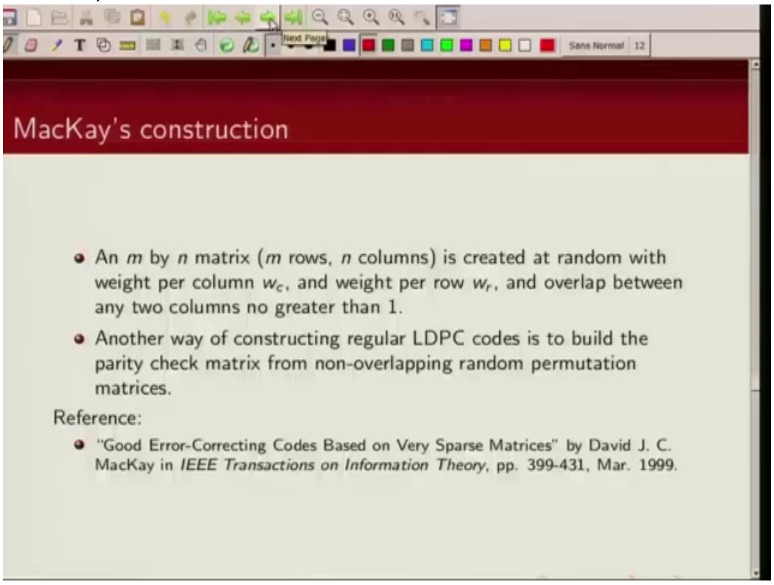


Now we will talk about simple constructions based on permutation matrix, these were given by MacKay, you can read this paper Good Error-Correcting Codes on Very Sparse Matrices by David MacKay which appeared in I triple E transactions on Information Theory in May 1999, so one way of designing a m cross n matrix which has w c column weight and w r column weight is you can randomly put 1s ensuring these criteria are satisfied and also you want to ensure that overlap between 2 rows

(Refer Slide Time 26:16)

of this parity check matrix is not more than 1. Otherwise you will have cycles 4

(Refer Slide Time 26:23)

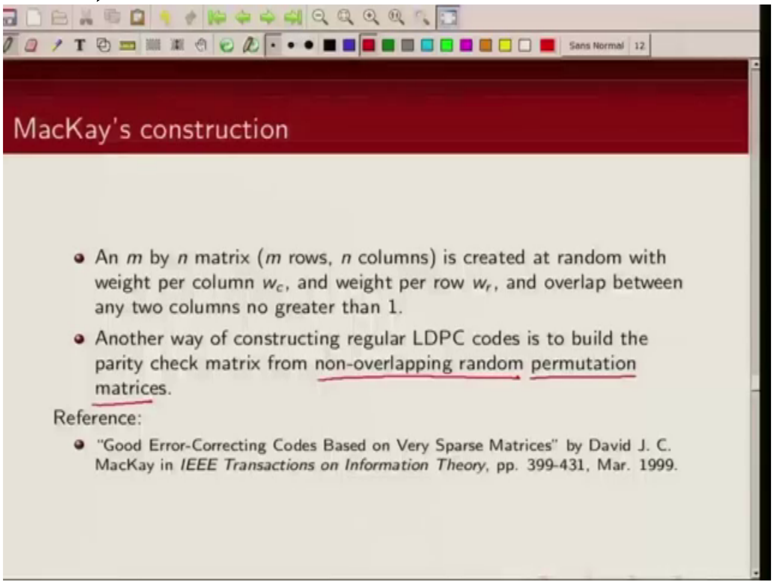


in your L D P C code. Now next what we are going to talk about is how we can use permutation matrix to

(Refer Slide Time 26:38)

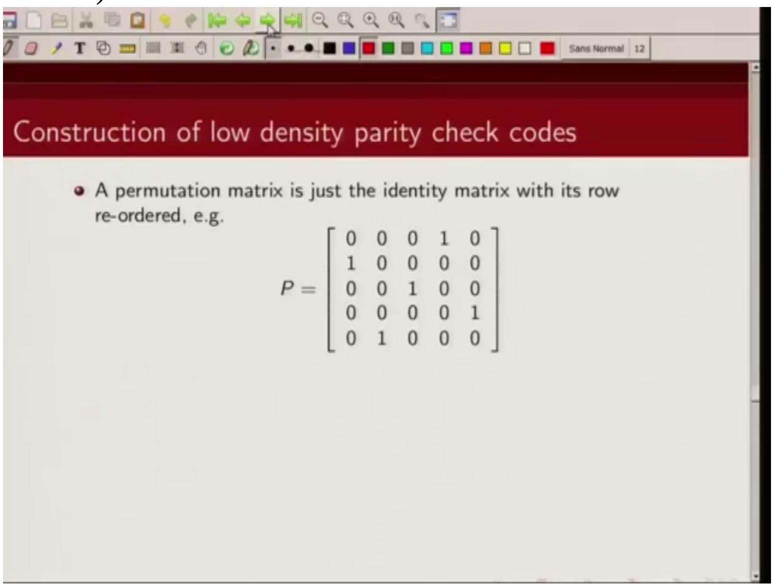
design our L D P C codes? So that's what I am going to show

(Refer Slide Time 26:43)



in the next few slides. So let's define

(Refer Slide Time 26:46)

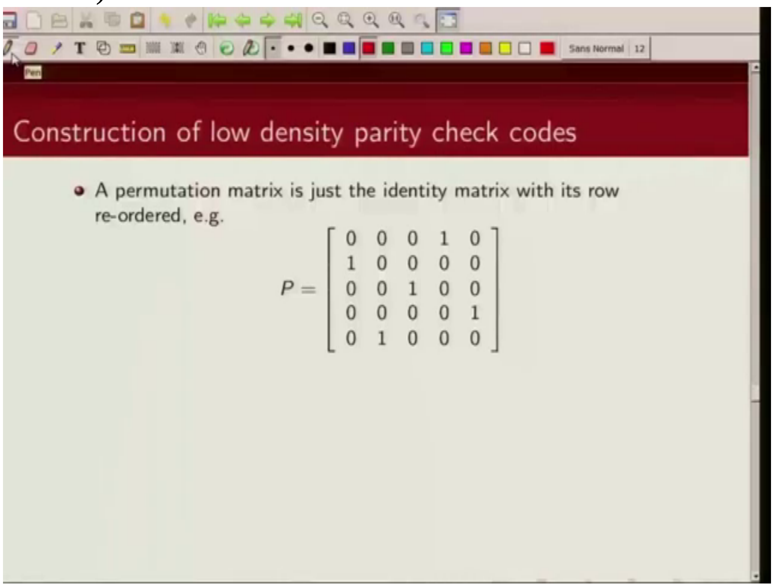


what a permutation matrix, so a

(Refer Slide Time 26:49)

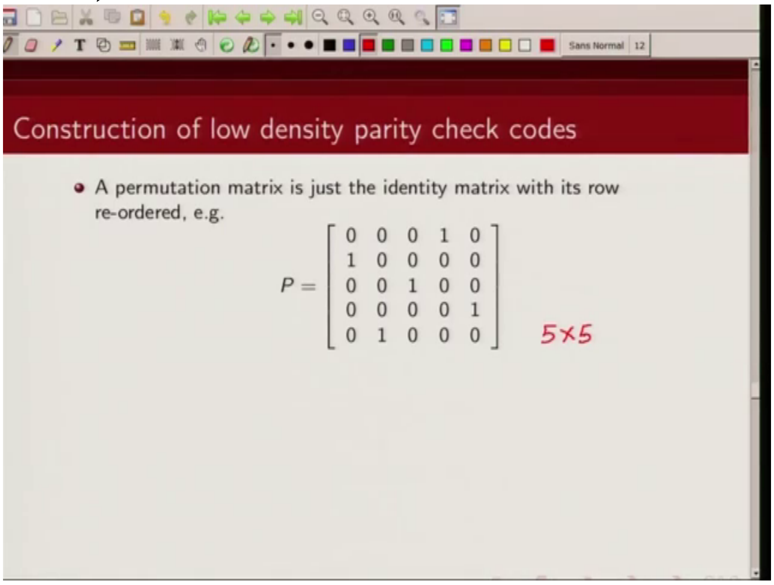
permutation matrix is an identity matrix which is row reordered. So

(Refer Slide Time 26:55)



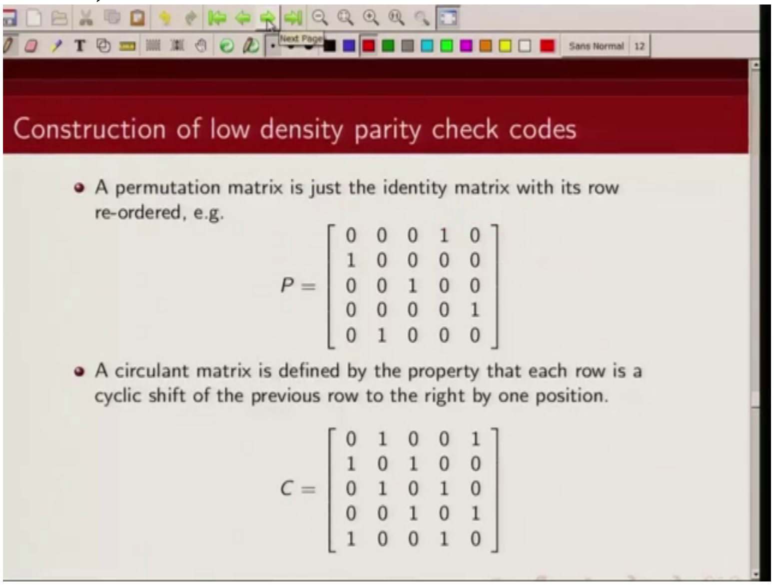
this is an example of 5 cross 5 permutation matrix.

(Refer Slide Time 27:00)



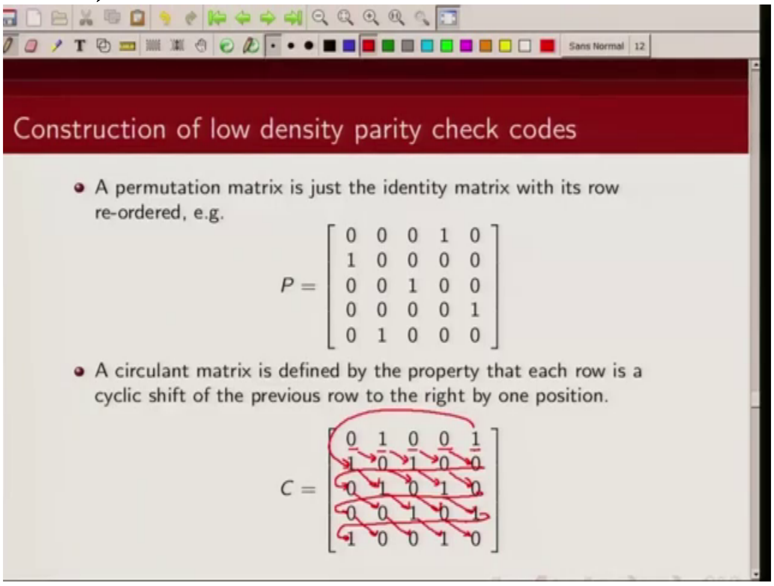
You can see each row has one 1 and each column, each row has only one 1, and each column has a single 1. And this is just an identity matrix with row reordered. Now we could also use

(Refer Slide Time 27:17)



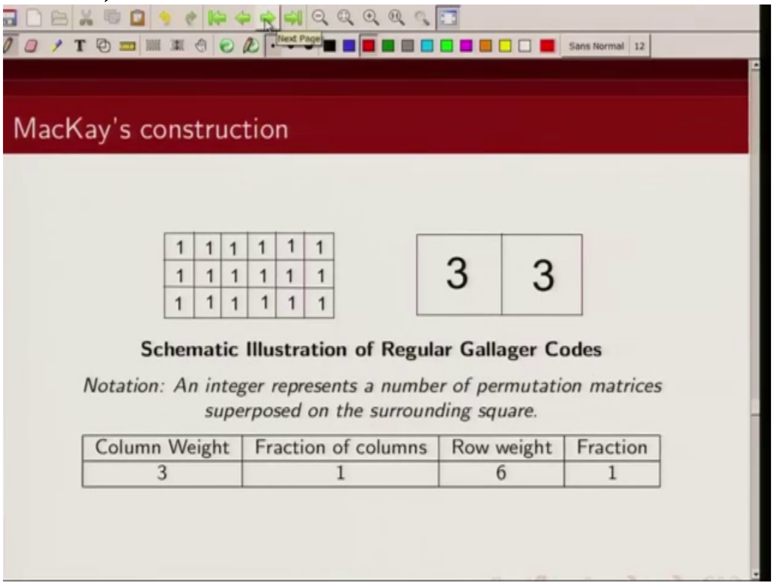
a circulant matrix to design our L D P C codes. Now what is a circulant matrix? So a circulant matrix has a property that each row is just a circular shift of previous row. So for example you take this example, this first row is 0 1 0 0 1, now this 0 comes here, this 1 comes here, this 0 comes here, this 0 comes here and this 1 comes here. So that's your second row. The third row, this 0, 1 comes here. This 0 comes here, 1 comes here, this 0 comes here and this 0 comes here. Similarly here this 0 comes here, this 0 comes here, 1 comes here, 0 comes here, and 1 comes here. And likewise this 1 comes here, this 0 comes here, this 0 comes here, 1 here, 0 here, so you can see

(Refer Slide Time 28:17)



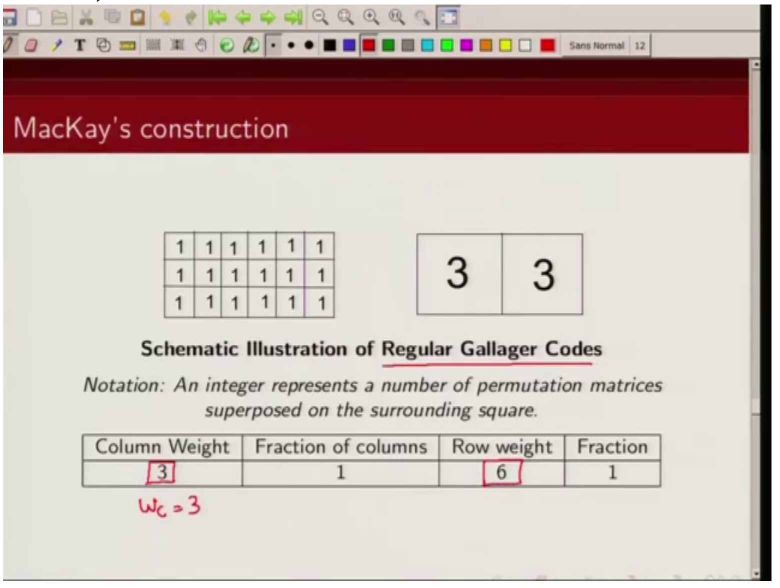
each row is a circular shift of the previous row. So we will now show how, we will randomly construct these permutation matrix. This is just an identity matrix which is row reordered. Now we will show how we can construct our L D P C codes using this permutation matrix.

(Refer Slide Time 28:43)



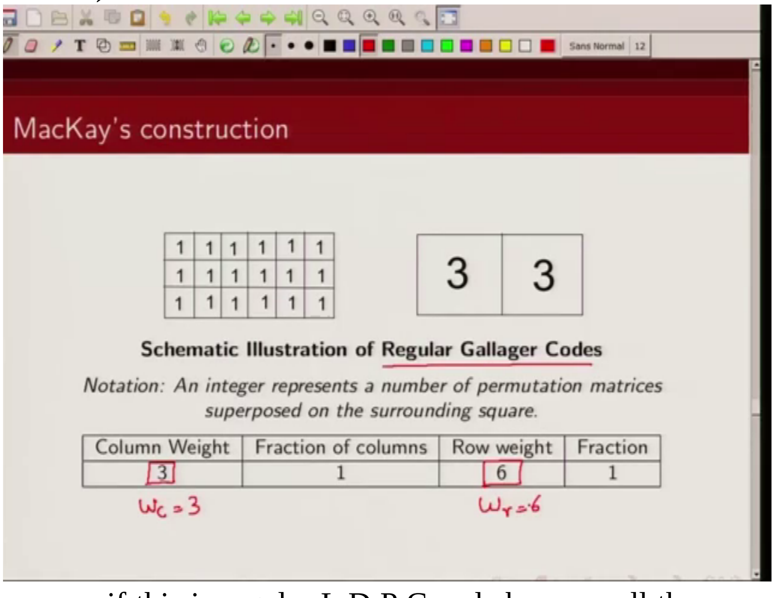
So the first example that we are going to consider is an example of a regular L D P C code. Now this regular L D P C code has column weight 3 so w c is 3 and

(Refer Slide Time 29:01)



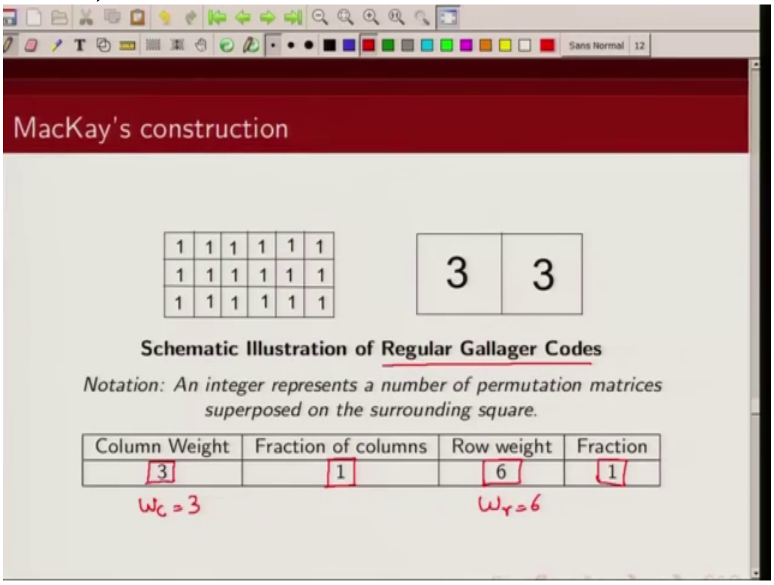
row weight is 6.

(Refer Slide Time 29:03)



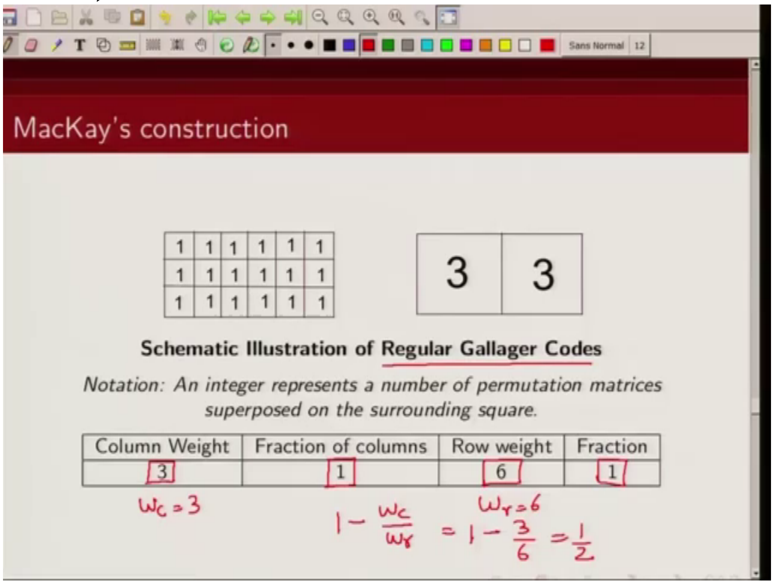
So w r is 6. You can see if this is regular L D P C code because all the rows and the columns have the same weight,

(Refer Slide Time 29:14)



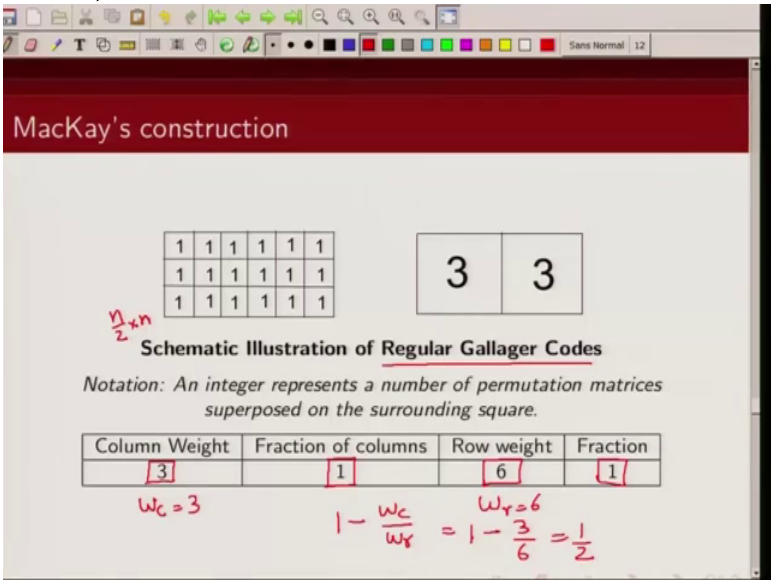
Ok. Now how can we use permutation matrix to design this? So let's take this example. Let's say you have to design an, so what's the rate here? Rate here is 1 minus w c by w r. So this is 1 minus 3 by 6 so that's rate half. m will

(Refer Slide Time 29:38)



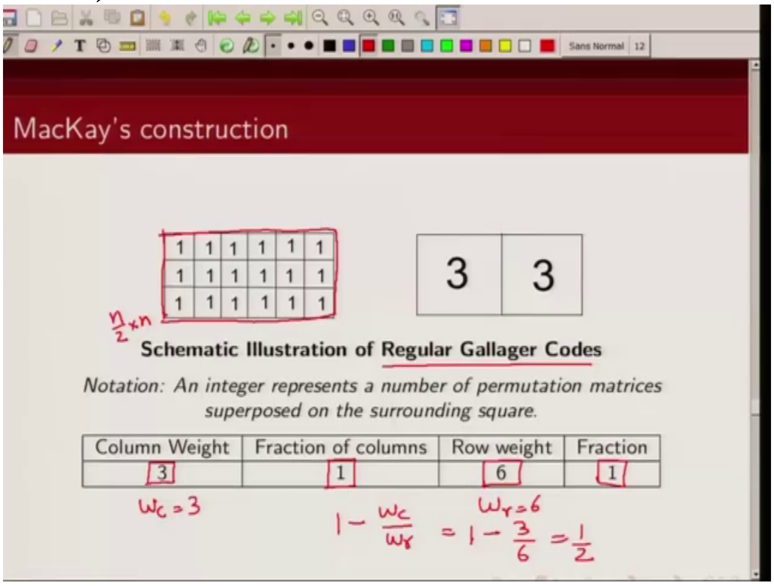
be n by 2. So m here would be n by 2. So what we did was, so this is you can think of its n by 2 cross n. So what we did was

(Refer Slide Time 29:49)



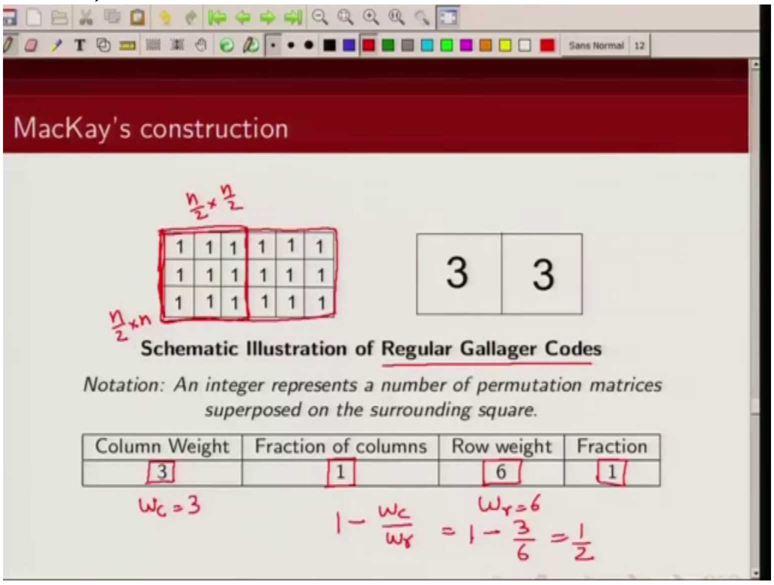
we divided this n by 2 cross n matrix into sub matrices

(Refer Slide Time 29:56)



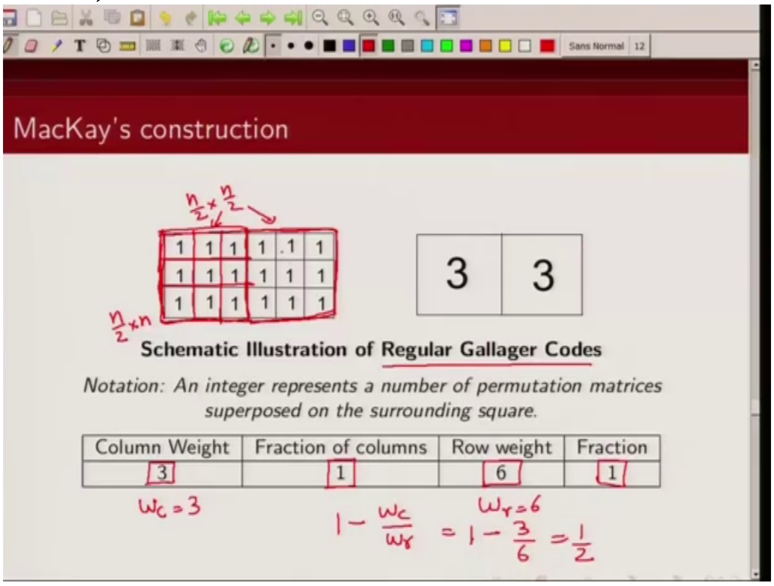
in this particular way. So this is, so this one that you see. This is n by 2 cross n by 2 matrix

(Refer Slide Time 30:11)



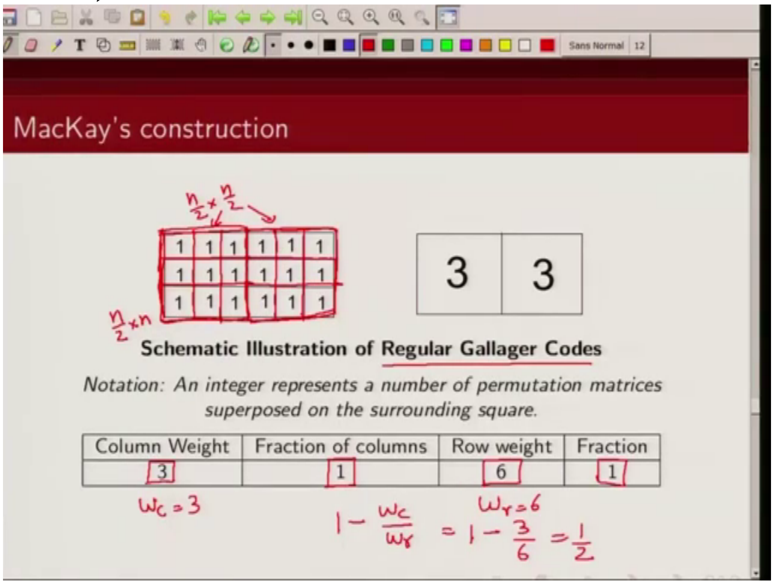
and this is another n by 2 cross n by 2 matrix. So these are 2 n by 2 cross n by 2 matrices which are further divided into sub matrices n by 6 by n by 6.

(Refer Slide Time 30:26)



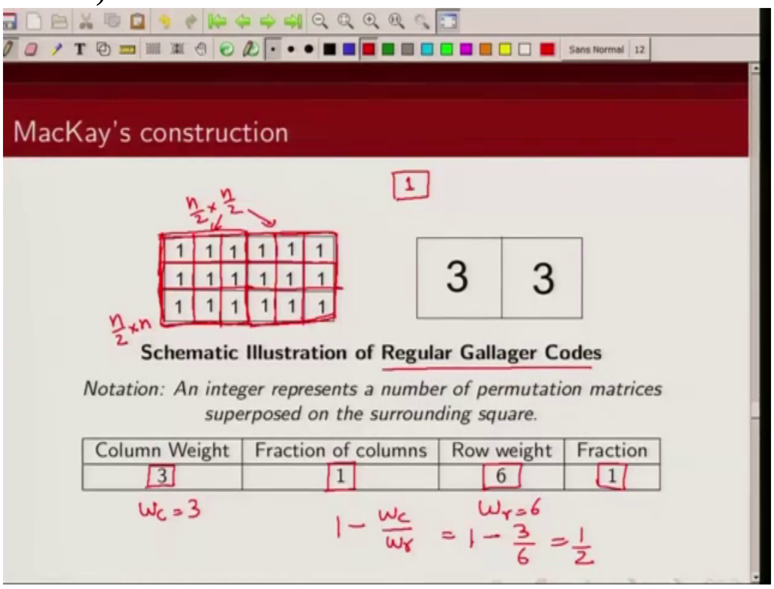
So each of them are your n by 6 by n by 6 matrix.

(Refer Slide Time 30:32)



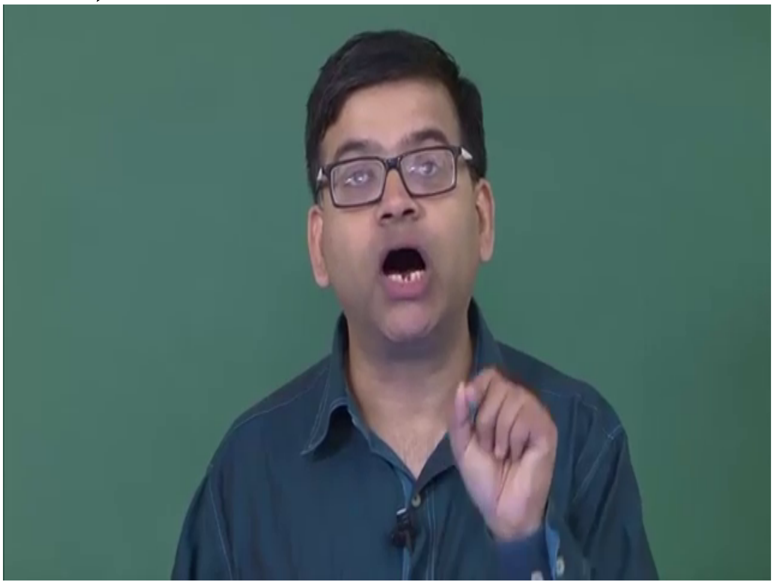
And what does this signifies? This

(Refer Slide Time 30:36)



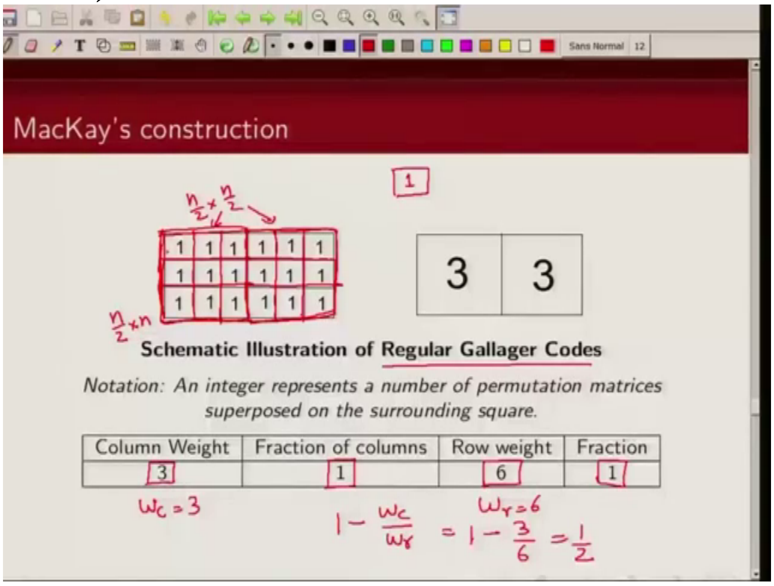
signifies that this is the notation we are using to denote a permutation

(Refer Slide Time 30:41)



permutation matrix.

(Refer Slide Time 30:44)

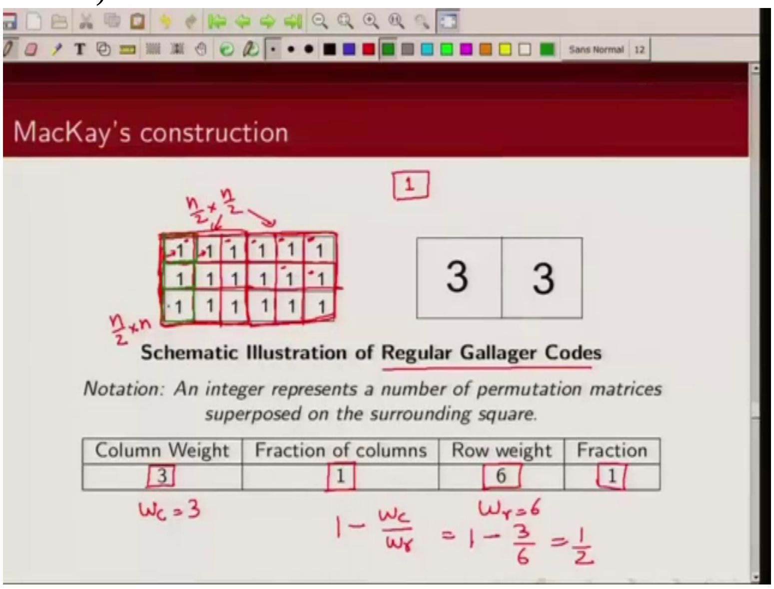


So this is one random permutation matrix. This is another random permutation matrix where location of one is different from what was here in this particular matrix. So this is another random permutation matrix. So these 1s that you see here, these are all random permutation matrices, Ok. Now note that we need a column weight of 3. Now if we stack, if we stack 3 permutation matrices like this, now each of these permutation matrix has one 1 in its column. So

(Refer Slide Time 31:25)

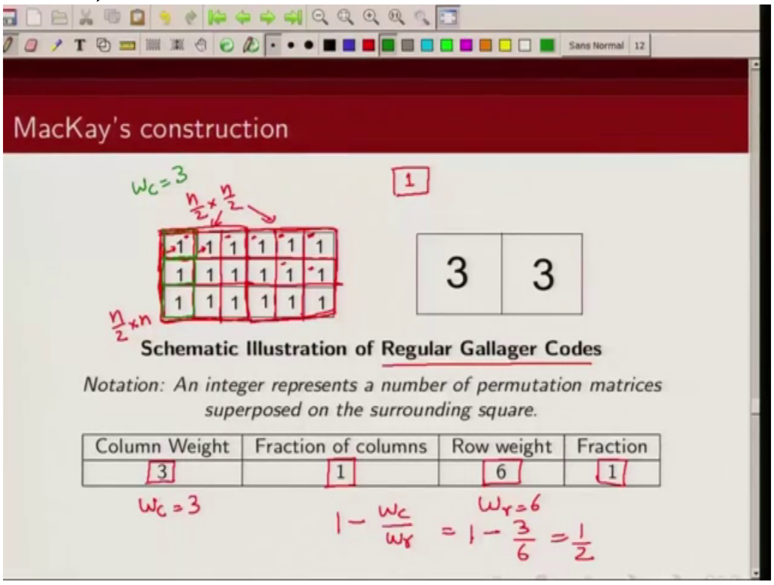
over all we will get,

(Refer Slide Time 31:26)



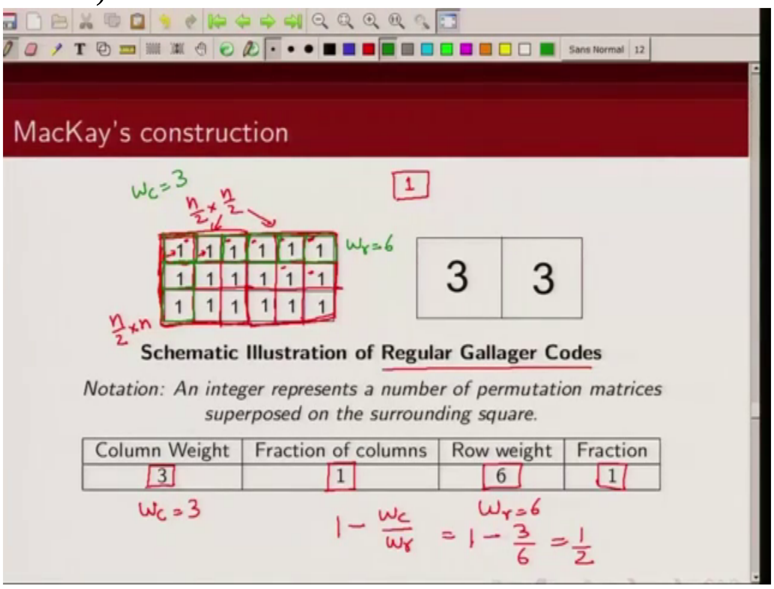
column weight 3. And

(Refer Slide Time 31:31)



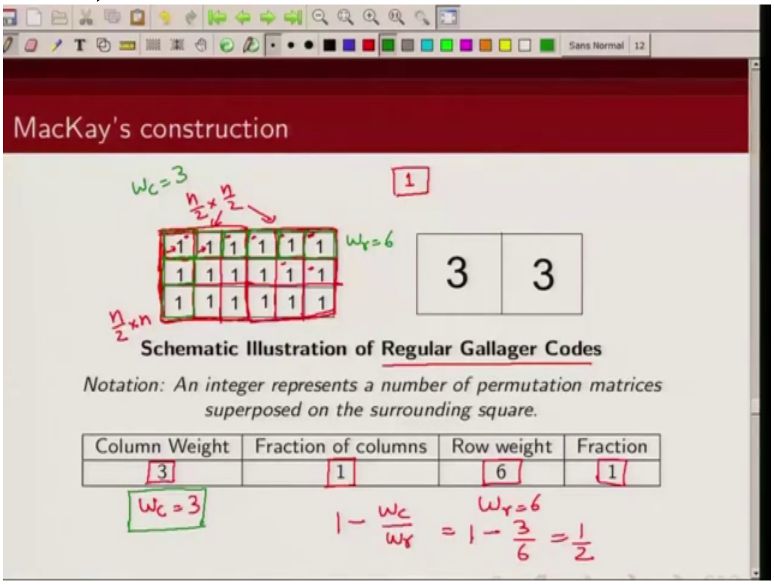
if we stack 6 of them like this, then each row also has one 1 so we will get overall six 1s

(Refer Slide Time 31:46)



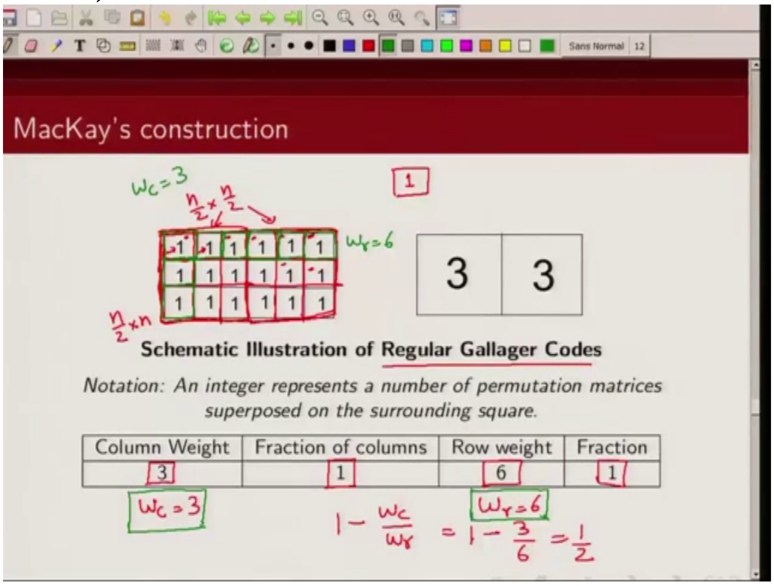
in each row. So this way we can generate a L D P C code with these parameters, column weight 3 and

(Refer Slide Time 32:01)



row weight 6.

(Refer Slide Time 32:04)

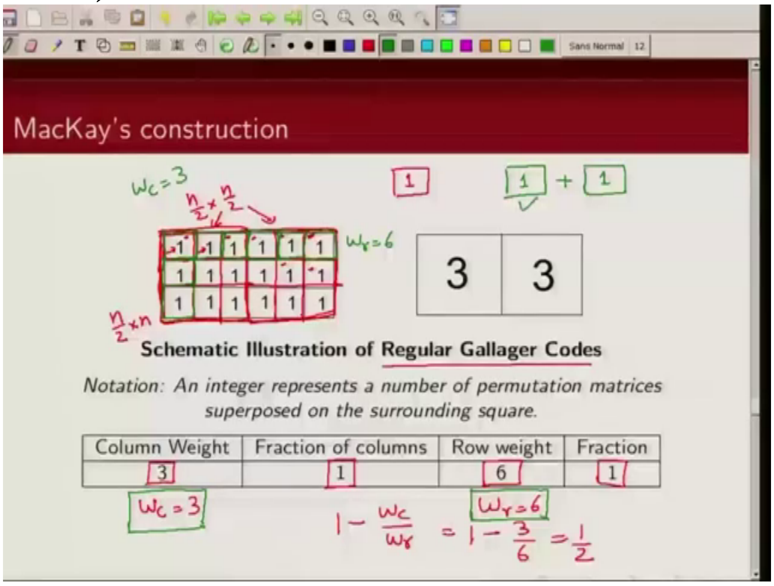


Now the same thing can be generated using by overlapping of random permutation matrices. Now what do I mean by overlap of random permutation matrix? So let's say you have a permutation matrix. And you add another permutation matrix. Now please note you ensure that

(Refer Slide Time 32:32)

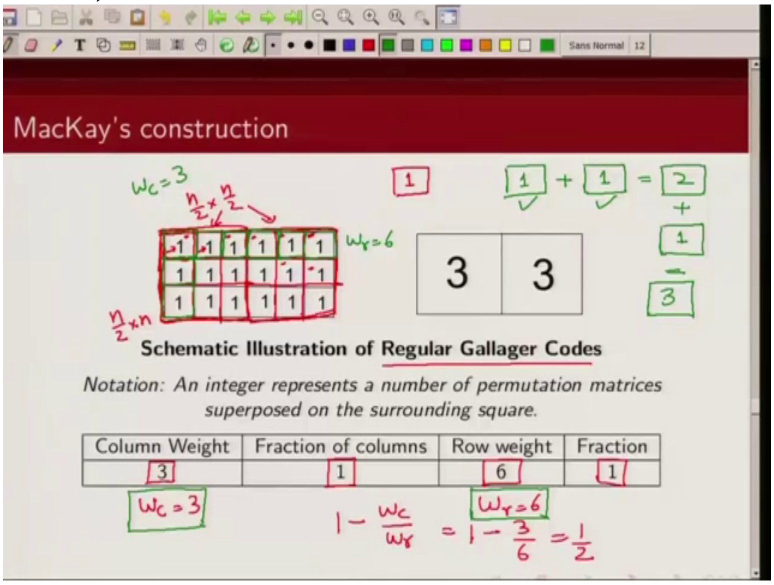
there is no overlaps between 1s in this matrix

(Refer Slide Time 32:37)



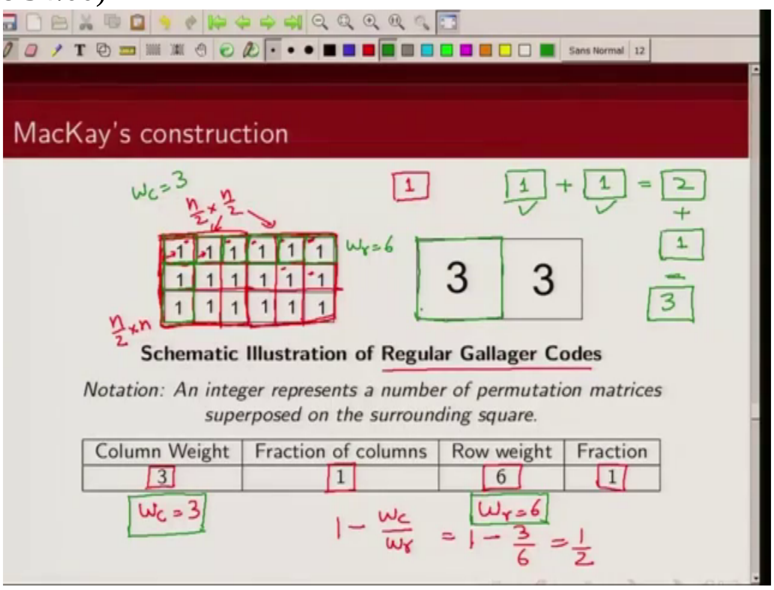
and this matrix. If you add these 2 permutation matrix what will you get? You will get a matrix which will have two 1s in each row and two 1s in each column. Now if I add another permutation matrix and I ensure that this, the 1 in this permutation matrix does not overlap with 1s in this matrix which I got by adding to permutation matrix, then the resultant permutation matrix that I will get will have

(Refer Slide Time 33:17)



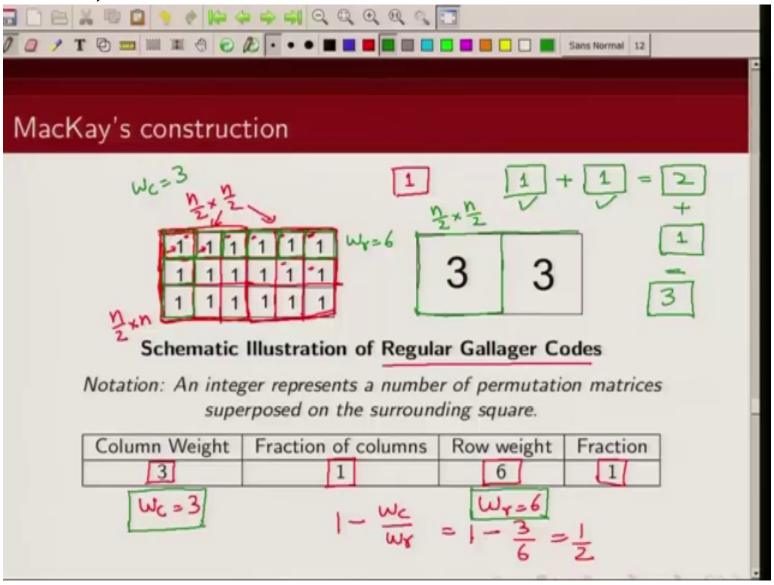
row weight 3 and column weight also 3. So another way of designing this L D P C code is by overlapping of random permutation matrices. Please note when I overlap them I have to ensure that there is no 1s that are getting overlapped. Then only I will be able to retain the weight. So if I overlap 3 such random permutation matrices and I ensure that there is no overlap between the 1s between each of these 3 random permutation matrix then what I will get is a matrix which will have three

(Refer Slide Time 34:00)



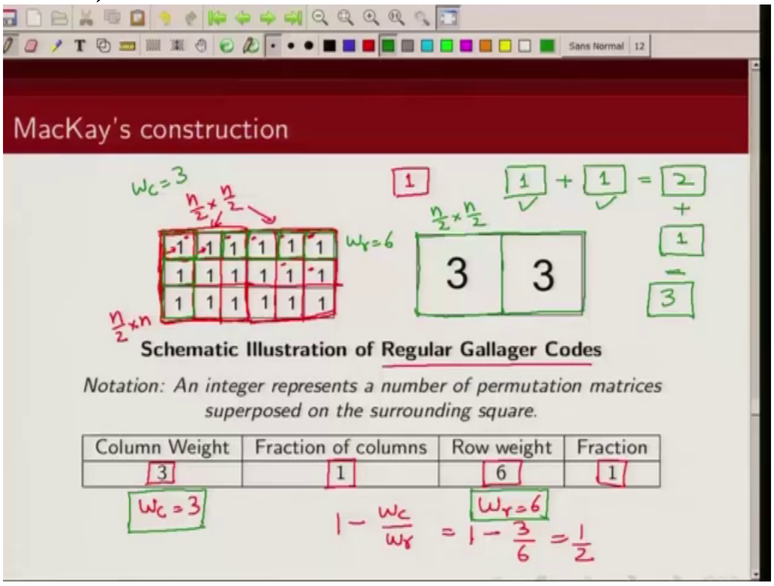
1s in each row and three 1s in each column. I can construct another n by 2 by n by 2 matrix, this one

(Refer Slide Time 34:10)



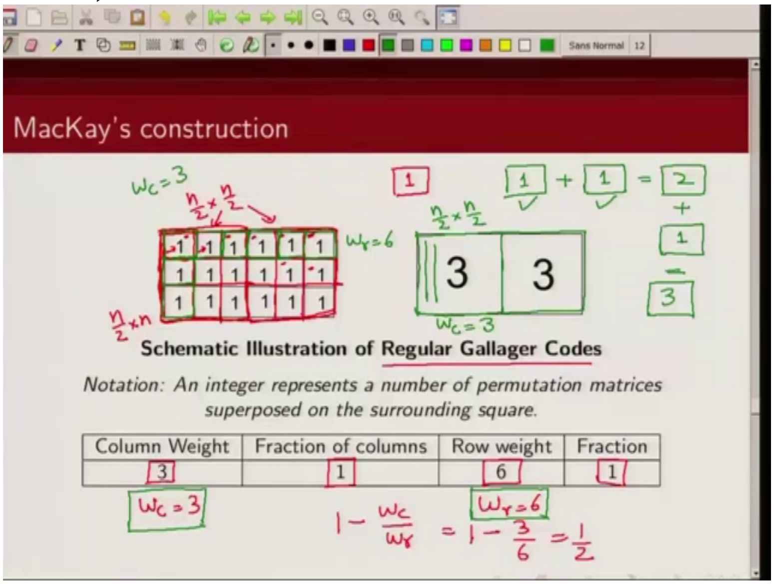
which is again by

(Refer Slide Time 34:14)



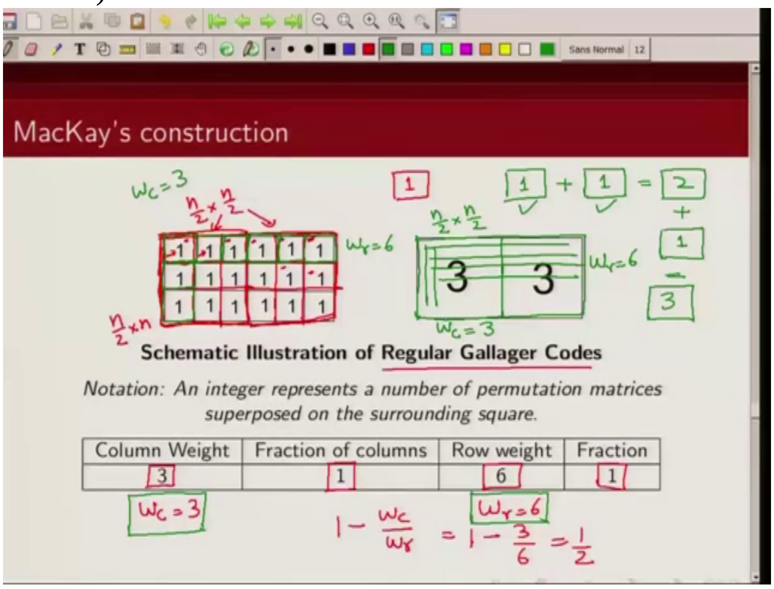
overlapping of three different random permutation matrices ensuring that there is no overlap of 1s; I can get another matrix which will have three 1s in each row and three 1s in each column and this will ensure that each column of this matrix will have column weight 3 and

(Refer Slide Time 34:37)



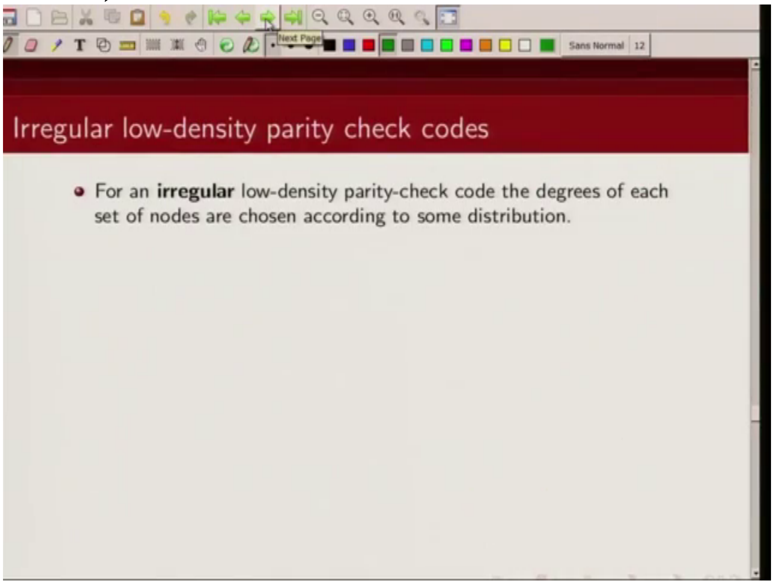
each of the rows of this matrix will have

(Refer Slide Time 34:43)



row weight 6. So this is another way I can use the permutation matrices to construct my regular L D P C code.

(Refer Slide Time 35:00)

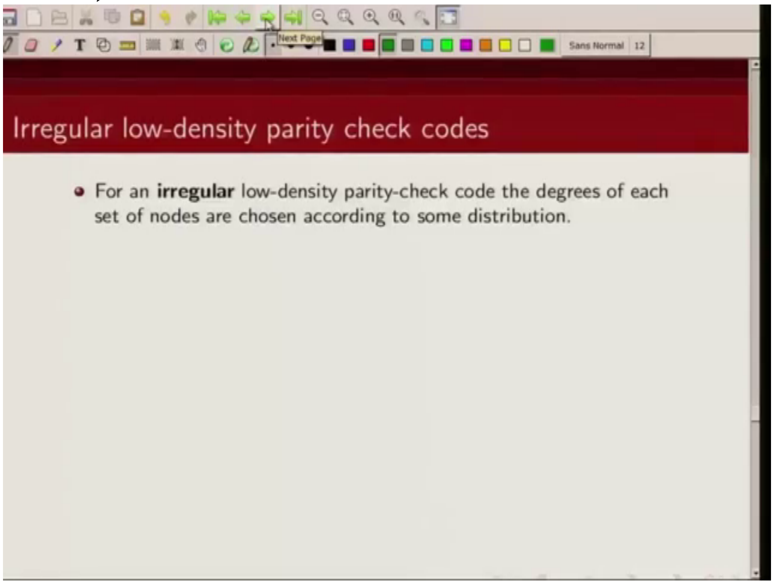


Now let's talk about what is an

(Refer Slide Time 35:03)

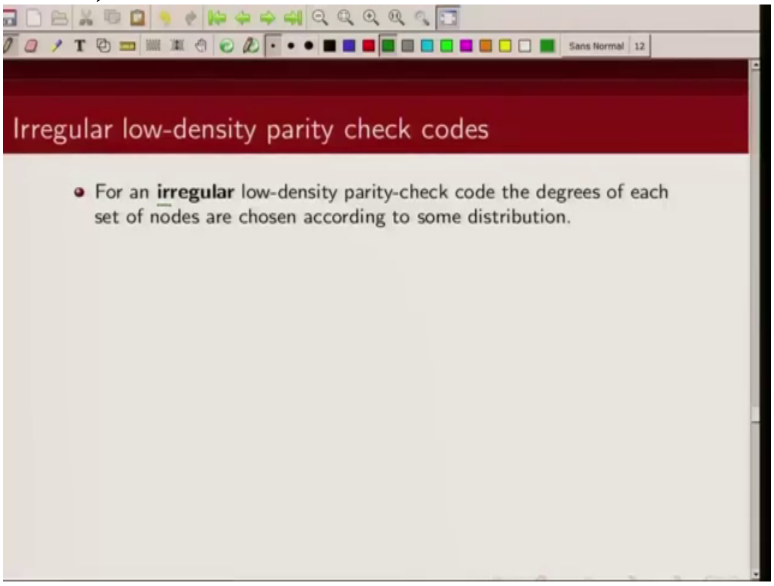
irregular L D P C code. So irregular L D P C code as opposed to regular L D P C code, the number of 1s in each column and number of 1s in each row are different. So we

(Refer Slide Time 35:17)



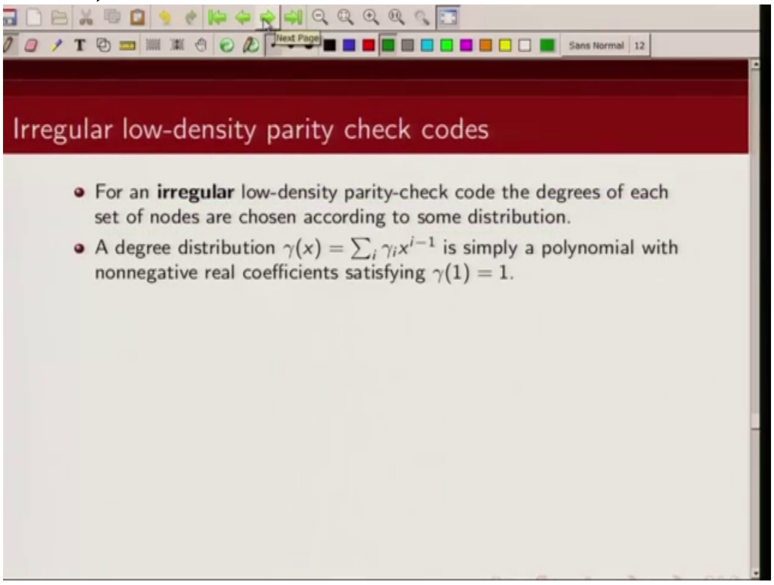
will have to specify the degree, the node distribution, the column node distribution as well as the row node distribution. So for an

(Refer Slide Time 35:28)



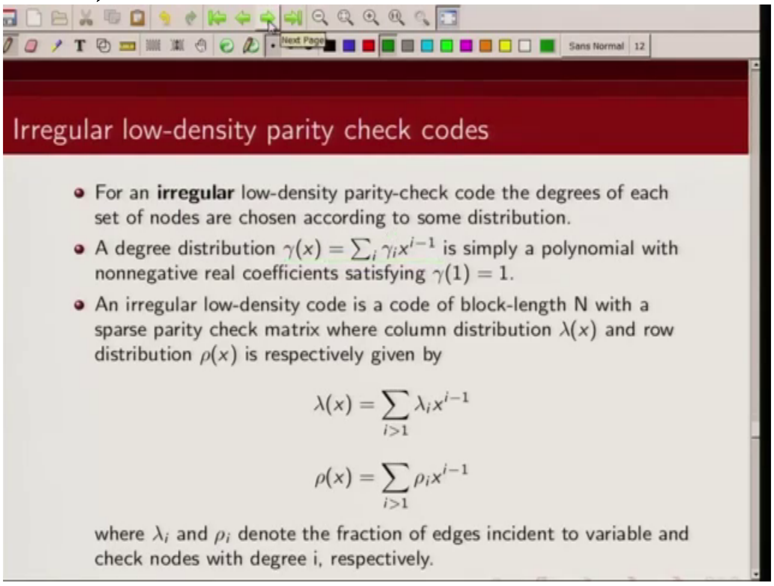
irregular L D P C code, we define the distribution of column nodes as well as row nodes according to some distribution.

(Refer Slide Time 35:41)



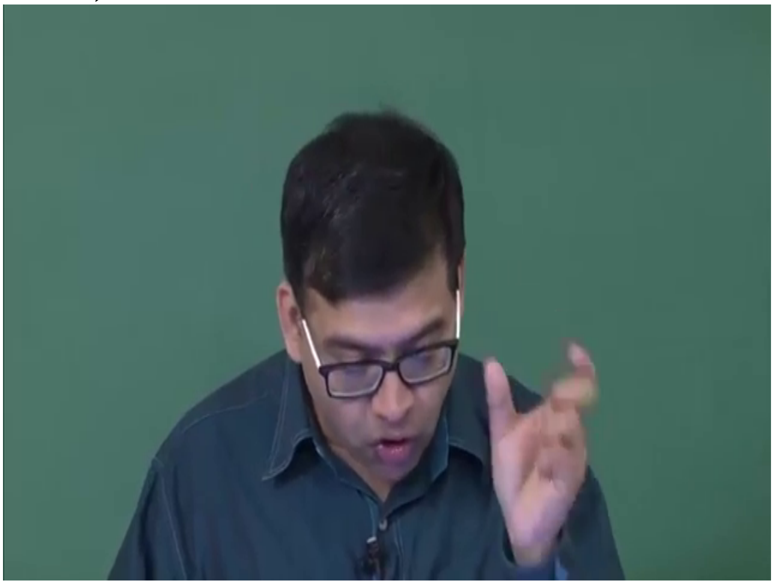
So what is a degree distribution? We define the degree distribution by this polynomial which has the property that gamma 1 is basically 1, so these are the fraction of nodes with degree i.

(Refer Slide Time 35:59)



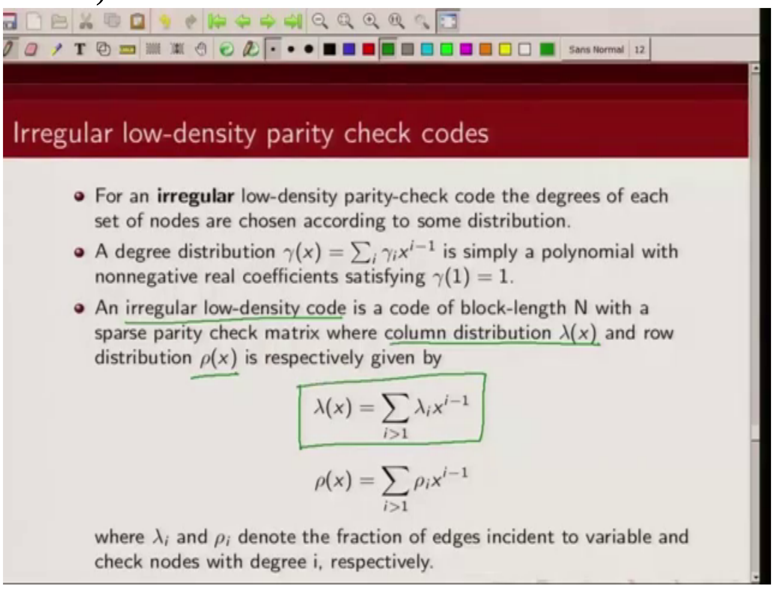
Now an irregular L D P C code we have to specify 2 degree distribution, one is the column degree distribution; other is the row

(Refer Slide Time 36:11)



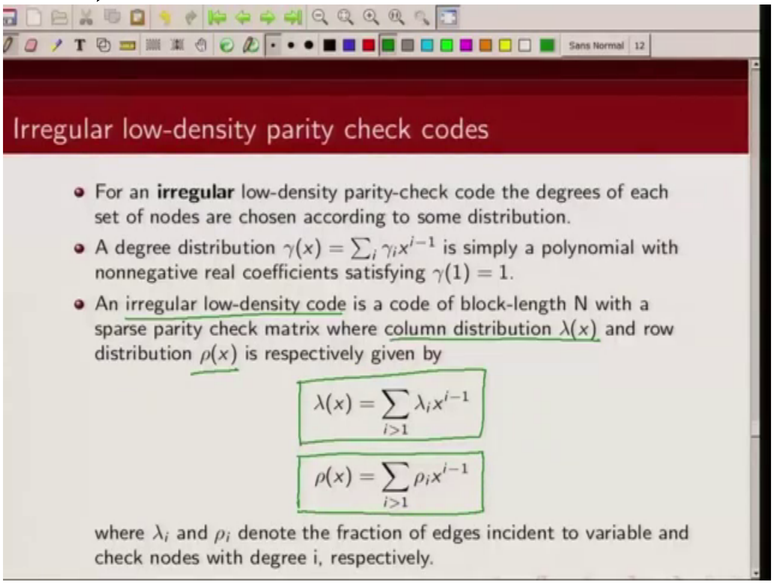
degree distribution. So the column degree distribution we are denoting by lambda x and the row degree distribution we are denoting by row x. So this is my column degree

(Refer Slide Time 36:26)



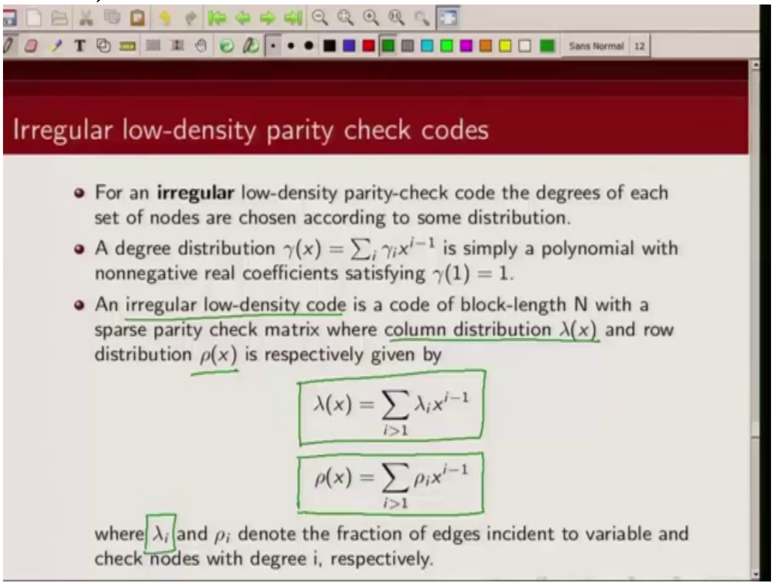
distribution, this is my row degree distribution

(Refer Slide Time 36:30)



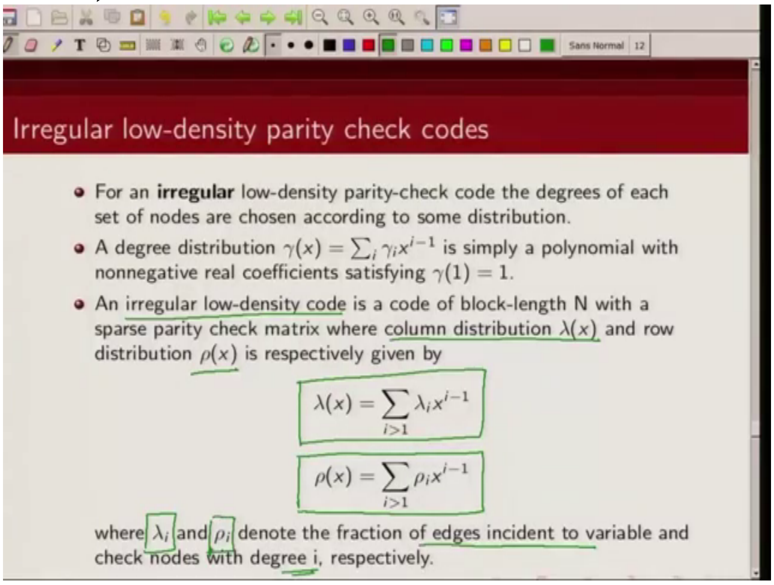
where lambda i

(Refer Slide Time 36:33)



is the fraction of edges incident on the variable node which has degree i and row i

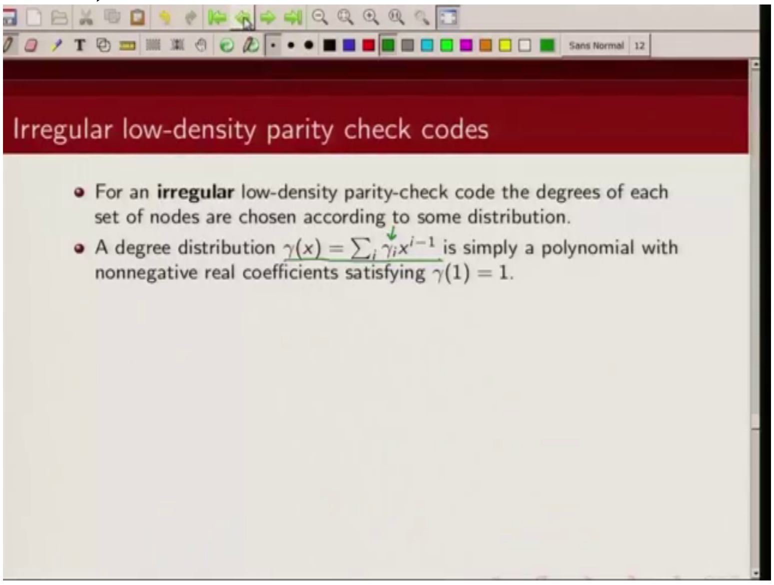
(Refer Slide Time 36:43)



is fraction of edges incident to the check node with degree i.

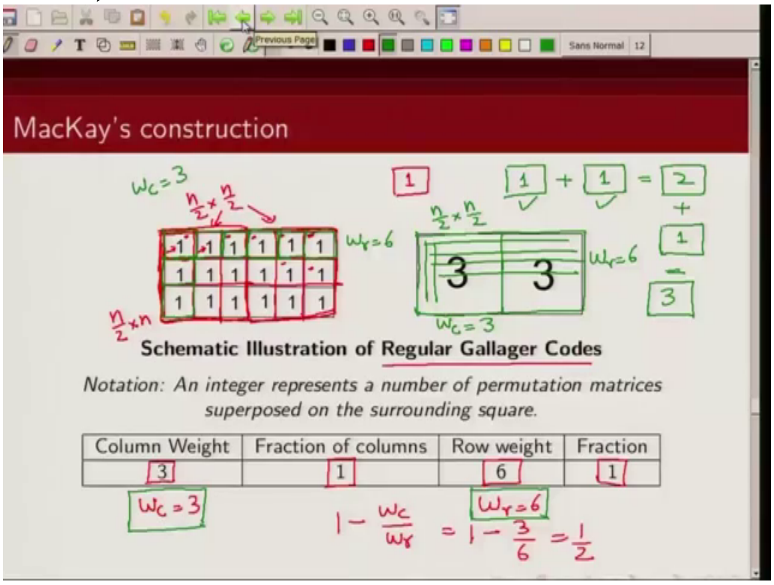
So let's take an example to illustrate this. Let's just see if we use

(Refer Slide Time 36:56)



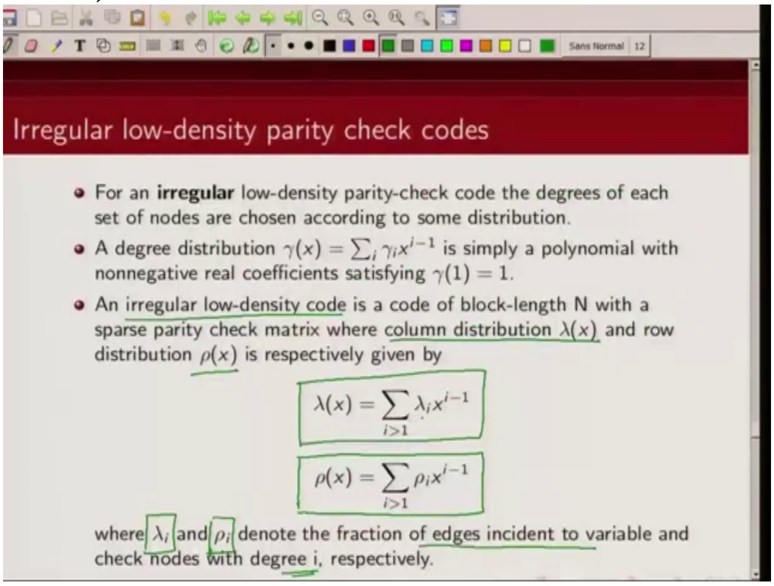
same degree notation to represent

(Refer Slide Time 36:59)



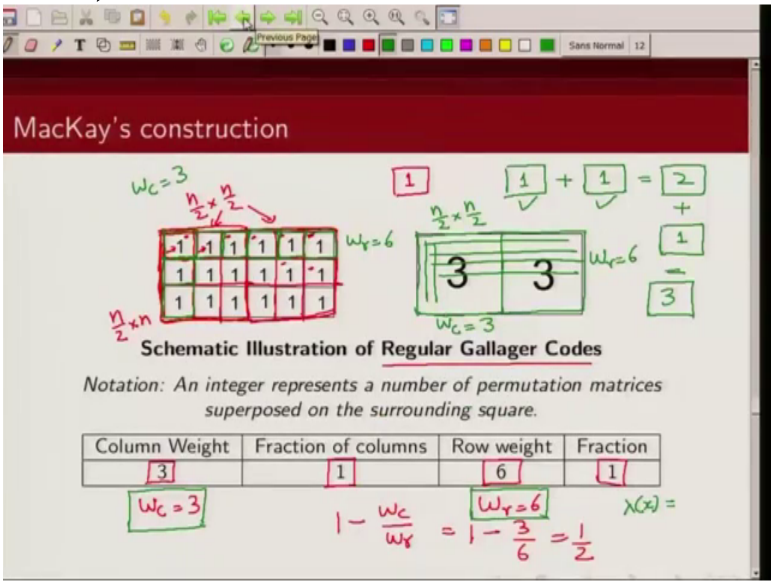
this. So the column row representation is defined by lambda x. Now here all

(Refer Slide Time 37:05)



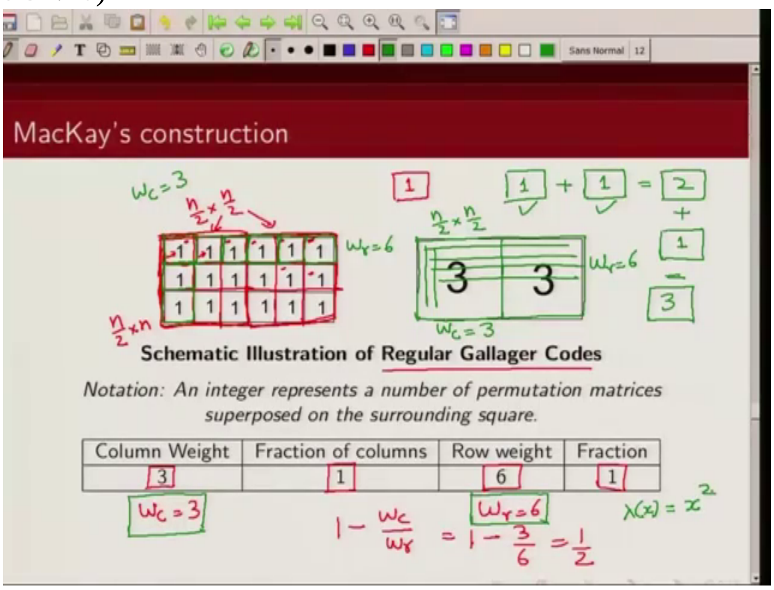
the columns for the regular L D P C code, all the columns have weight 1, so then

(Refer Slide Time 37:12)



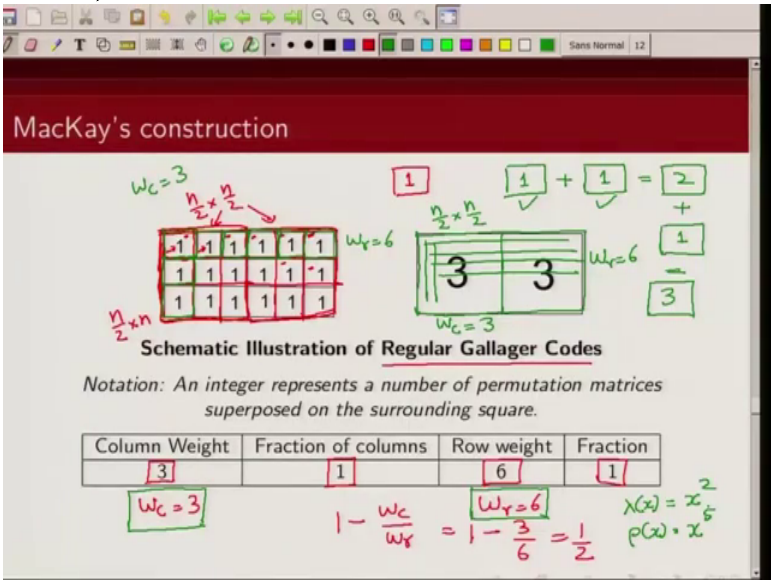
lambda here is 1 and we will define x raised to power i minus 1 and i degree here is 3. So degree distribution

(Refer Slide Time 37:20)



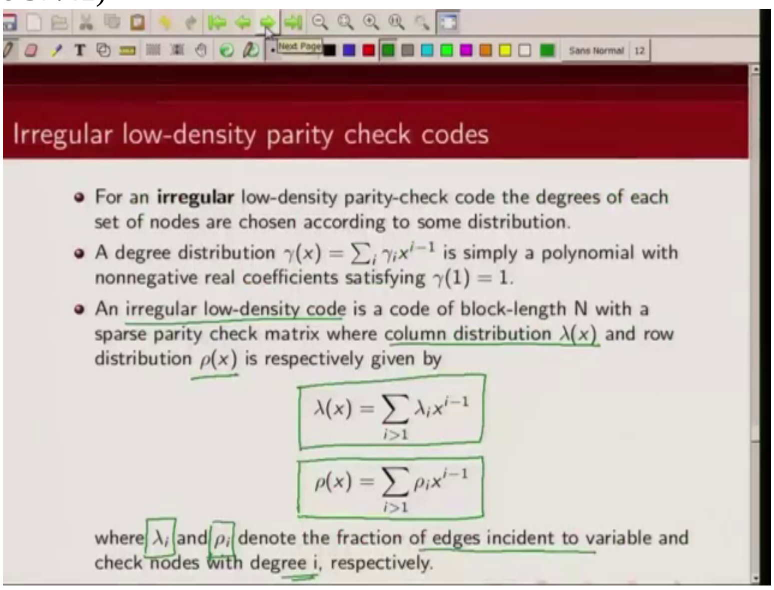
for this column for this regular L D P C code will be x square. Similarly the row distribution here is because all the nodes have row weight 6 so this will be x raised to power 6 minus 1 x 5.

(Refer Slide Time 37:36)



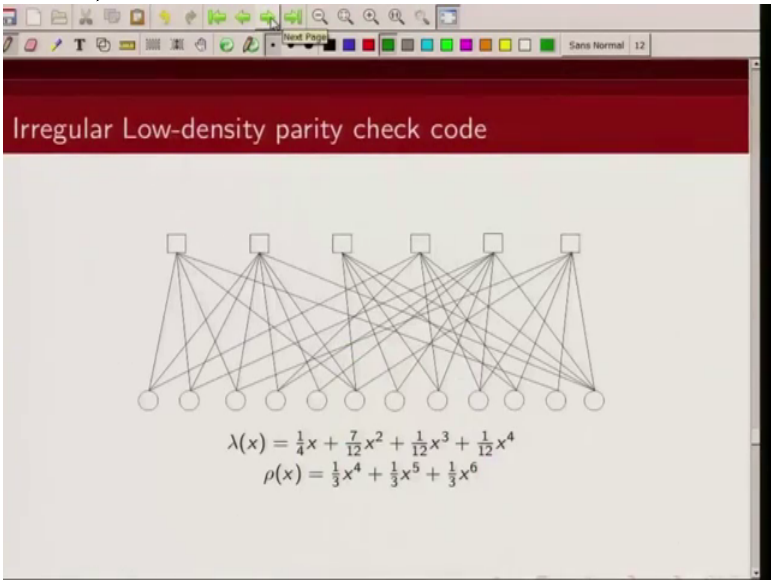
So that's how we are writing the degree distribution. So again

(Refer Slide Time 37:41)



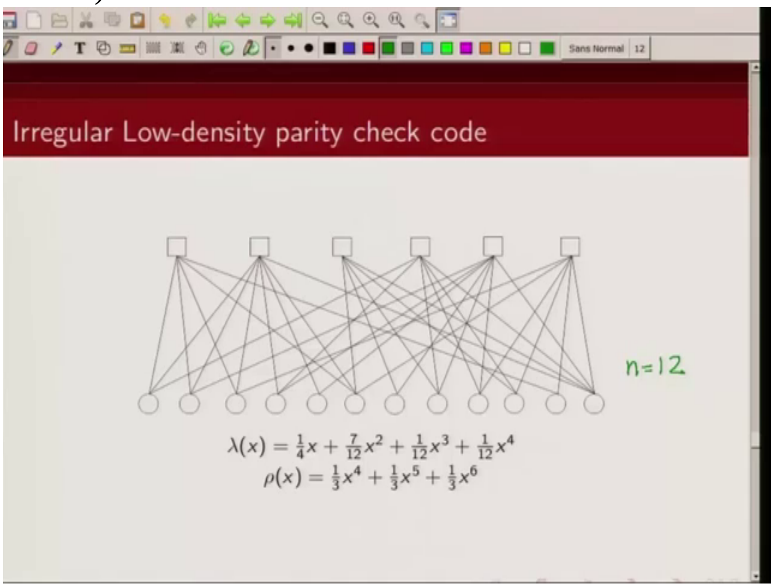
have a look at the degree distribution. This is a fraction of nodes with degree i and these are fraction of nodes with degree i. This is a row degree distribution; this is the column degree distribution.

(Refer Slide Time 37:57)



Let us take this example. So we have 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, so n is 12, Ok.

(Refer Slide Time 38:11)

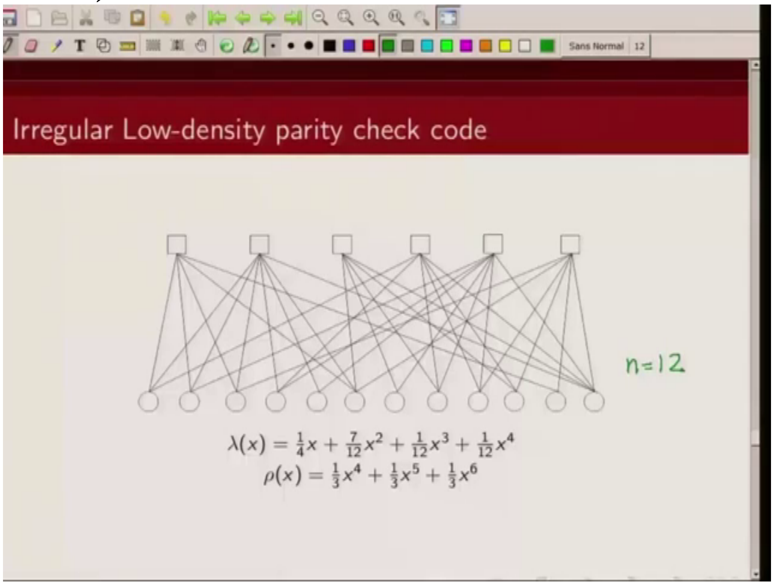


n is 12. Now what is column degree distribution? So this is basically each node participating in how

(Refer Slide Time 38:21)

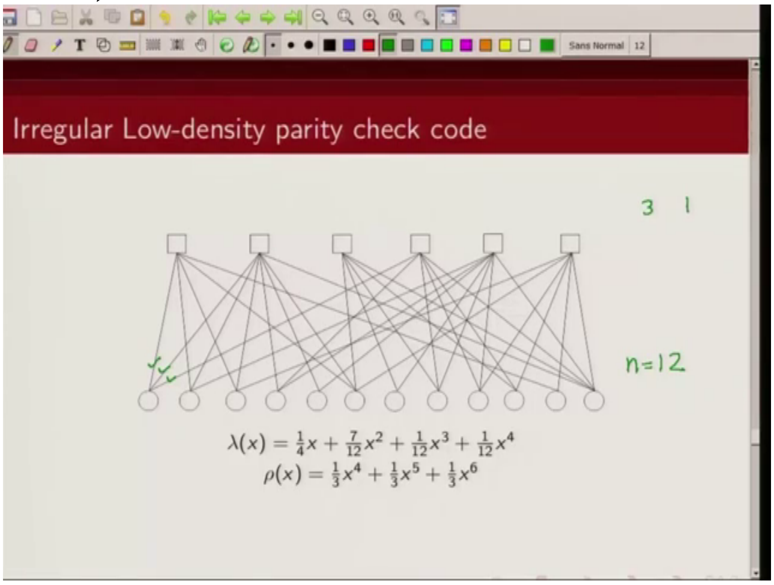
many parity check equations. Let us look at

(Refer Slide Time 38:25)



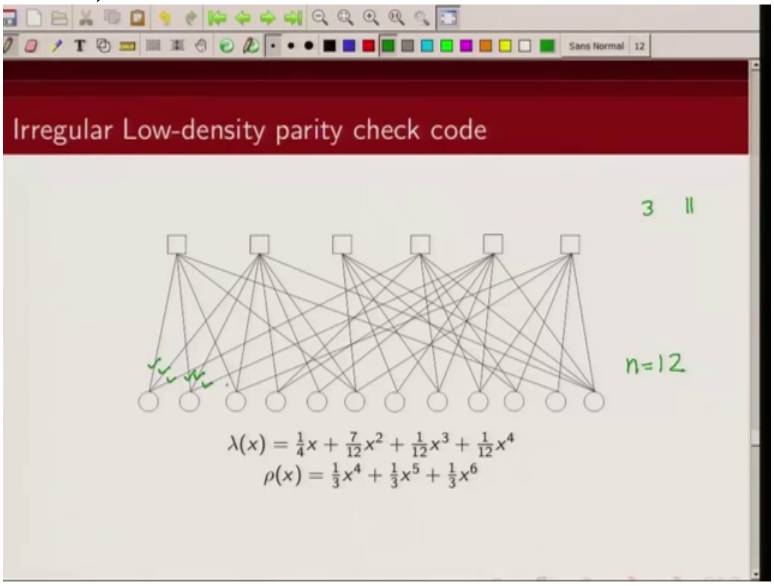
this node, how many parity check equations it is participating in? 1, 2, 3 so let's just, so there is, this is participating

(Refer Slide Time 38:38)



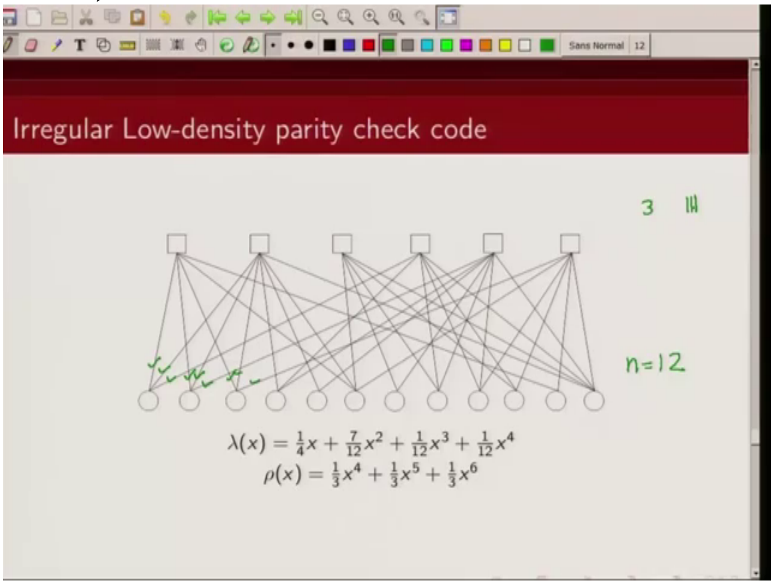
in 3 parity check equations, what about this? It is participating in 1, 2, 3. So it is also participating in three. This is participating

(Refer Slide Time 38:48)



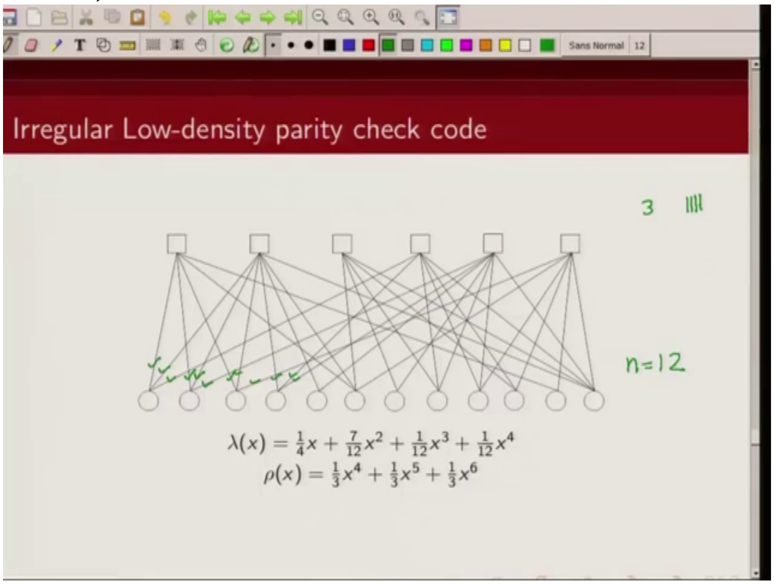
in 1, 2, 3. This is participating in three.

(Refer Slide Time 38:53)



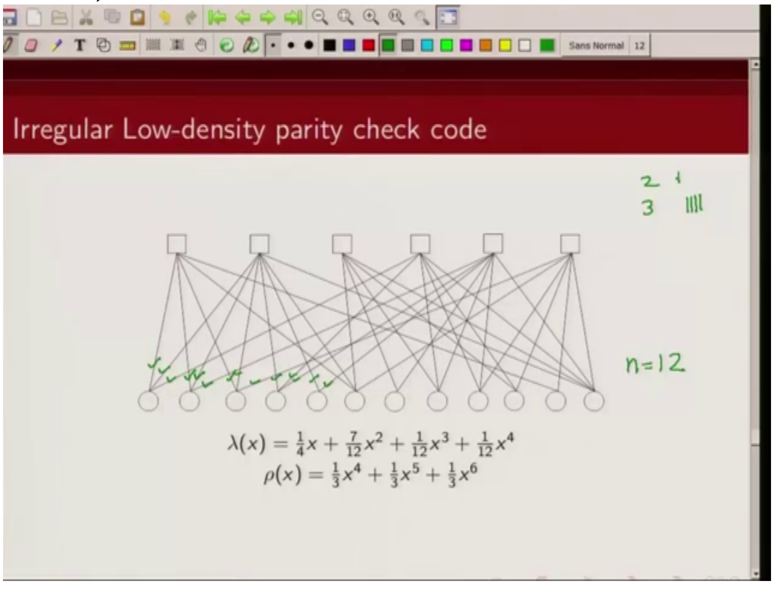
This is participating in 1, 2, 3. This is participating in two,

(Refer Slide Time 39:00)



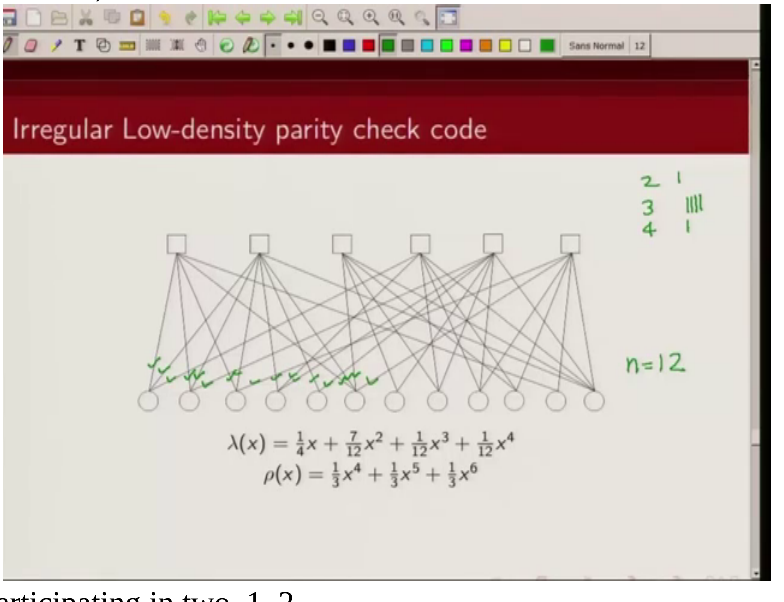
1, 2. So there is one node which is participating in 2.

(Refer Slide Time 39:05)



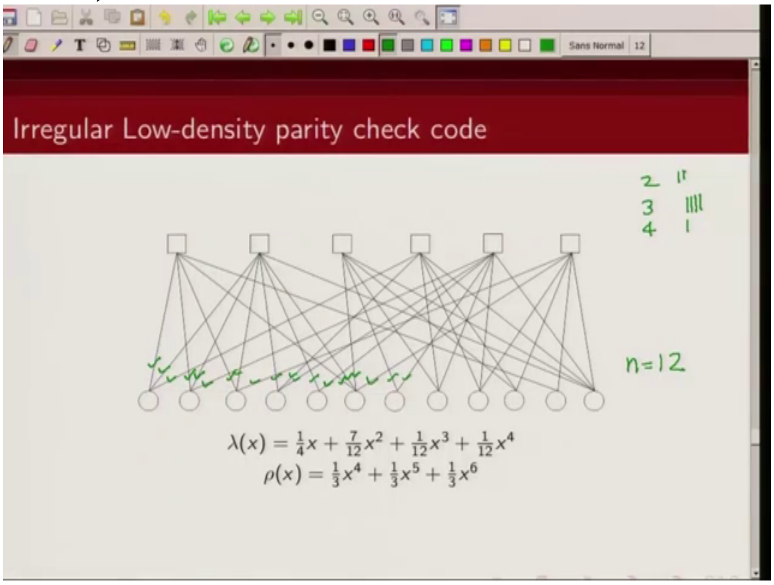
What about this? This is participating in four, 1, 2, 3, 4. So this is one node which is participating

(Refer Slide Time 39:13)



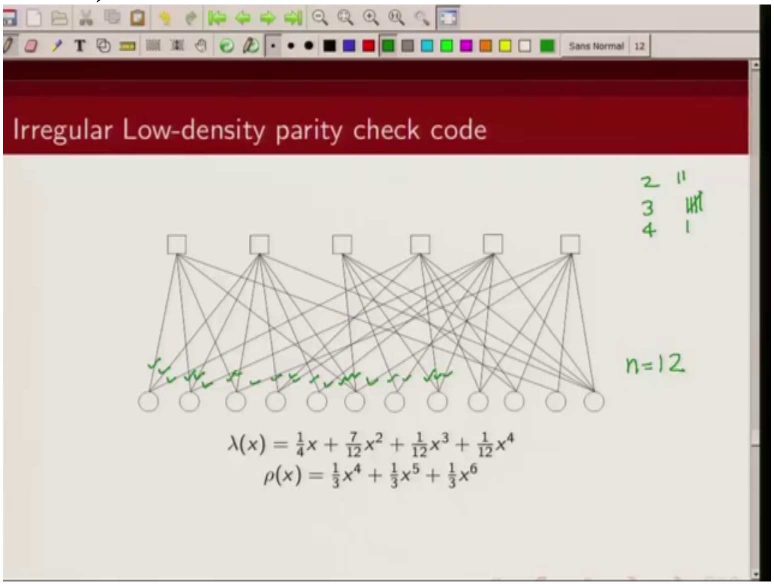
in four. This is participating in two, 1, 2.

(Refer Slide Time 39:19)



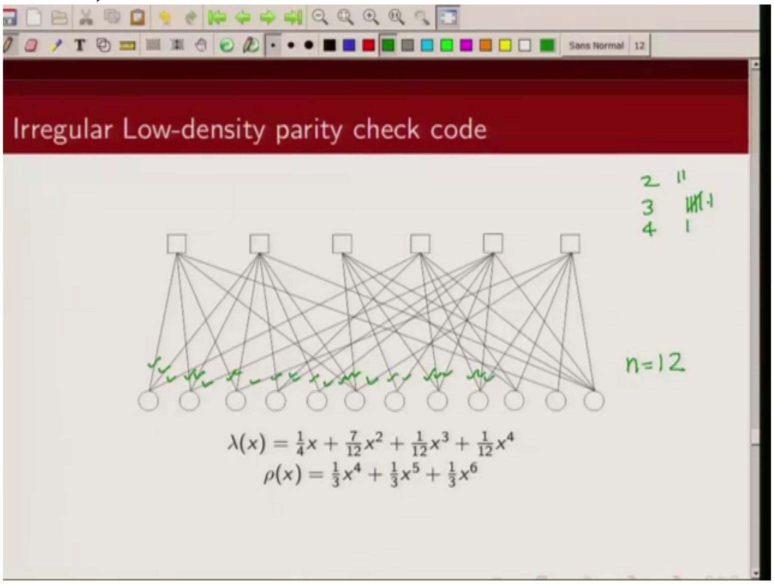
This is participating in three, 1, 2, 3.

(Refer Slide Time 39:24)



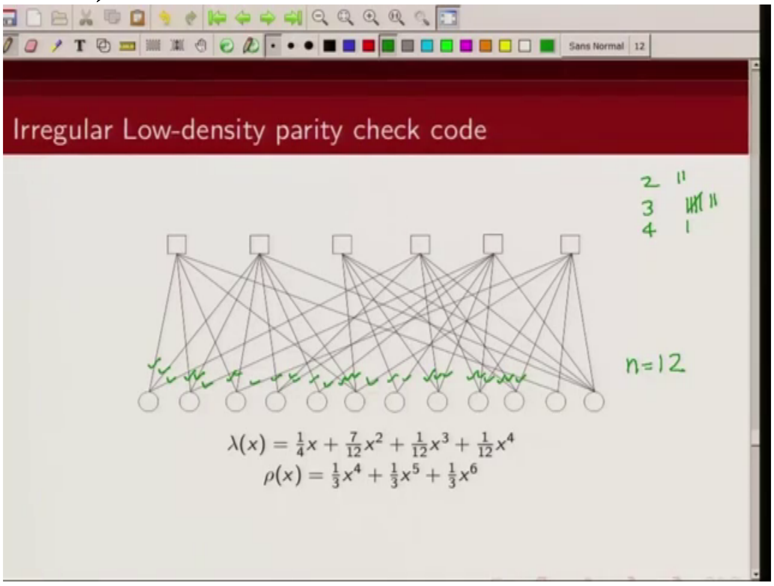
So this is participating in three, 1, 2, 3.

(Refer Slide Time 39:28)



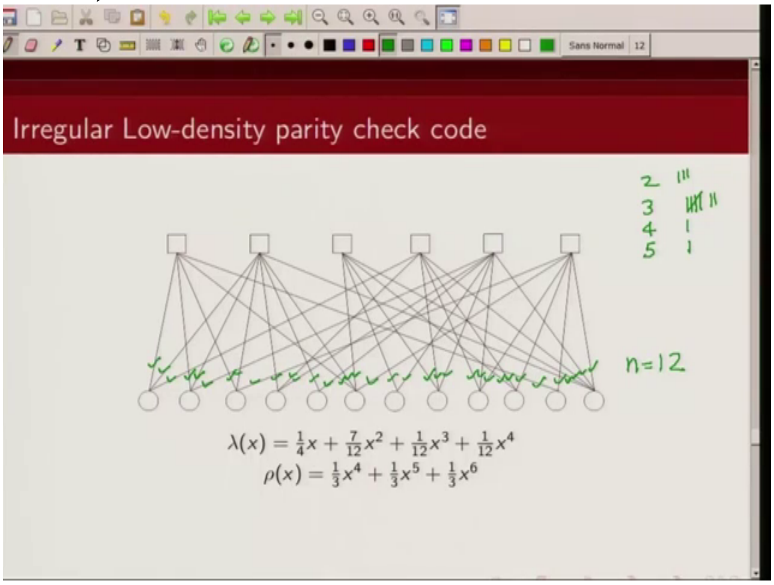
It is participating in three, 1, 2, 3.

(Refer Slide Time 39:32)



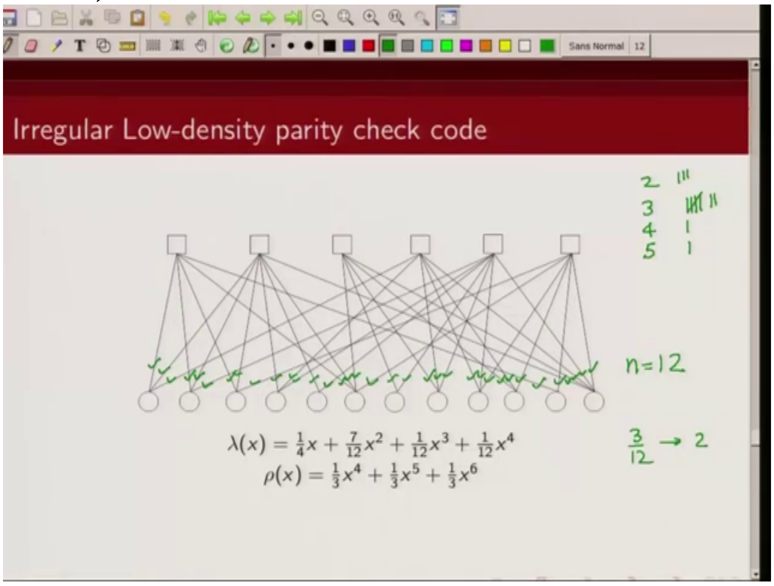
It is participating in two, 1, 2. And this is participating in 1, 2, 3, 4, 5,

(Refer Slide Time 39:41)



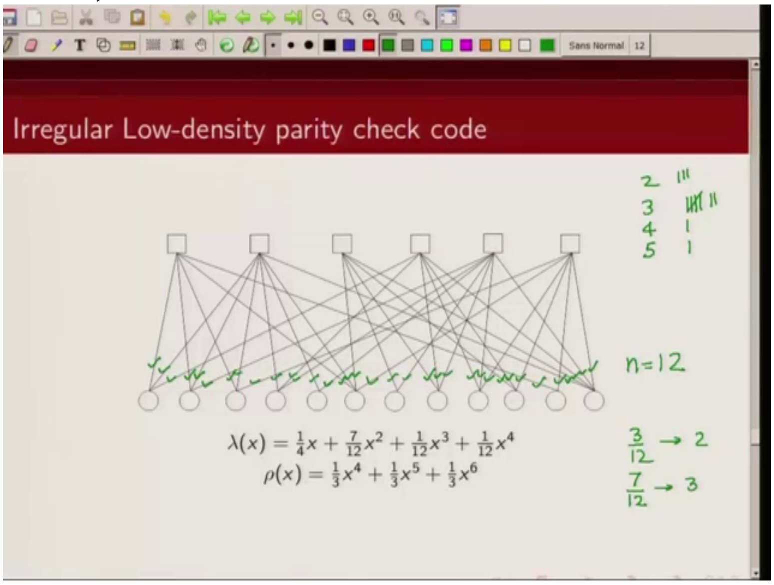
Ok. So then how many nodes are participating in two parity check constraints, that is three. So what's the fraction? That is 3 by 12. They are participating in two parity check constraints.

(Refer Slide Time 39:57)



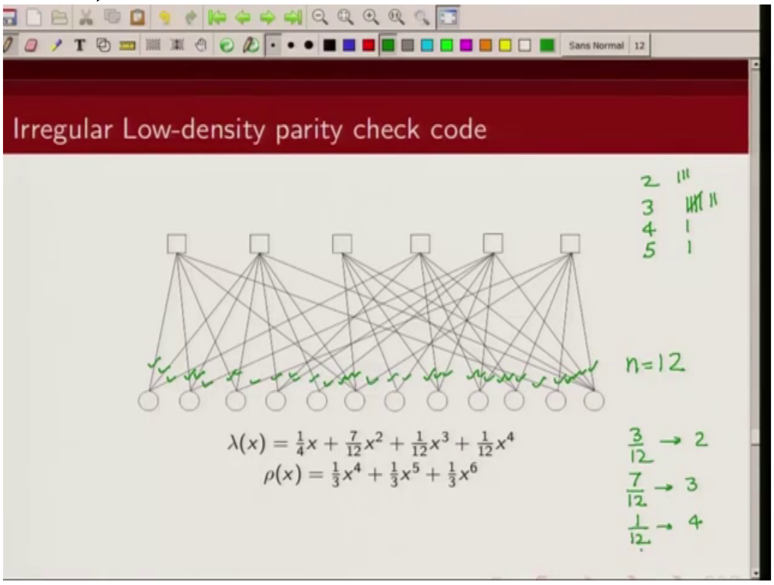
There are 7 of them which are participating in 3 parity check constraints. So

(Refer Slide Time 40:04)



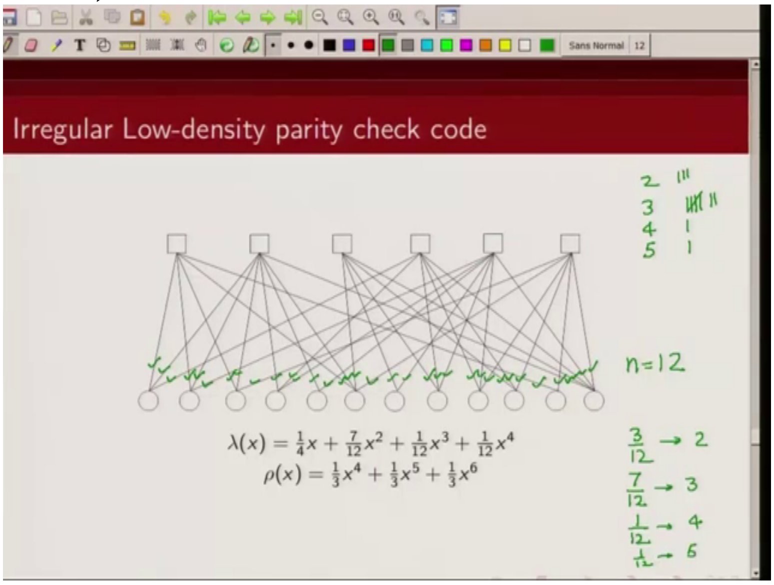
fraction of them is 7 by 12. Then this is 1 by 12 and this is participating in four. And then

(Refer Slide Time 40:13)



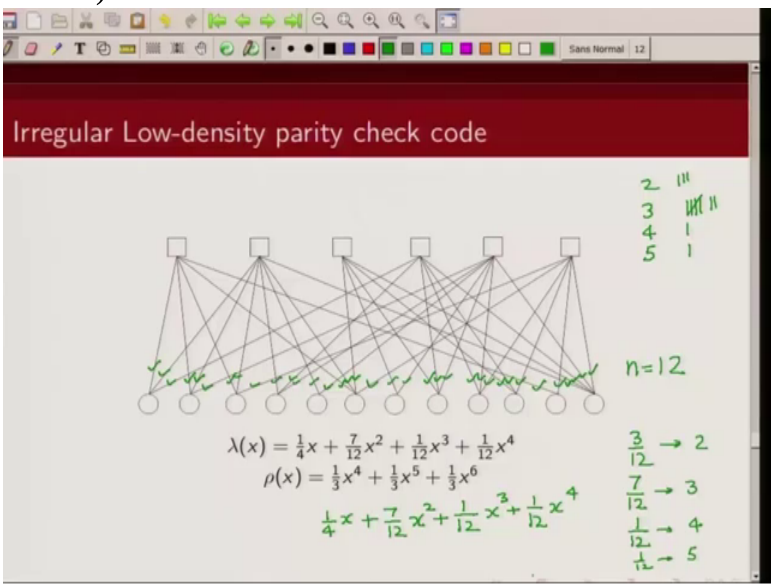
1 by 12, it is participating in five.

(Refer Slide Time 40:15)



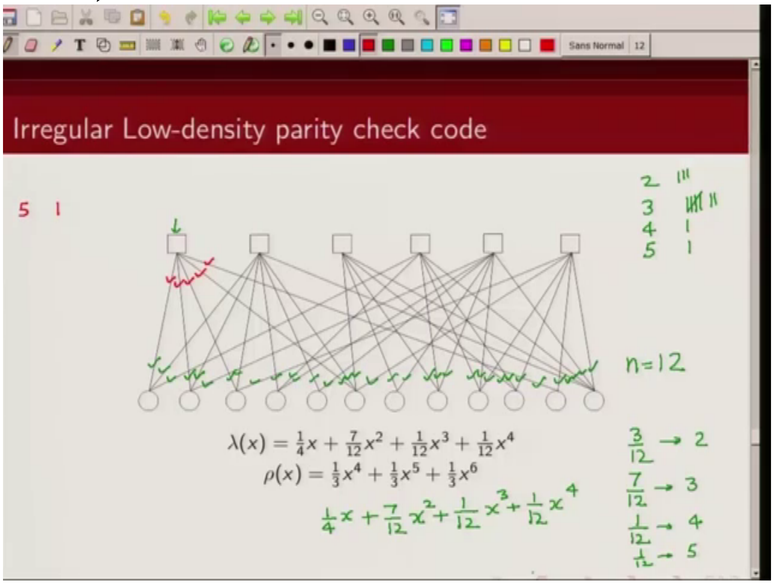
So then what is the column distribution? So this, this fraction is 1 by 4 x raised to power 2 minus 1 that is x, plus 7 by 12 x raised to power 3 minus 1 that's x square plus 1 by 12 x raised to power 4 minus 1, that is three plus 1 by 12 x raised to power 4. So this is the column

(Refer Slide Time 40:49)



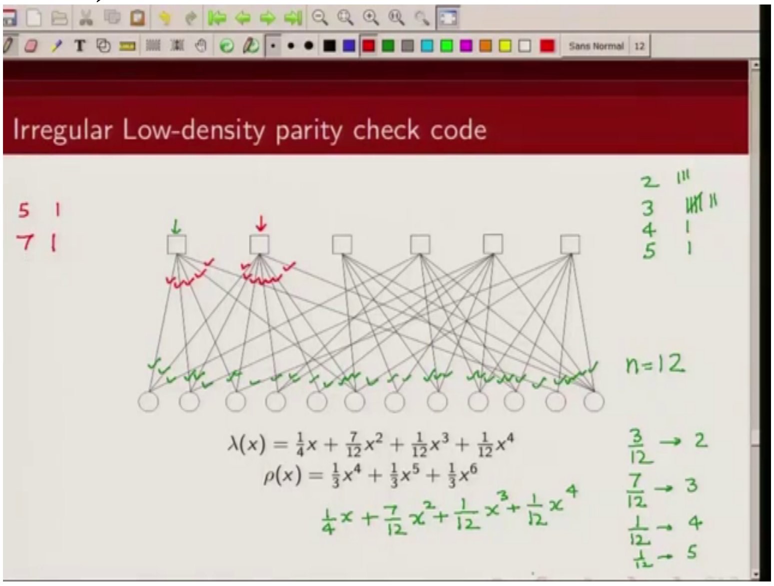
degree distribution for an L D P C code which is described by this Tanner graph. Similarly we can find the row distribution. Let's look at parity check constraints. So let's look at this parity check constraint. So here, 1, 2, 3, 4, 5 so there are 5 nodes which are participating in this.

(Refer Slide Time 41:15)



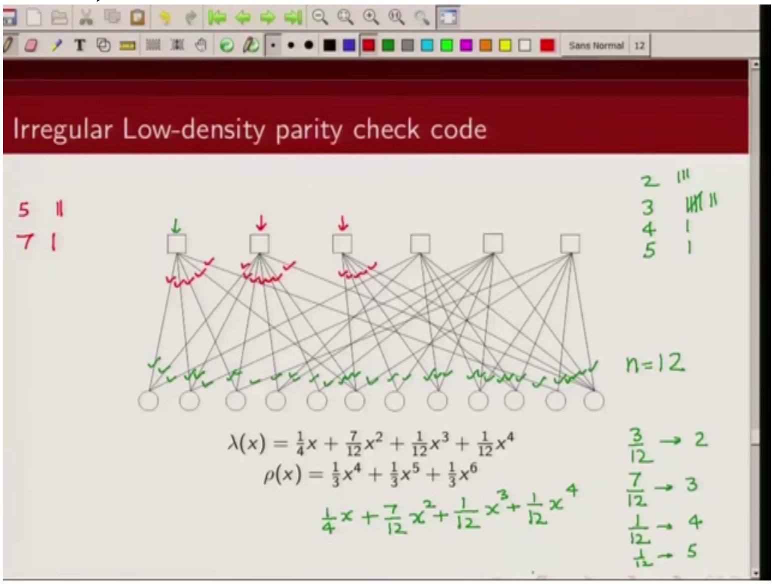
What about this one? 1, 2, 3, 4, 5, 6, 7 so there are 7 bits which are participating in this.

(Refer Slide Time 41:24)



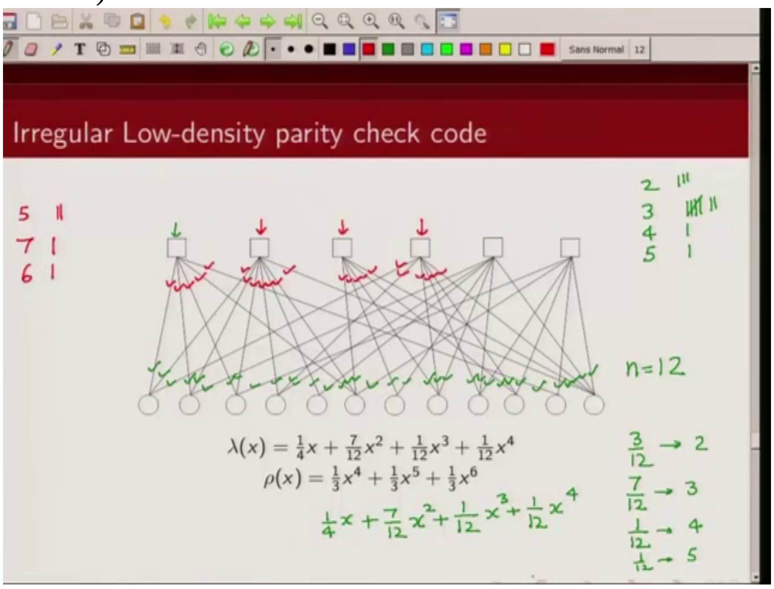
This one, 1, 2, 3, 4, 5 so there is 5 of them.

(Refer Slide Time 41:31)



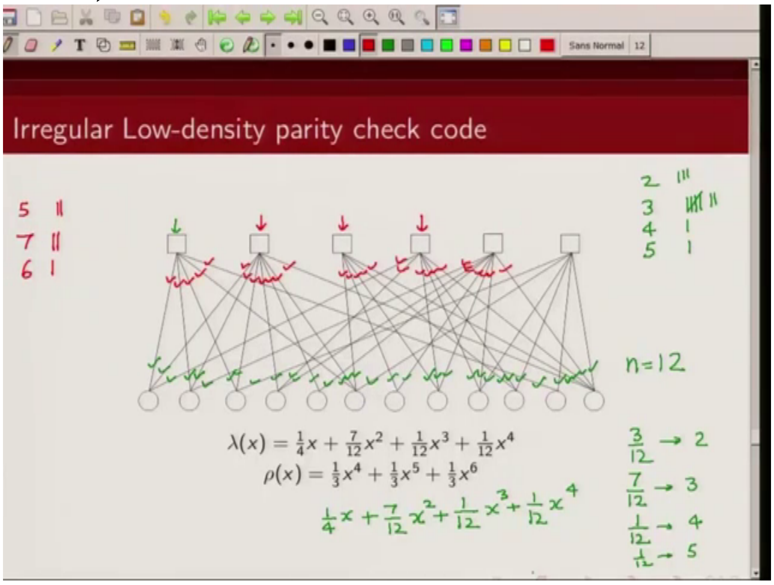
Then here 1, 2, 3, 4, 5, 6 so there is

(Refer Slide Time 41:37)



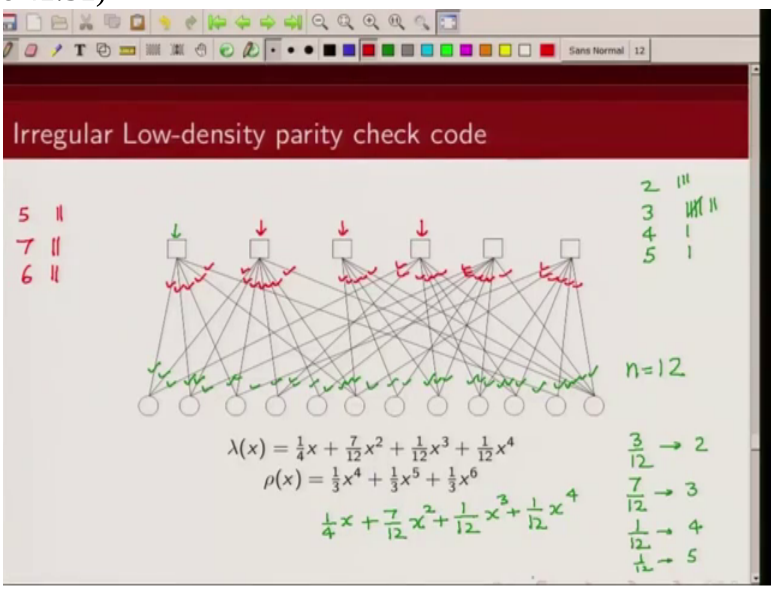
6. This one is 1, 2, 3, 4, 5, 6, 7

(Refer Slide Time 41:45)



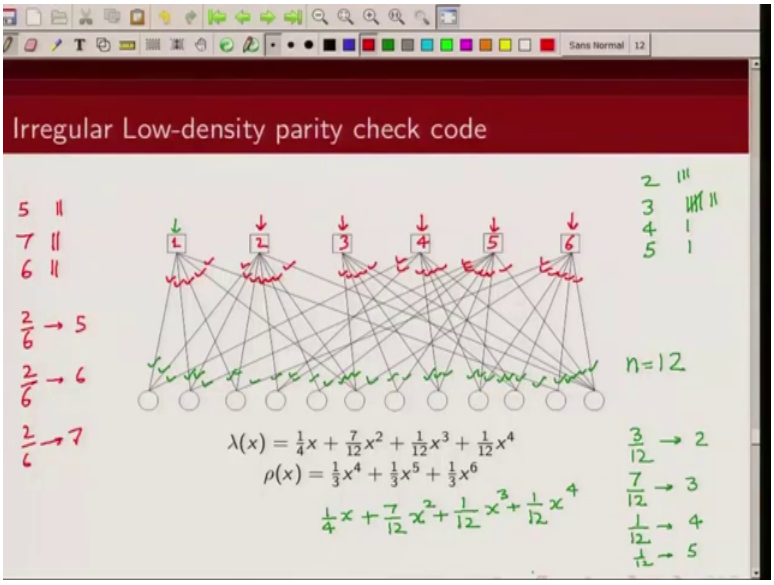
and this one is 1, 2, 3, 4, 5, 6. So

(Refer Slide Time 41:51)



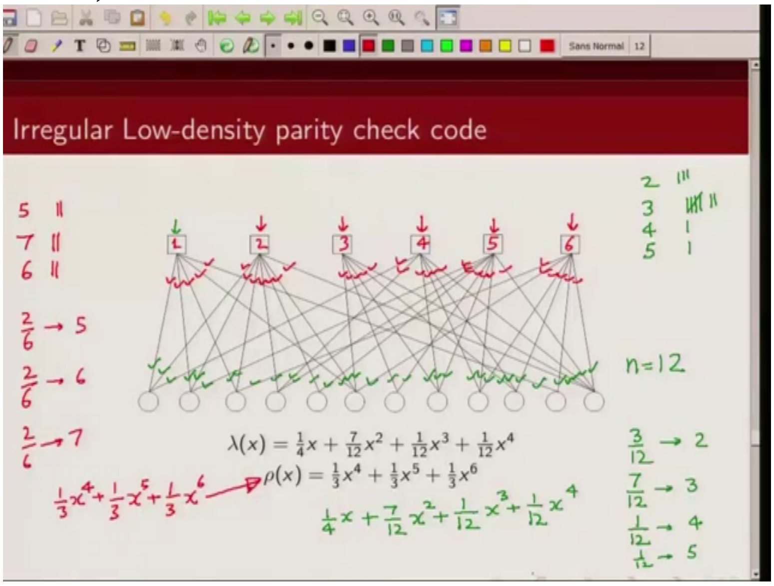
we can see out of these and what's the total number of parity check equations? These are six, 1, 2, 3, 4, 5 and 6. So then fraction of parity check equations where five bits are participating is 2 by 6. And same is the ratio for 6 and 7. So then we can write the row distribution as

(Refer Slide Time 42:23)



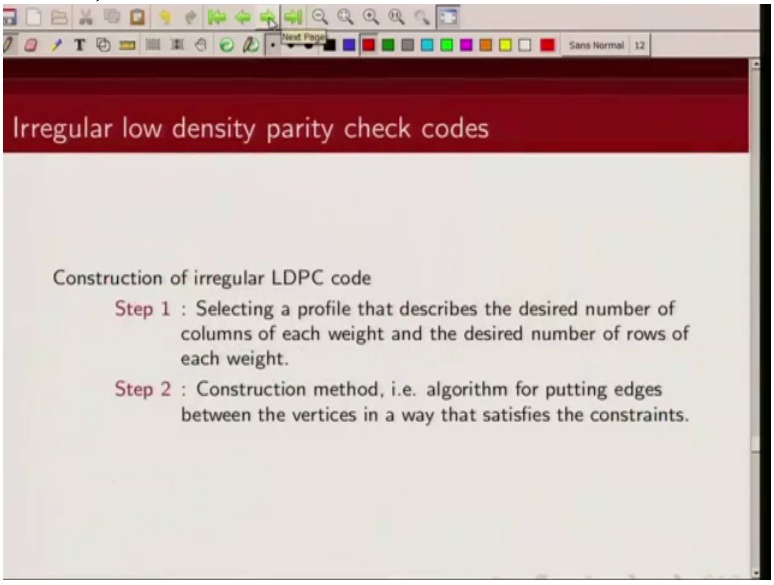
1 by 3 x raised to power 5 minus 1, that's 4 plus 1 by 3 x raised to power 6 minus 1 that is 5 plus 1 by 3 x to power 7 minus 1 that's 6. And that's precisely what I have written

(Refer Slide Time 42:43)



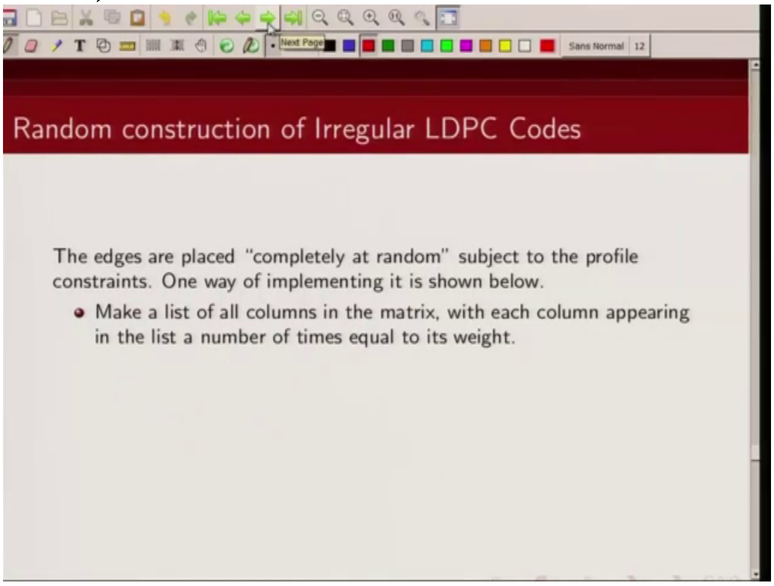
here, Ok. So to describe an irregular L D P C code we need to describe these 2 degree distribution, namely the column degree distribution and the row degree distribution.

(Refer Slide Time 43:03)



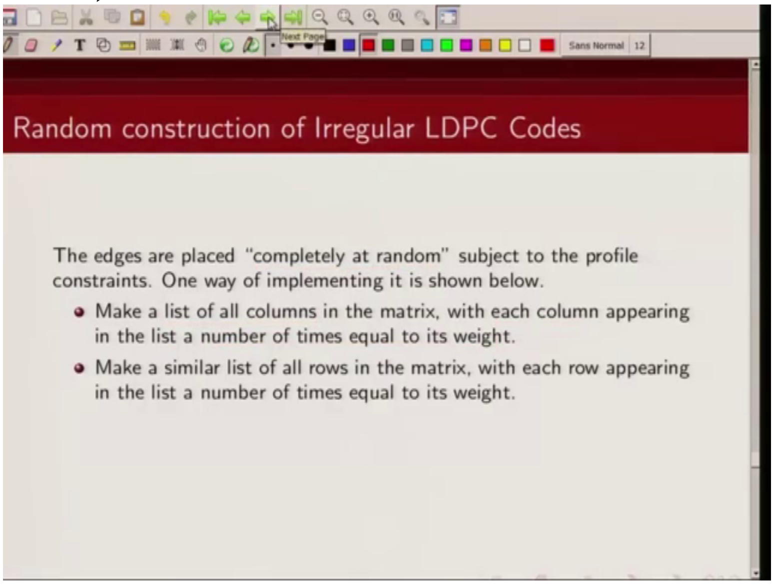
So we can use a random algorithm to construct L D P C code so we first select the profile that describes the number of rows of each weight and desired number of columns of each weight and we need to put the mechanism of putting edges between the vertices in such a way that this constraint is satisfied, that so many number of rows should have so many weight and so many number of columns should have so many weight.

(Refer Slide Time 43:37)



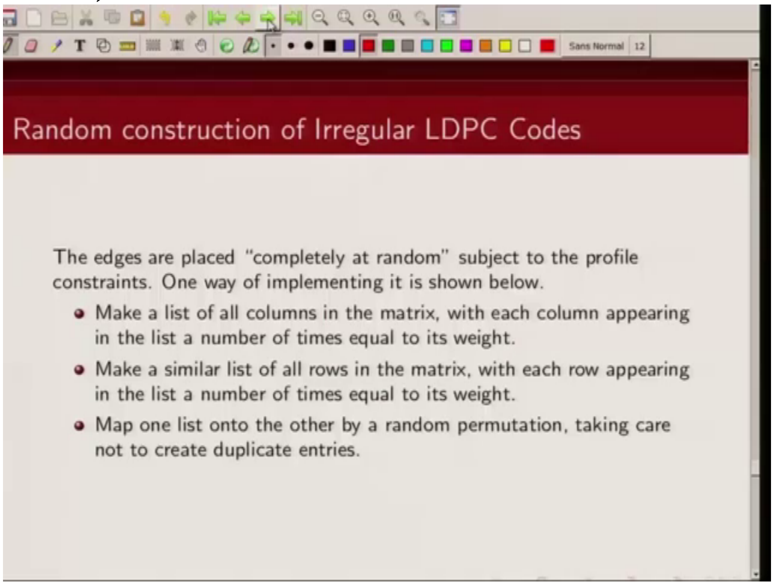
So we can place edges at random subject to profile constraint and one way of constructing it is as follows. So make a list of all columns in a matrix with each column appearing in the list number of times which is equal to the, it's weight. So if, let's say there are 5 columns which are weight 3. So then you will repeat each of these 5 columns 3 times in this matrix.

(Refer Slide Time 44:08)



Similarly you make a list of all rows in the matrix such that each row is repeated equal to the its weight and then what you do is you map

(Refer Slide Time 44:22)

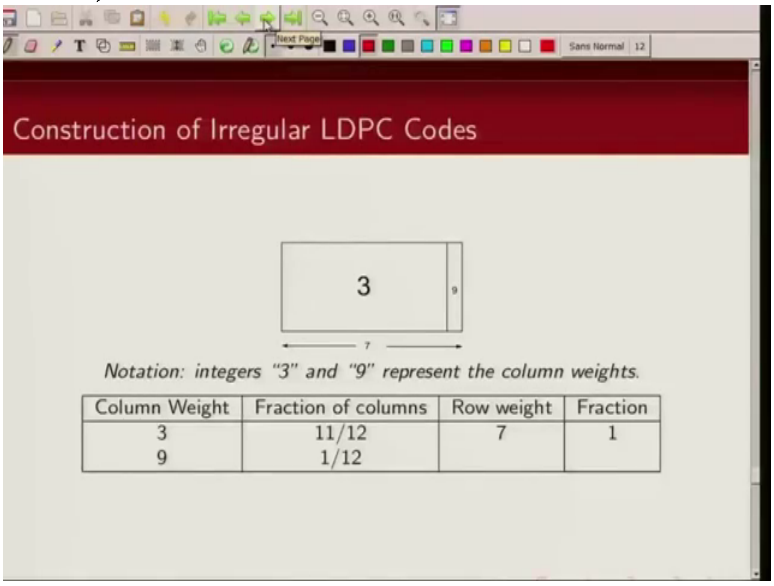


one list which is the list of columns into the other list which is the list of rows by random permutation making sure that there is no duplication. You just create a link from the list of columns to the list of rows and this way you can construct an irregular L D P C code which will satisfy the degree distribution

(Refer Slide Time 44:49)

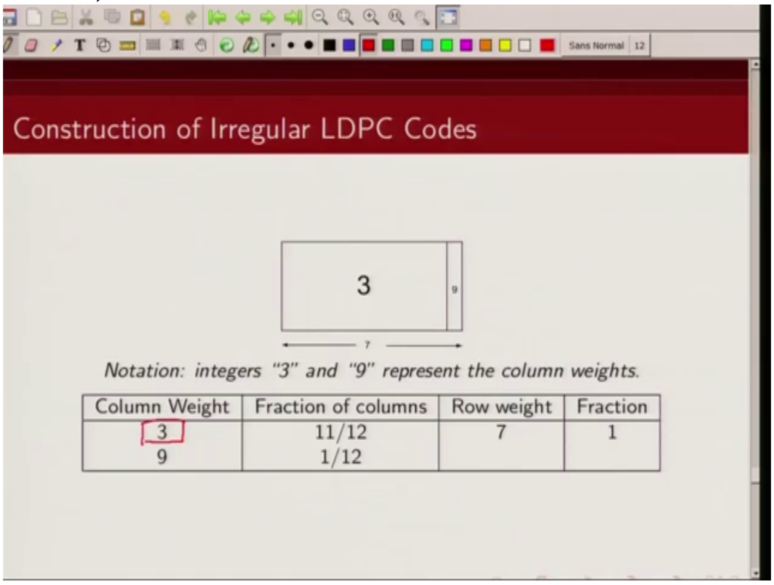
profile. We can also use random permutation matrices to generate our irregular L D P C code; again we are going for very simple construction. So we will

(Refer Slide Time 45:06)



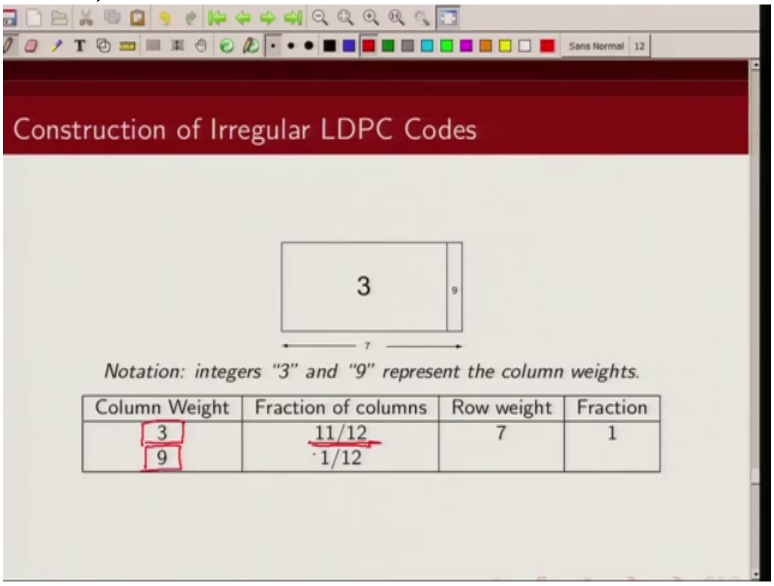
go with this Mackay's construction, so let's say we would like to design an L D P C code which has column weight

(Refer Slide Time 45:18)



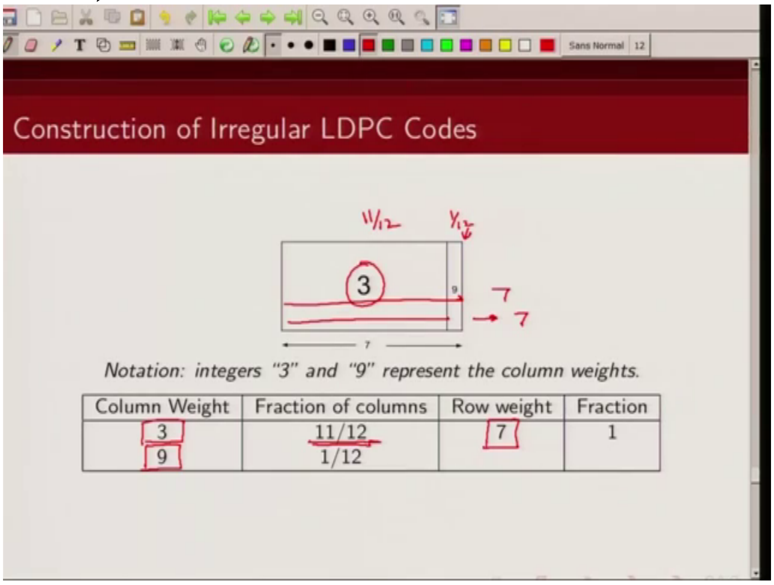
such that eleven twelfth of the columns have column weight 3 and one twelfth of the columns has

(Refer Slide Time 45:32)



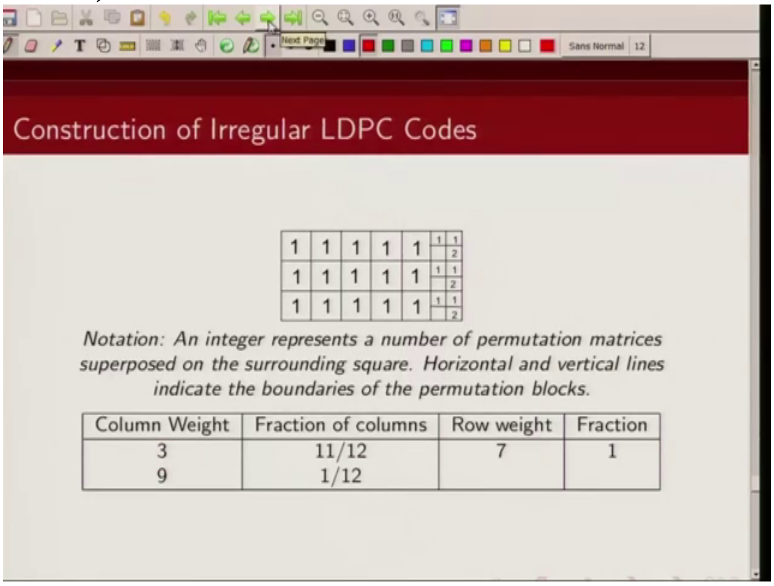
column weight 9.So we are interested in, and we are interested in row weight of 7. So we want each of these rows to weight 7, each of these rows should have weight 7 and we want that eleven twelfth of the column should have weight 3 and one twelfth of the column should have weight

(Refer Slide Time 45:55)



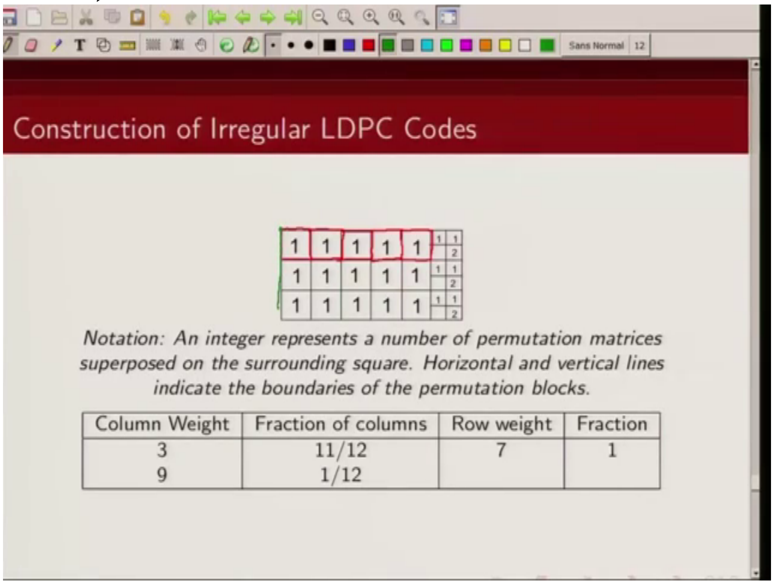
9. So how can we construct this

(Refer Slide Time 45:59)



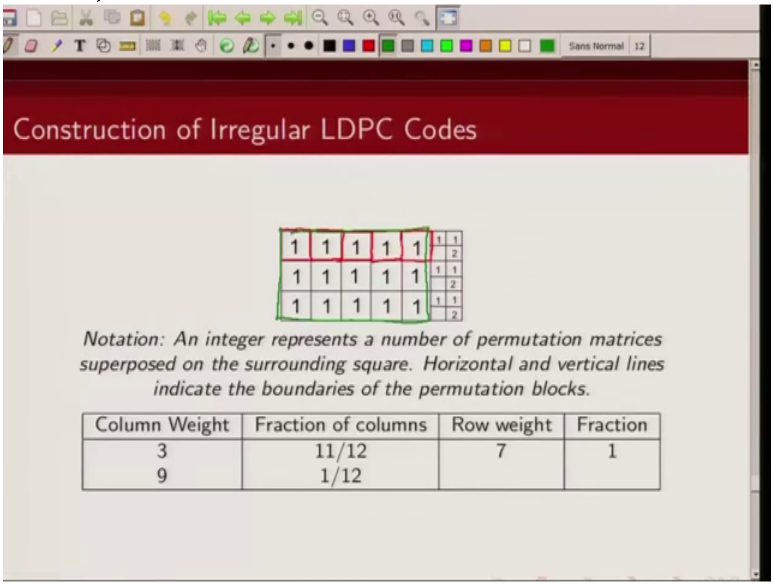
using the, construction using random permutation matrices that we have studied. So again I am splitting up this into, so these are my, this is my rate half code so n by 2 cross n. Each of them are n by 6 by n by 6 matrices. These are n by 6 by n by 6 matrices. Now note that up to

(Refer Slide Time 46:29)



this I am ensuring that

(Refer Slide Time 46:33)

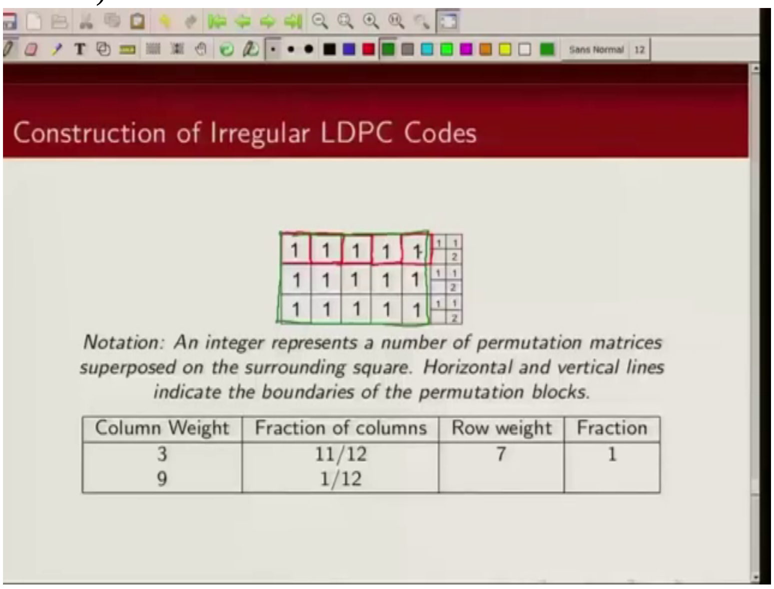


each column has weight 3 and each row has weight 5. Now I need

(Refer Slide Time 46:43)

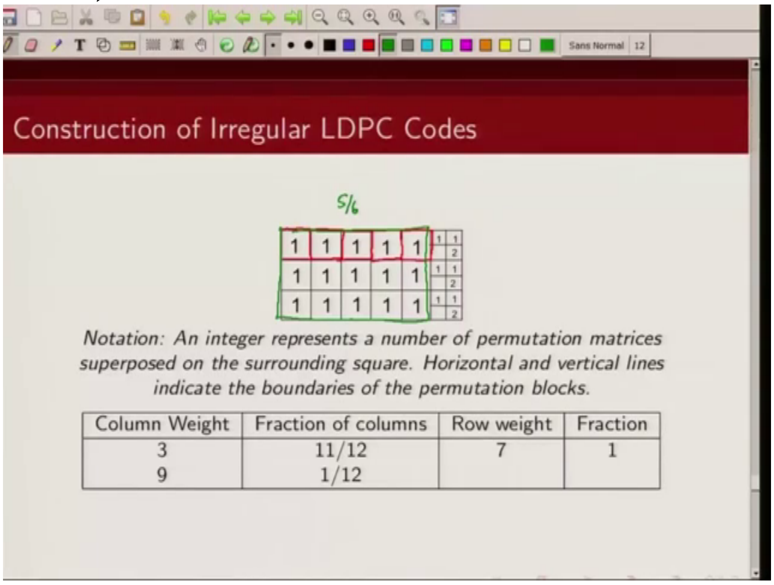
column weight of 11 by 12

(Refer Slide Time 46:46)



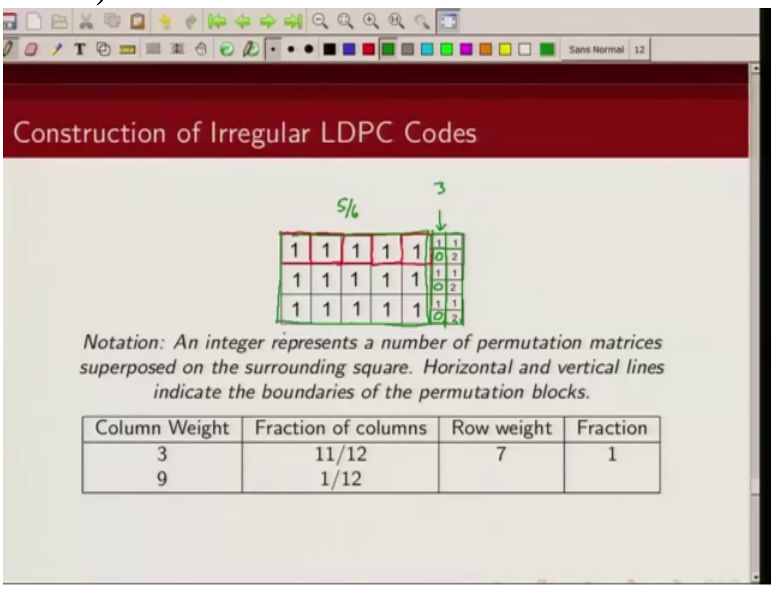
matrices to be 3. So this is already, I have got 5 by 6 columns;

(Refer Slide Time 46:54)



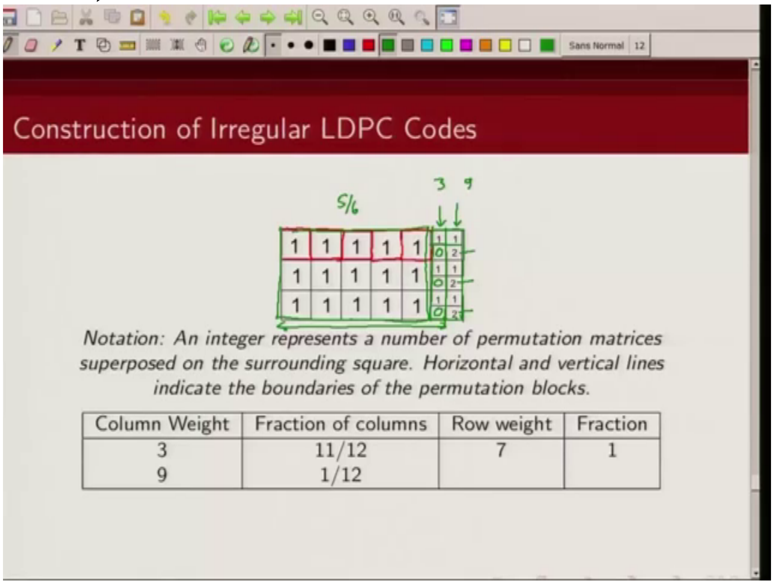
I have got column weight as 3. So then what I can do for the remaining 1 by 6 fraction of columns I split them into two and what I do is I have column weight here and I further split them into, so these are all zero matrices, so this you can see, this row will have weight 3,so this will ensure

(Refer Slide Time 47:19)



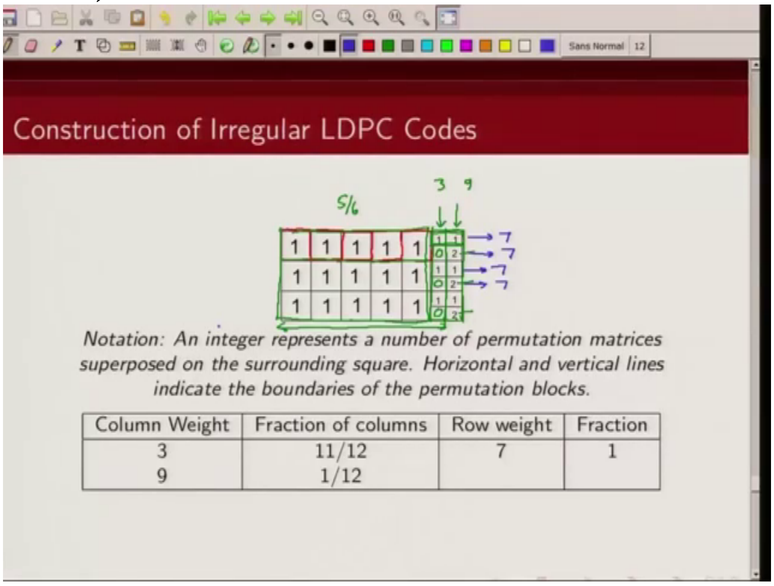
that 11 by 12 fraction of columns have weight 3 and here you can

(Refer Slide Time 47:39)



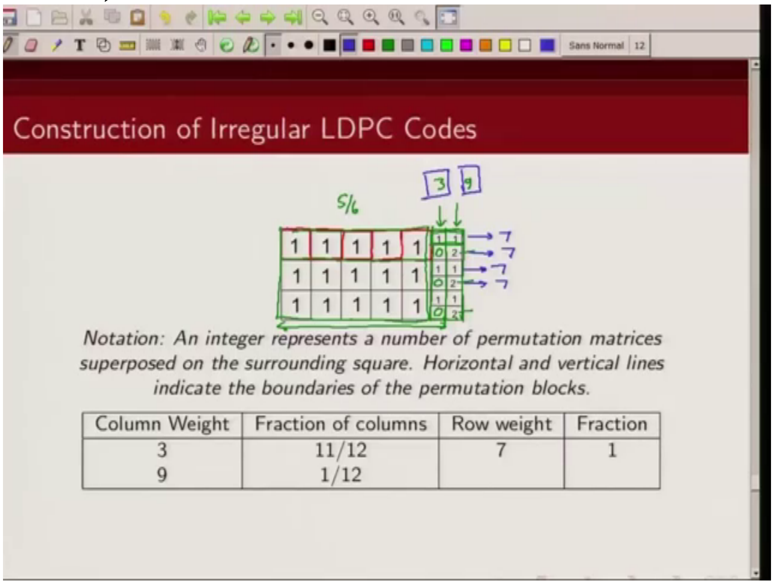
see this has weight 9, I have added this is weight 2, weight 2, weight 2 that's 6 plus 3, 9, so this column, this set of columns will have weight 9. And you can check I already had each row weight up to 5 and now I have, this row has weight 2, so this will be overall weight will be 7. This has weight 0 plus 2 so this has overall weight is 7. Here the overall row weight is 7. So I am ensuring that the way I split up these matrices, I am ensuring that

(Refer Slide Time 48:05)



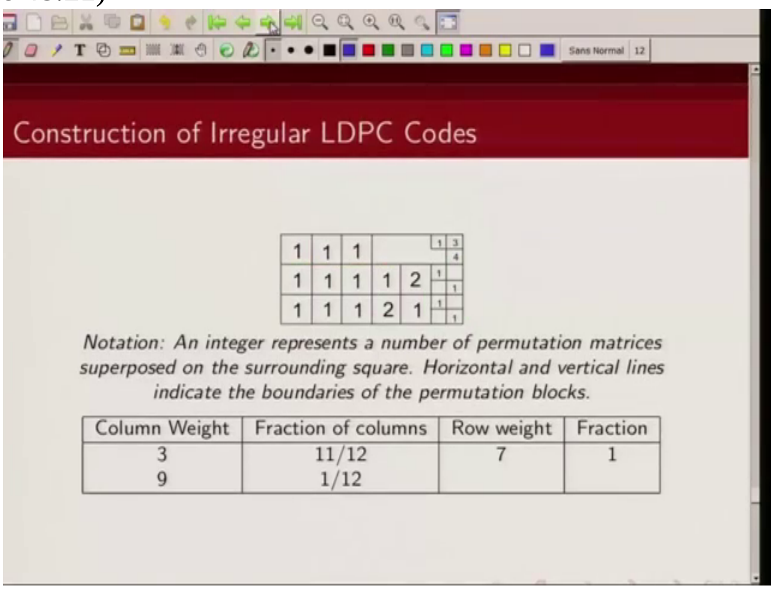
each row will have weight 7 where as 11 by 12 fraction of the columns have column weight 3, where as 1 by 12 fraction of the

(Refer Slide Time 48:18)



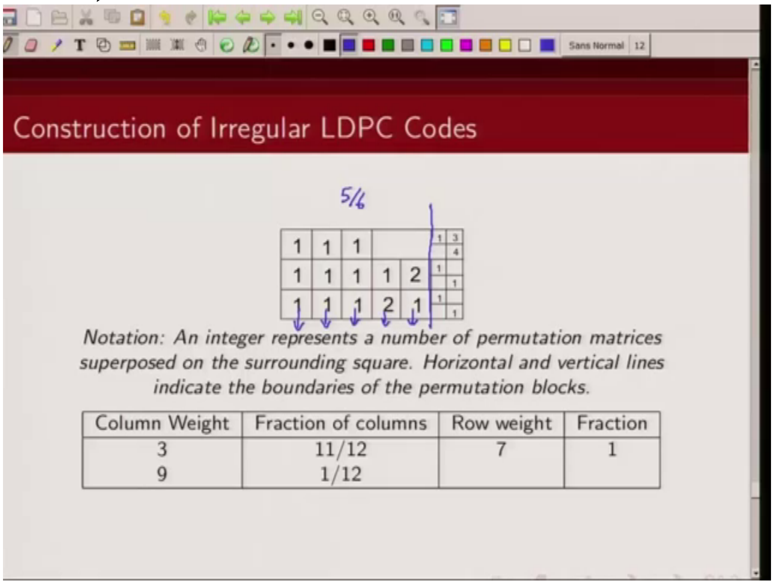
columns has weight 9.

(Refer Slide Time 48:21)



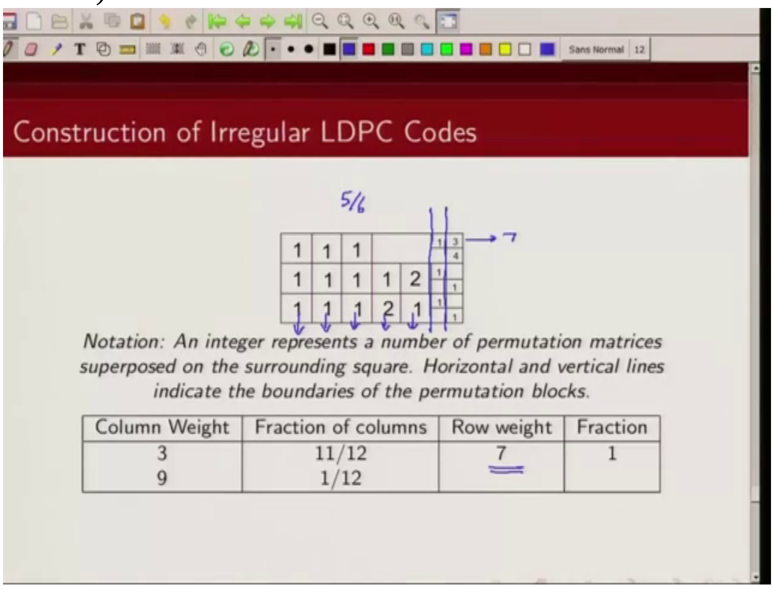
The same thing can be done in multiple ways. This is another construction. You can see here again I am ensuring that each of the columns have weight 3. So up to this point you have five sixth of the column

(Refer Slide Time 48:40)



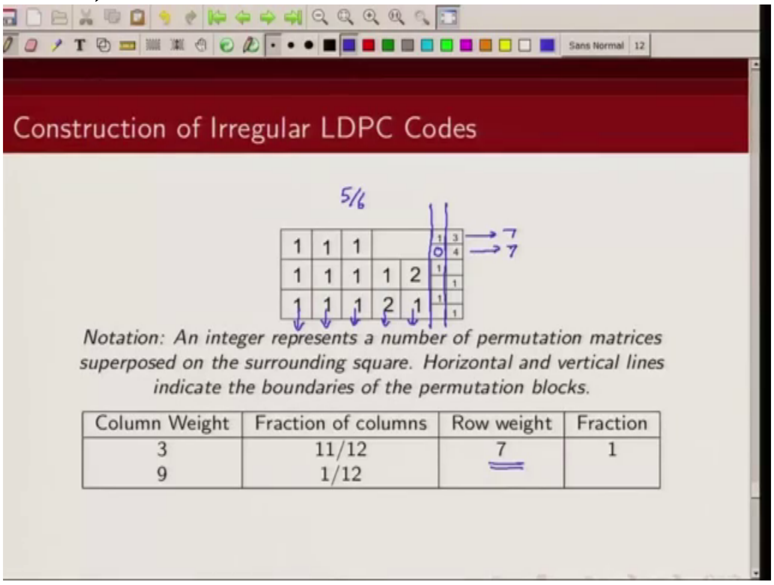
have weight 3. Now I need another one by twelfth column which has weight 3. So I can do that by placing 1s like this. Now I also have to ensure that row weight is 7. So how do I ensure? I have a 3 here, so I need a weight 4 here. So I do it by 1 and 3. So this is column weight,

(Refer Slide Time 49:02)



row weight 7. Here it is 3; this was 0, so this has to have weight of 4. Now again this,

(Refer Slide Time 49:11)

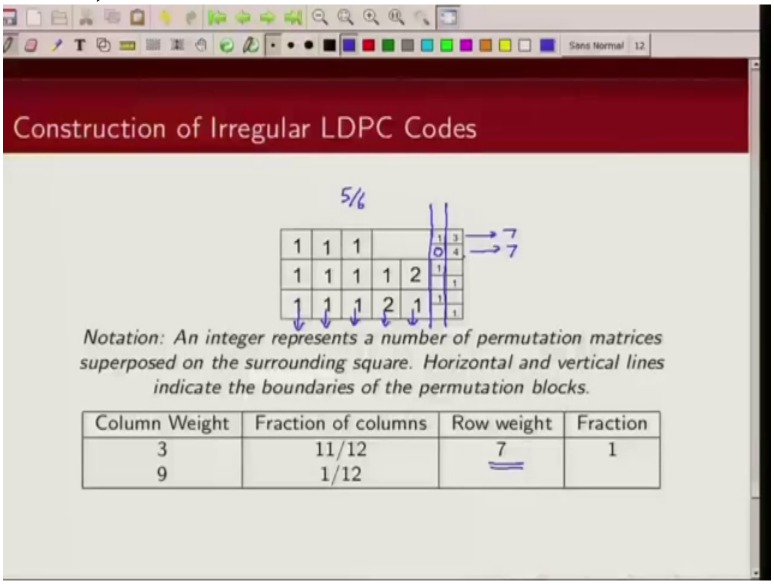


what are these 3, 4? These are again obtained by overlapping

(Refer Slide Time 49:16)

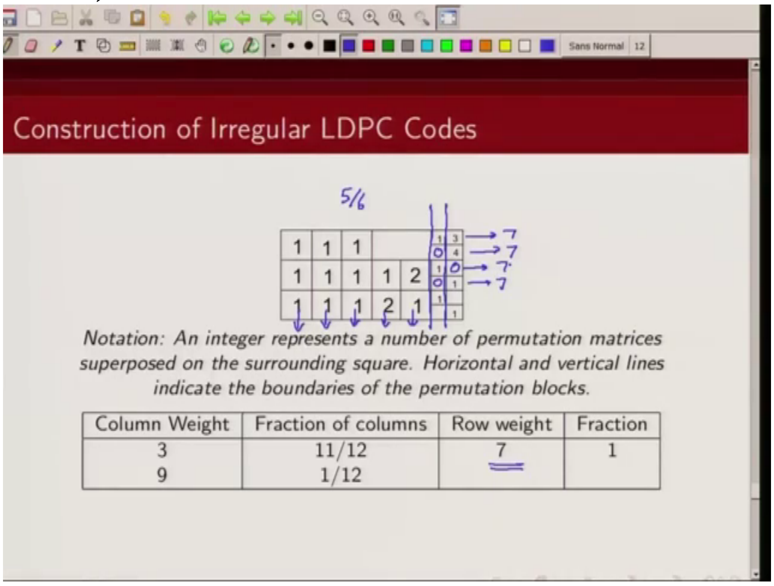
of random permutation matrices. And when we are overlapping, make sure that there is no overlaps of 1s, Ok.

(Refer Slide Time 49:25)



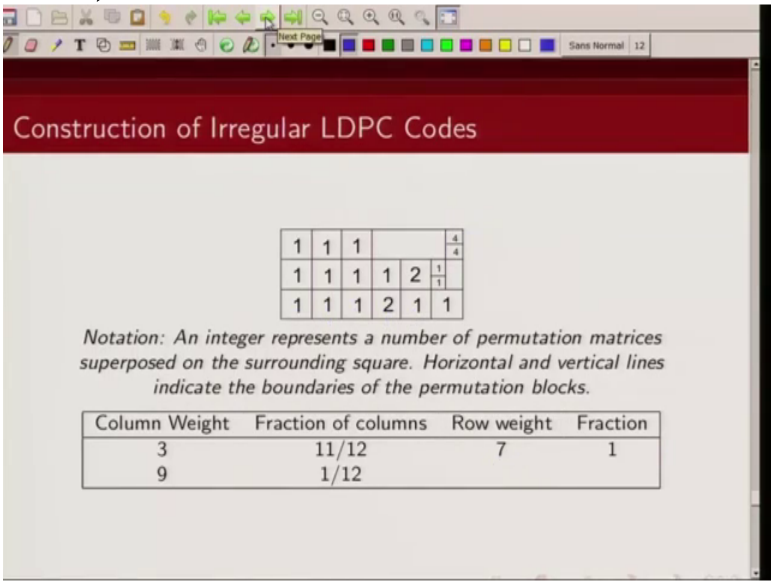
So you can verify that each row here, so 1, 2, 3, 4, 5, 6 so this is 7. This is weight 7. Again here this is weight 7.

(Refer Slide Time 49:37)



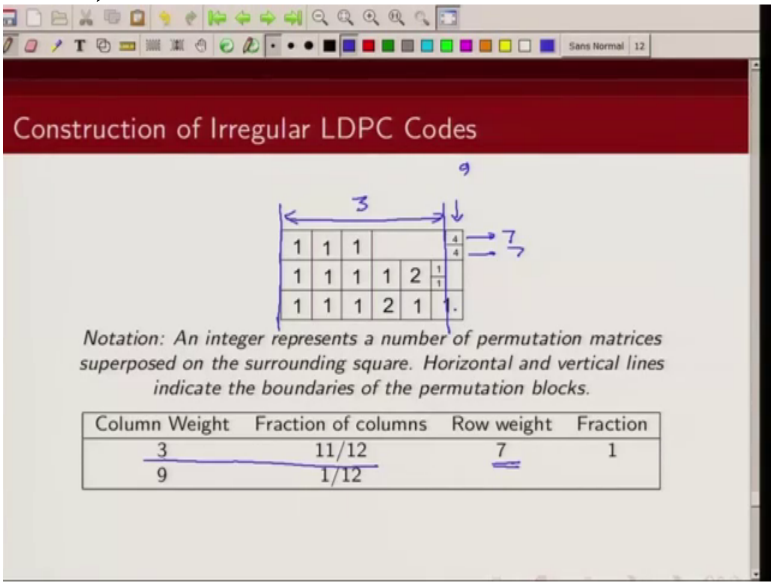
Each row will have weight 7. So it is the same profile, different construction. This is another

(Refer Slide Time 49:46)



construction. Same column weight distribution that 11 by 12 fraction of the bit should have column weight 3 and 1 by 12 should have column weight 9. And row weight should be 7 and you can check it. Each of the row here has weight 7 and all of these rows from here to here have column weight 3 and this has column weight 9; 4, 4 and this will have weight 1.

(Refer Slide Time 50:19)



So we can use random permutation matrix to construct irregular L D P C codes as well, thank you.