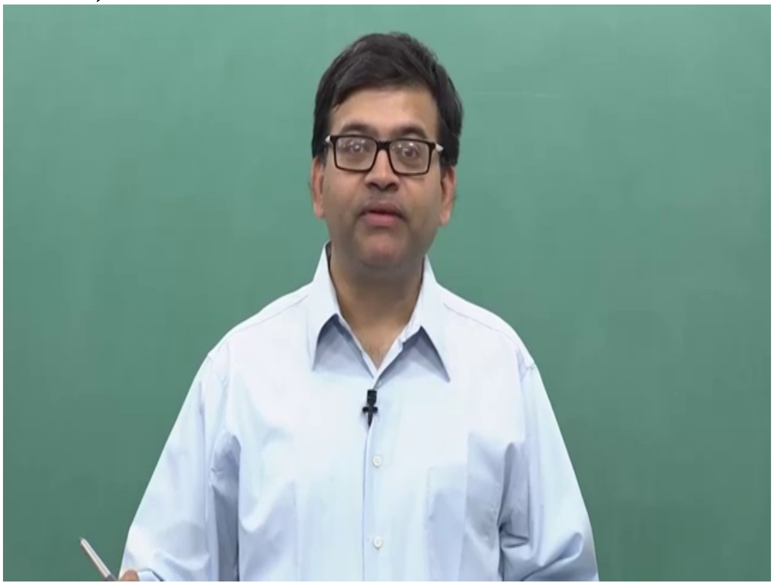
An Introduction to Coding Theory

Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

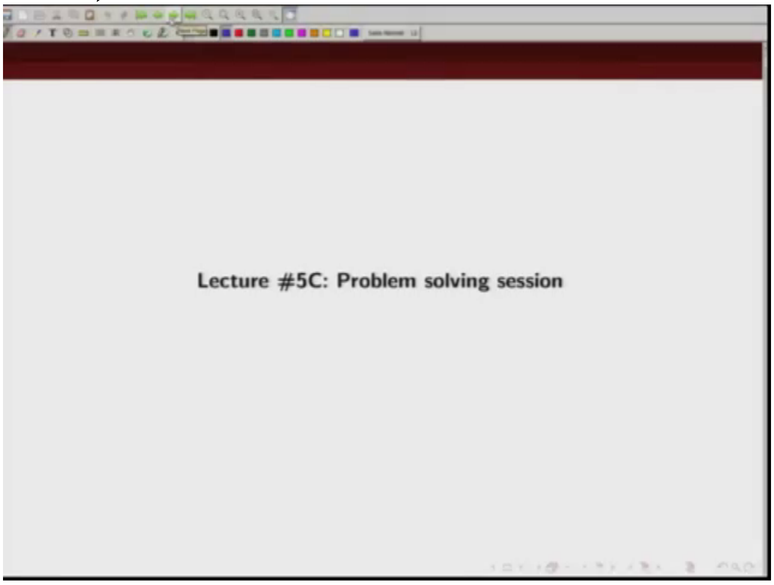
Module 02
Lecture Number 10
Problem Solving Session-II

(Refer Slide Time 00:27)



So, so far we have studied what are linear block codes, how do we describe linear block codes using generator matrix and parity check matrix, we talked about how we can use error correcting codes for error detection and error correction and we discussed the distance properties of linear block codes. Today we will spend some time solving some problems on linear block codes so today's session

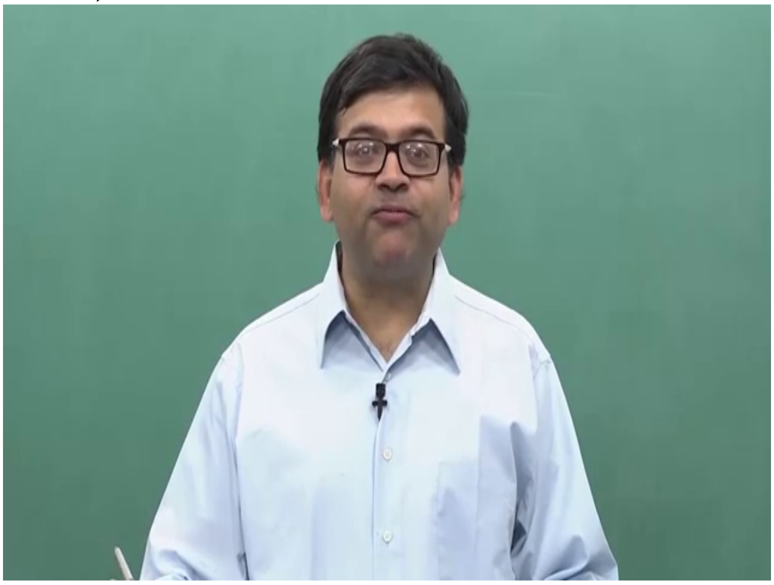
(Refer Slide Time 00:44)



will be on problem solving.

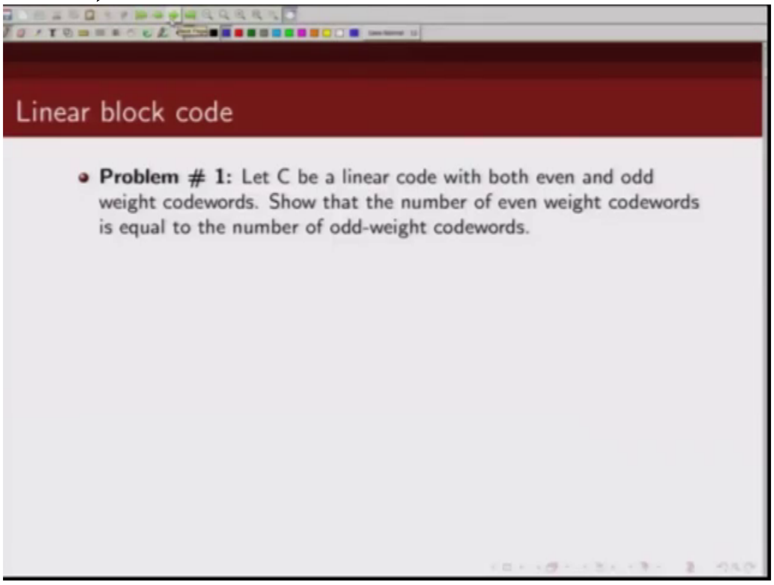
So the first problem

(Refer Slide Time 00:47)



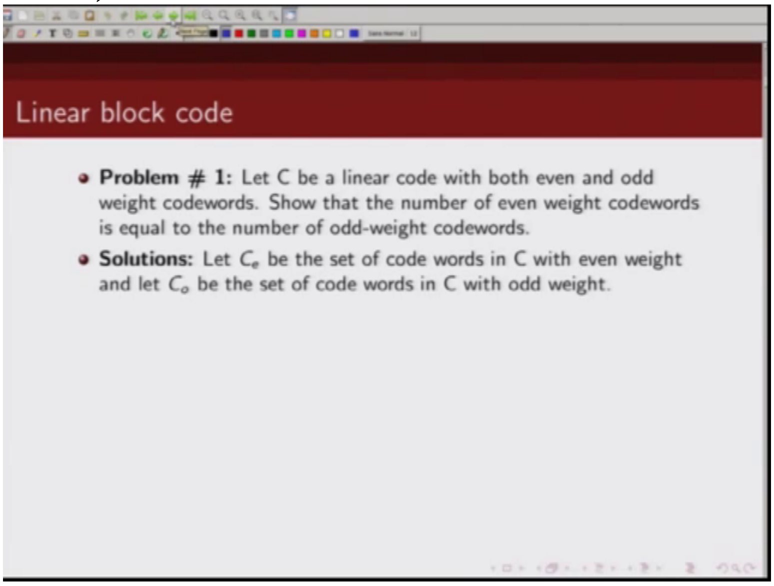
that we will look at is let C be a linear

(Refer Slide Time 00:51)



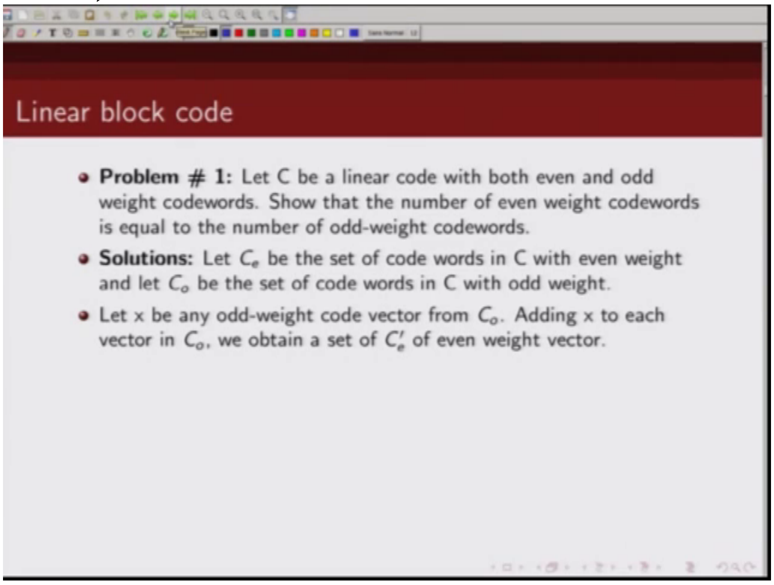
code with both even and odd weight codewords. Prove that the number of even bit codewords is equal to number of odd bit codewords.

(Refer Slide Time 01:04)



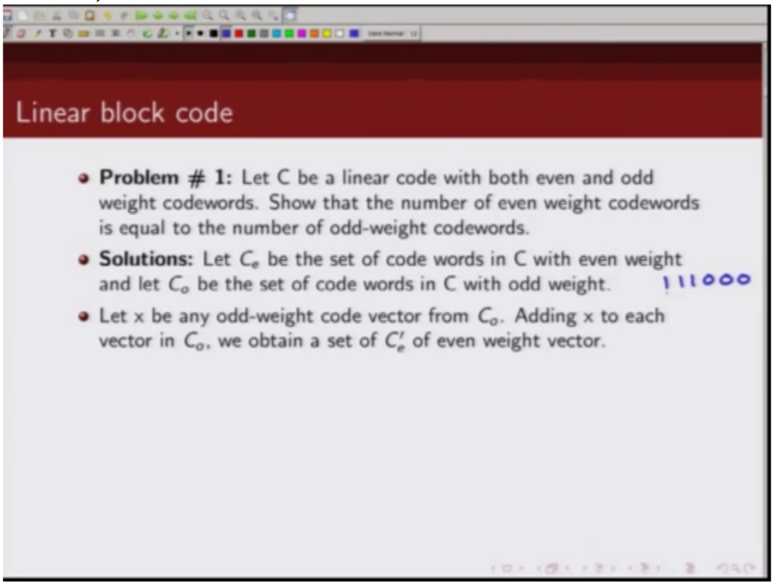
So let us denote the set of even codewords in C by C sub e and set of odd codewords in C by C sub o.

(Refer Slide Time 01:20)



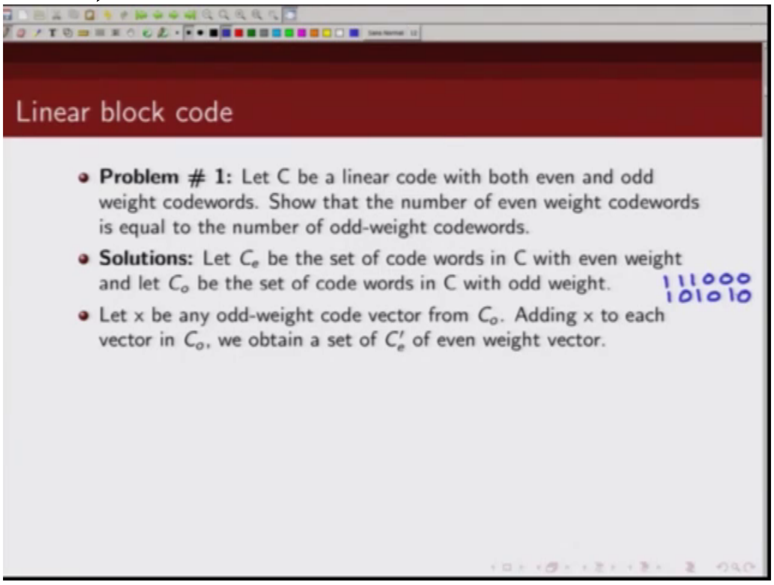
Now let us consider an odd weight codeword x which is taken from this set C o and let us add x to each of the codewords which are there in the set C o. So if we add a odd weight codeword to another odd weight codeword what we will get is a even weight codeword. For example let's say I add 1 1 1 0 0 0 and I add

(Refer Slide Time 01:58)



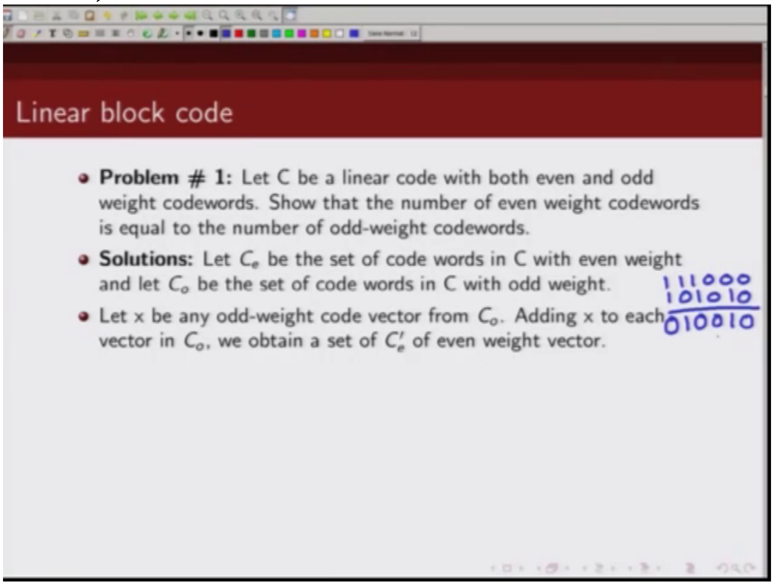
1 0 1 0 1 0 so

(Refer Slide Time 02:04)



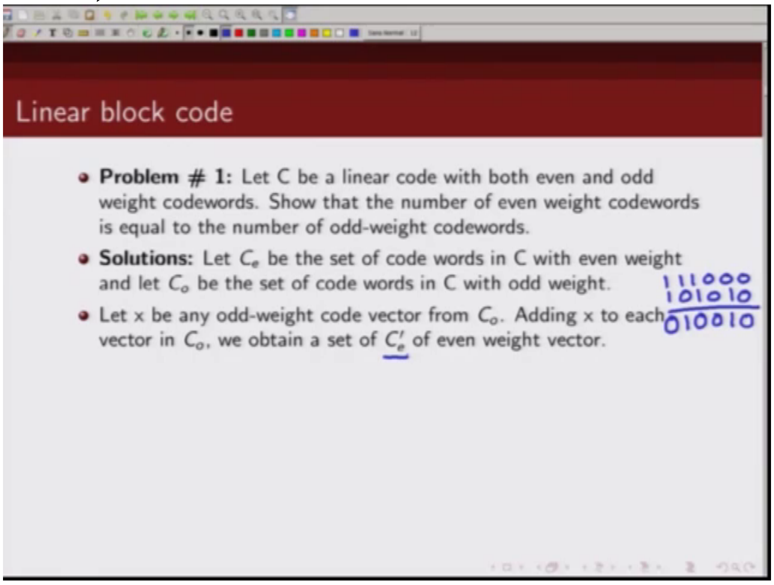
this first codeword this is odd weight codeword its weight is 3. Similarly this codeword also has weight 3. If I add both of them what do I get? I get 0 1 0 0 1 0 and this a

(Refer Slide Time 02:26)



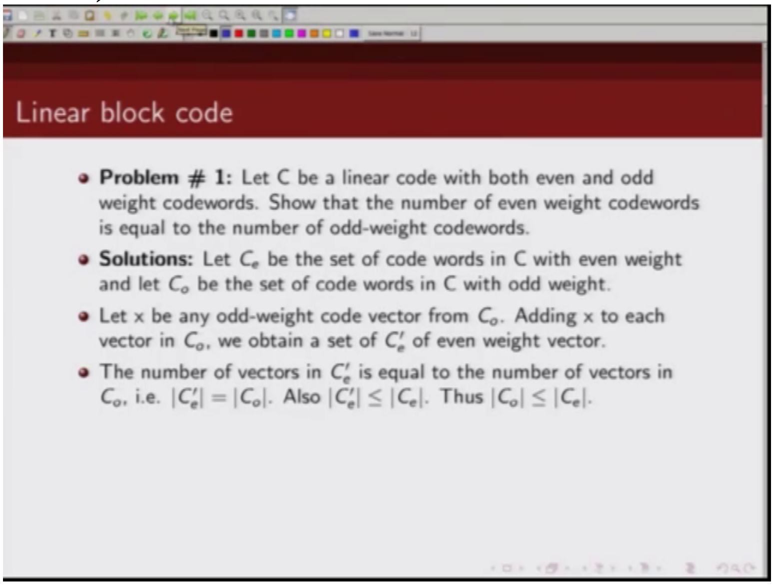
even weight codeword. So when I add x which is an odd weight code vector and I add x to each of the elements in this set C 0 what I get is a set of even codeword vectors and let us denote that set by

(Refer Slide Time 02:49)



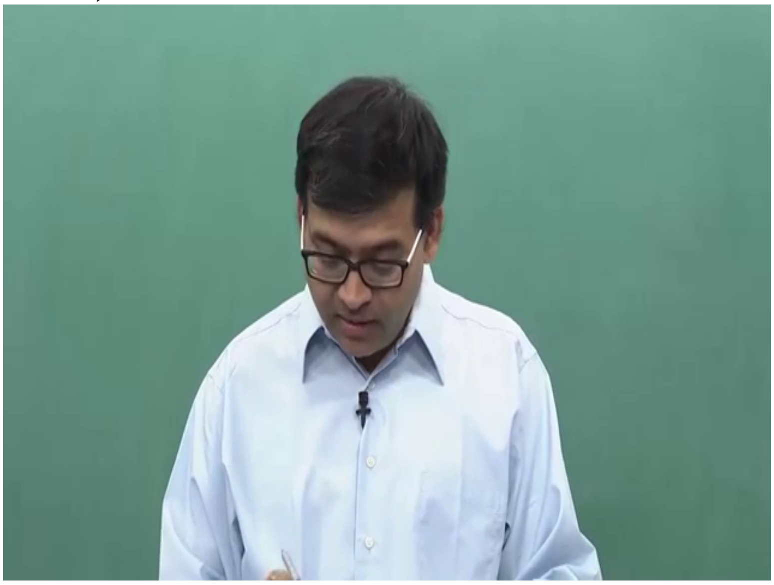
C e prime. Now

(Refer Slide Time 02:52)



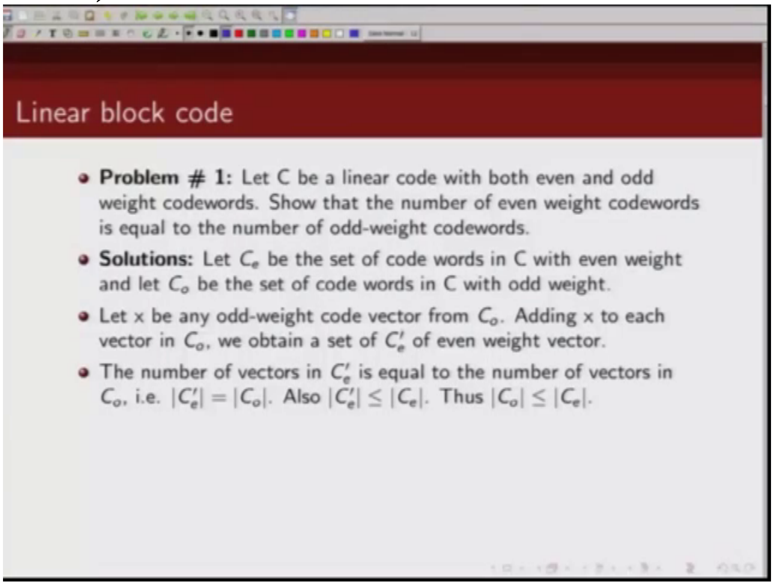
the number of code vectors in C e prime is going to be equal to number of vectors in C o. Why, because

(Refer Slide Time 03:02)



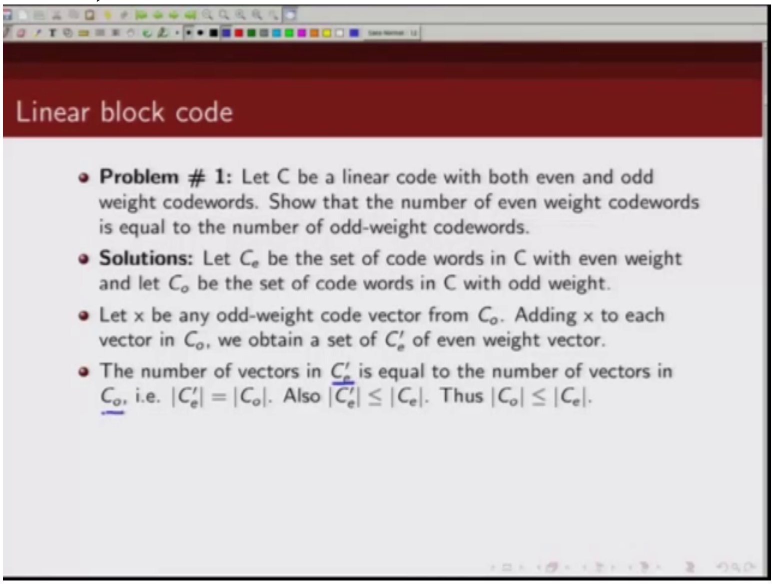
how did we get this c e prime? We added

(Refer Slide Time 03:05)



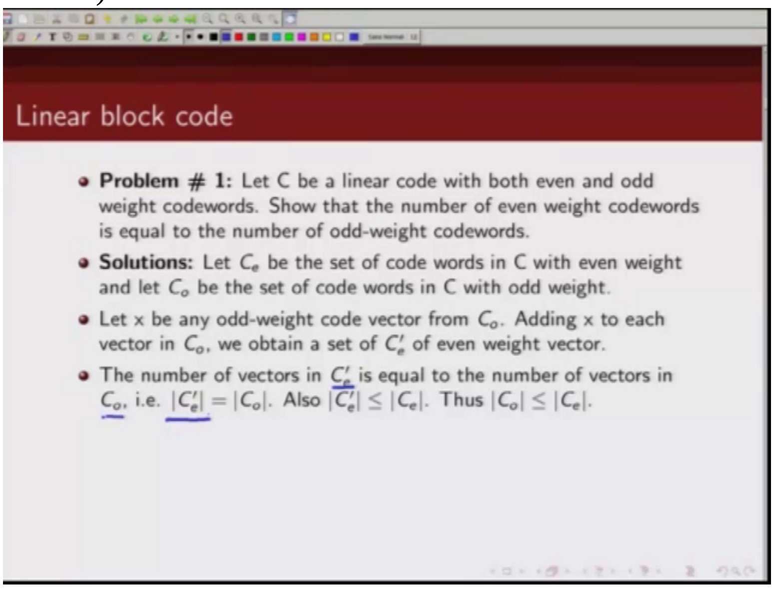
an odd vector x to the set C o. So number of vectors in this set is going to be equal to

(Refer Slide Time 03:15)



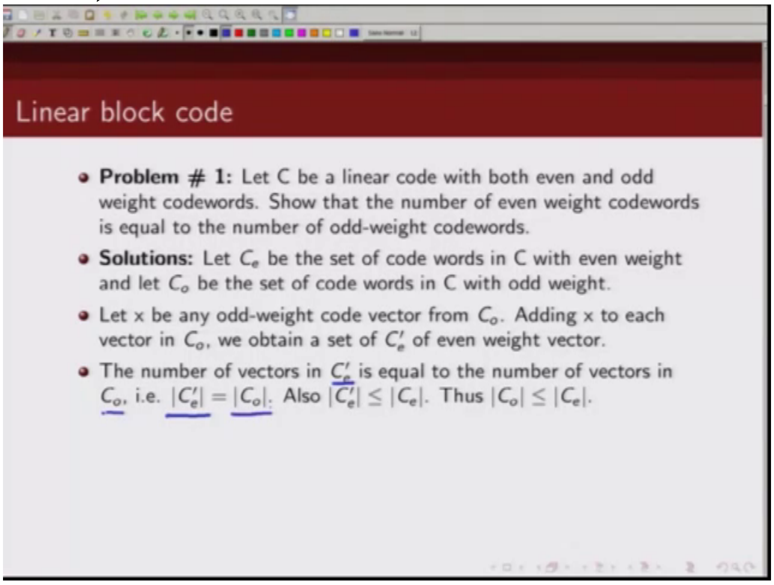
number of vectors in C o. Hence number of elements in

(Refer Slide Time 03:23)



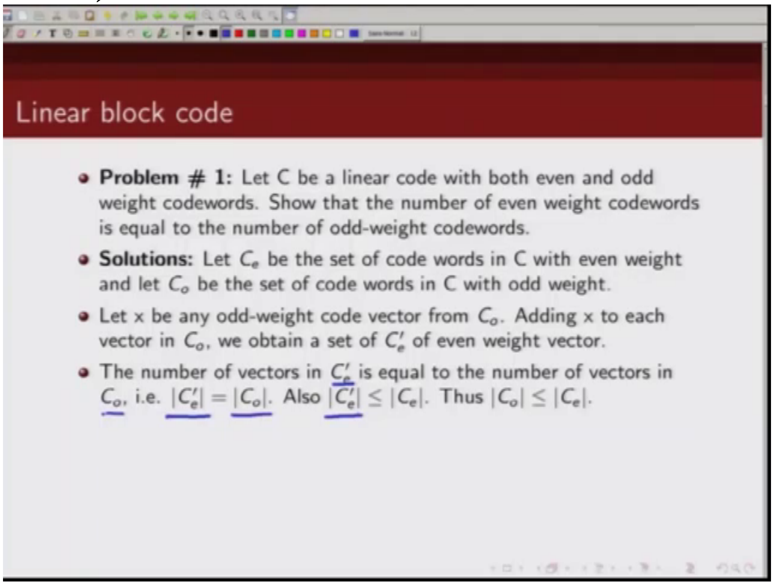
C e prime is going to be same as number of elements in

(Refer Slide Time 03:29)



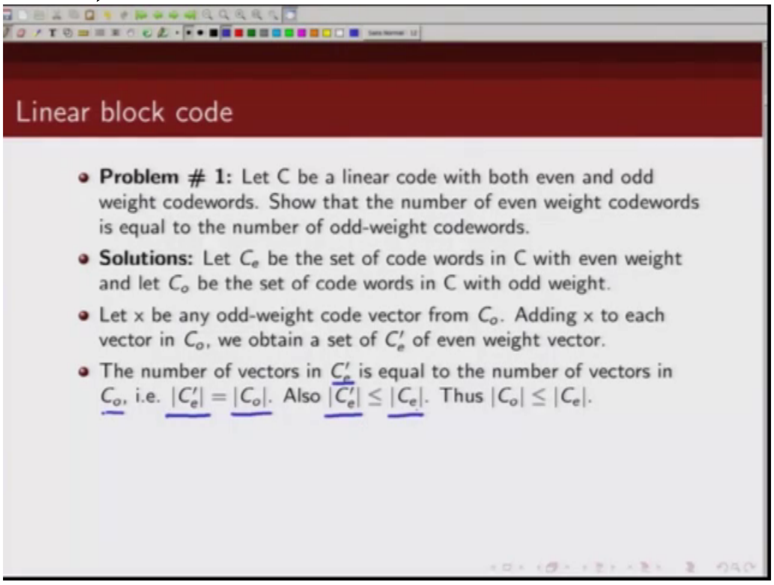
in C o and since we know that this set

(Refer Slide Time 03:35)



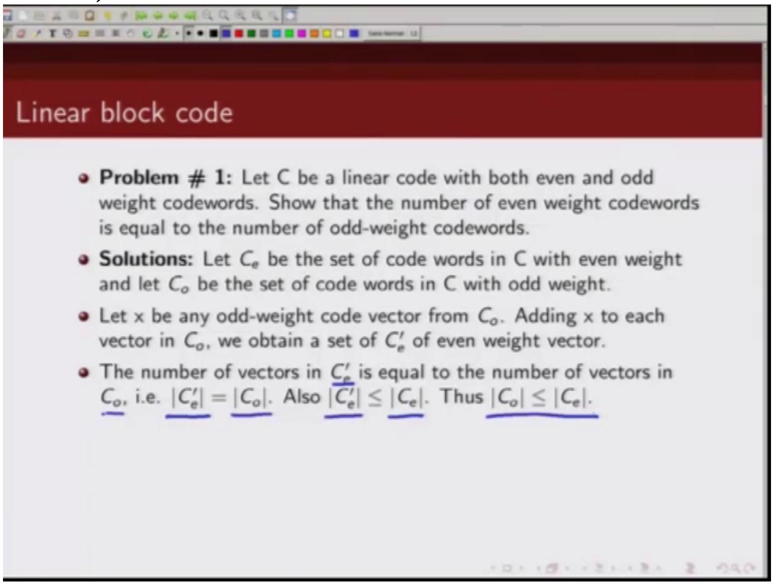
of even vectors C e hat is the subset of

(Refer Slide Time 03:41)



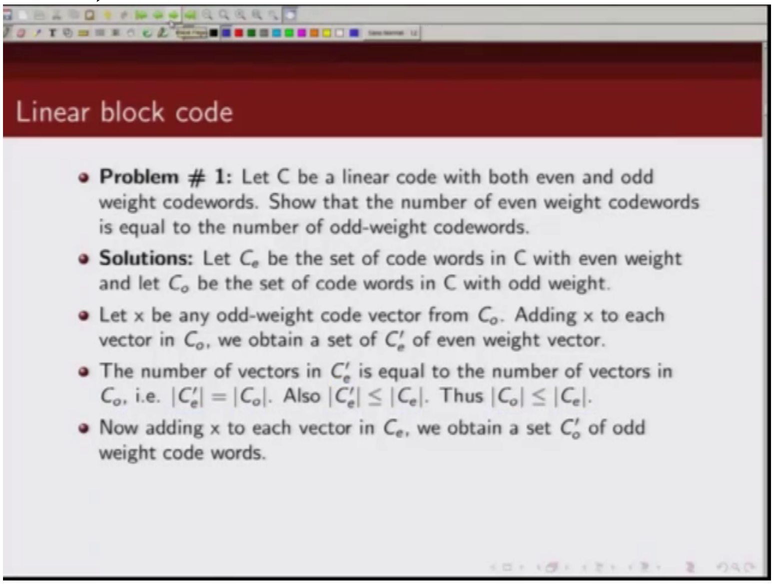
set of even vectors we can write from this that number of elements in the

(Refer Slide Time 03:50)



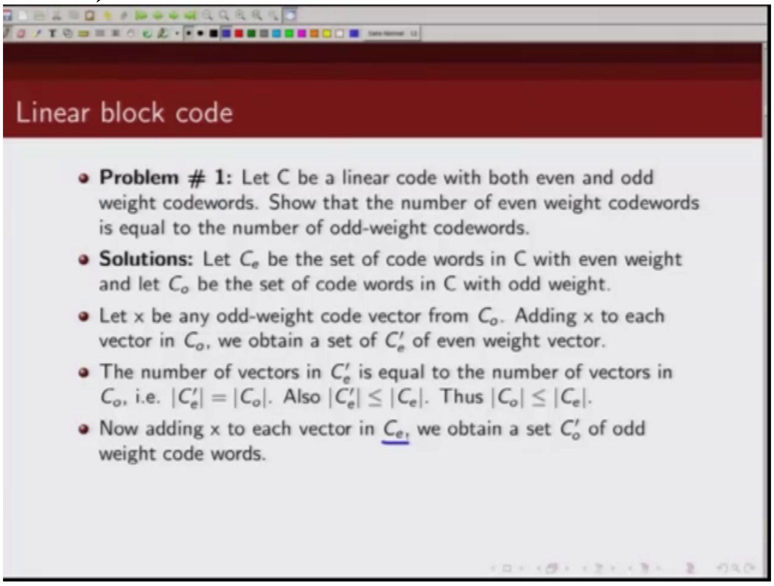
set of number of odd codewords is going to be a subset of number of even codewords. Next,

(Refer Slide Time 04:02)



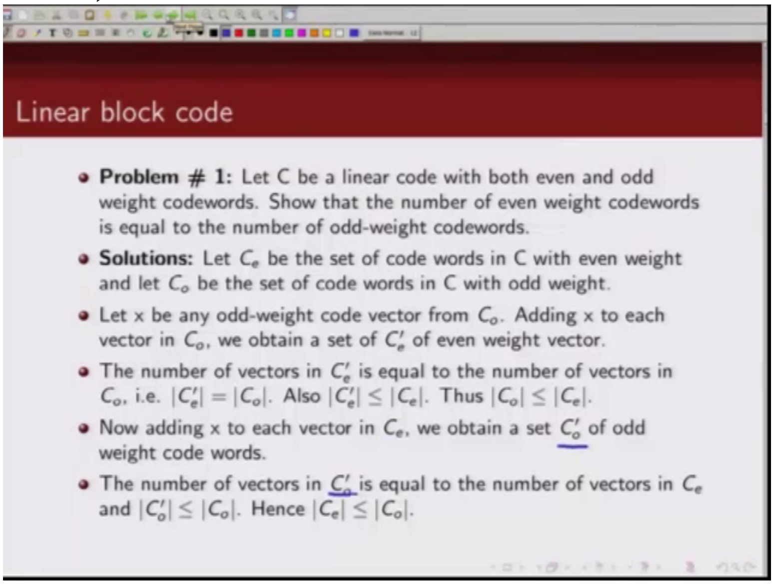
let us add the same odd weight codeword now to

(Refer Slide Time 04:08)



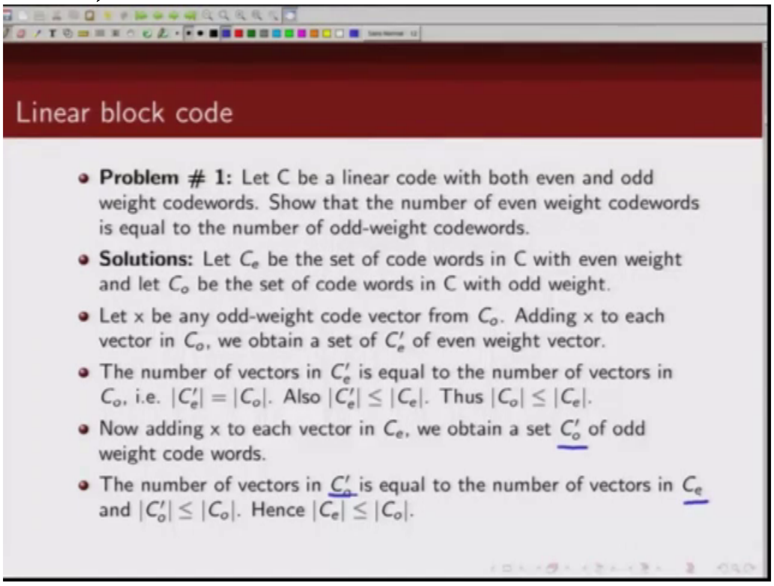
all the vectors in the set C e. So if we add an odd weight codeword to set of even weight codewords what we will get is a set of odd codewords. So the

(Refer Slide Time 04:28)



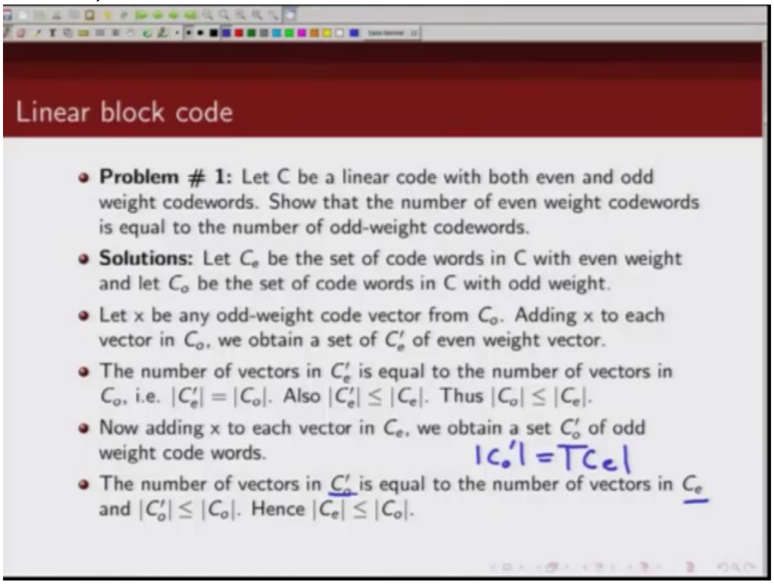
number of vectors in C o prime is going to be equal to number of vectors in, number of even vectors.

(Refer Slide Time 04:41)



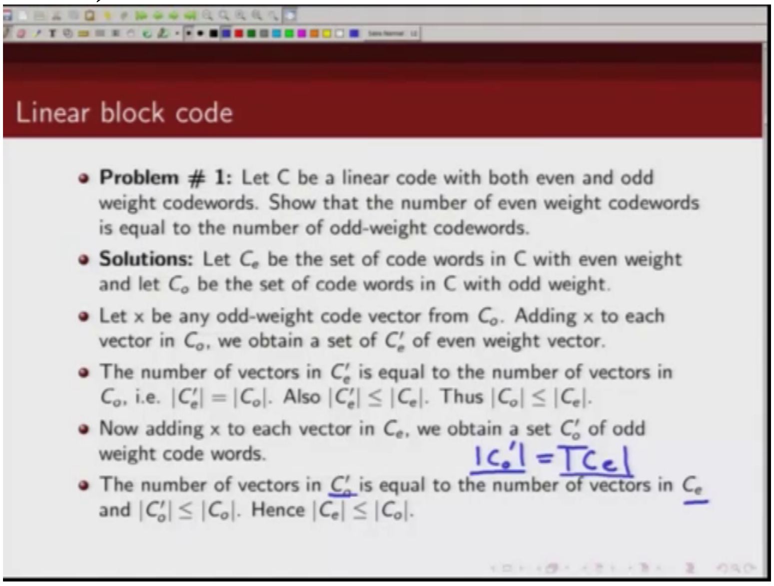
Why, because this set was generated by adding an odd vector x to the set of even codewords. So we can then write that C o prime is equal to this ok,

(Refer Slide Time 05:06)



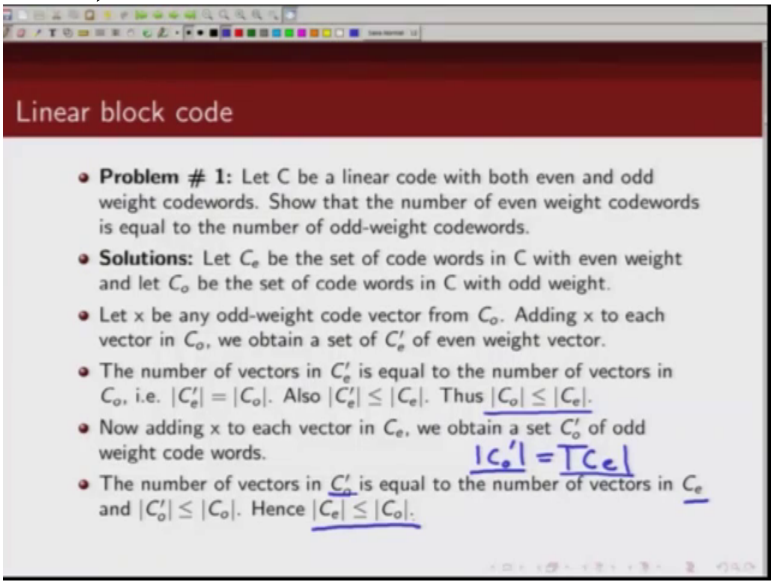
the set of codewords here is same as set of codewords here.

(Refer Slide Time 05:10)



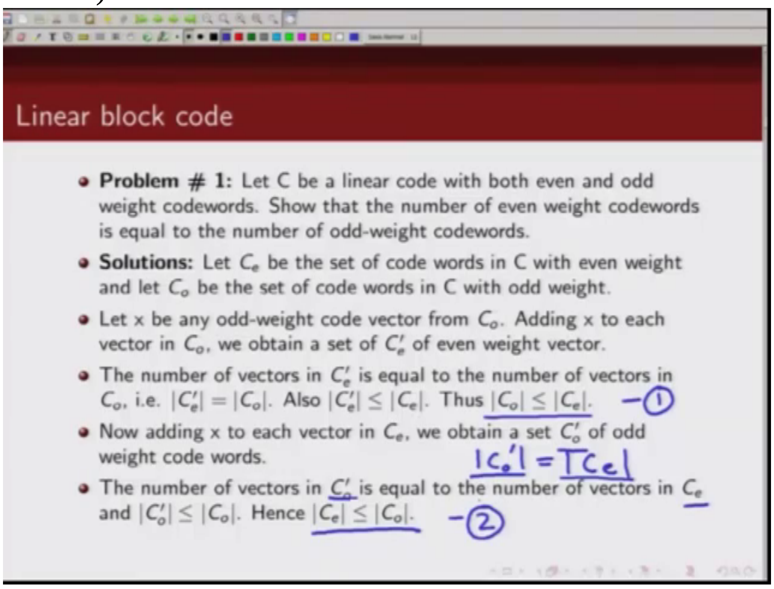
Now we know that C o dash is the subset of set of odd codewords. So then from this relation and this relation we can write that set of even codewords is the subset of set of number of elements in this is subset of number of is basically less than number of elements in this set. Now from this relation and this relation

(Refer Slide Time 05:43)



both of them can be true only if number of elements in C o is same as number of elements in C e. So this relation we call it 1 and let's call it 2. These

(Refer Slide Time 05:58)

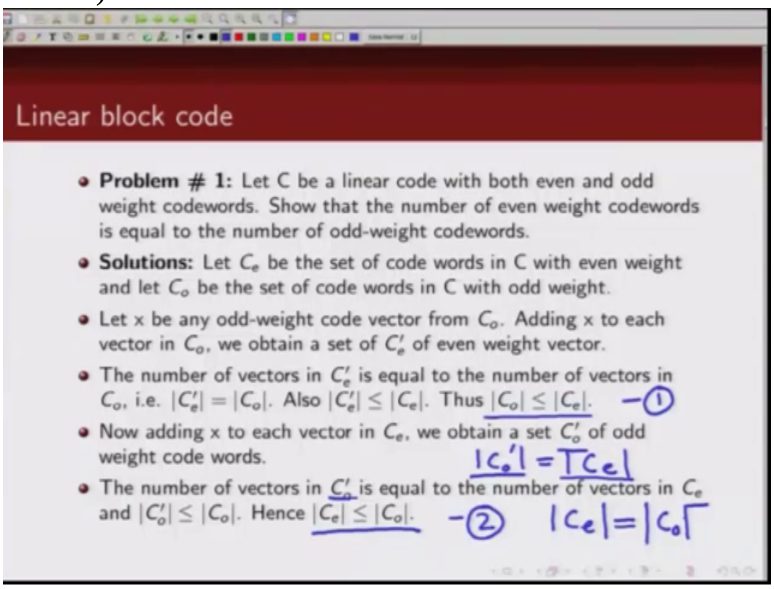


two relations are satisfied only if

(Refer Slide Time 06:02)

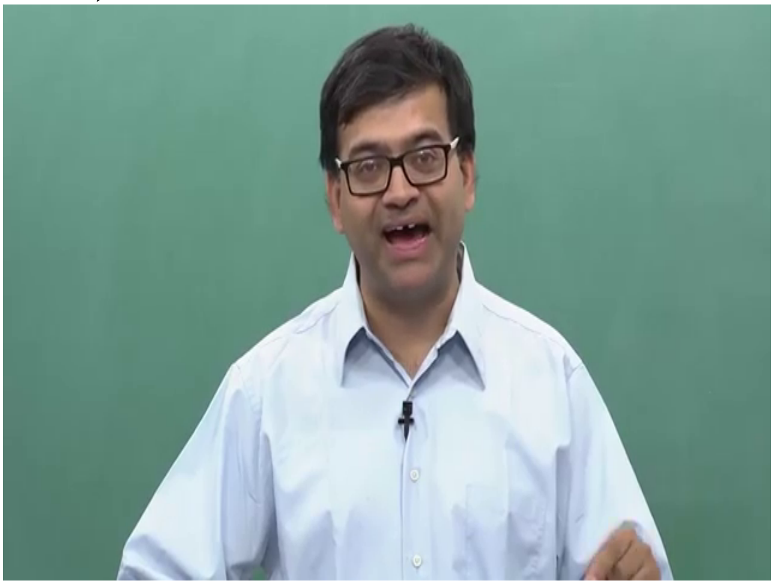
we have set of even codewords to be same as set of odd codewords.

(Refer Slide Time 06:09)



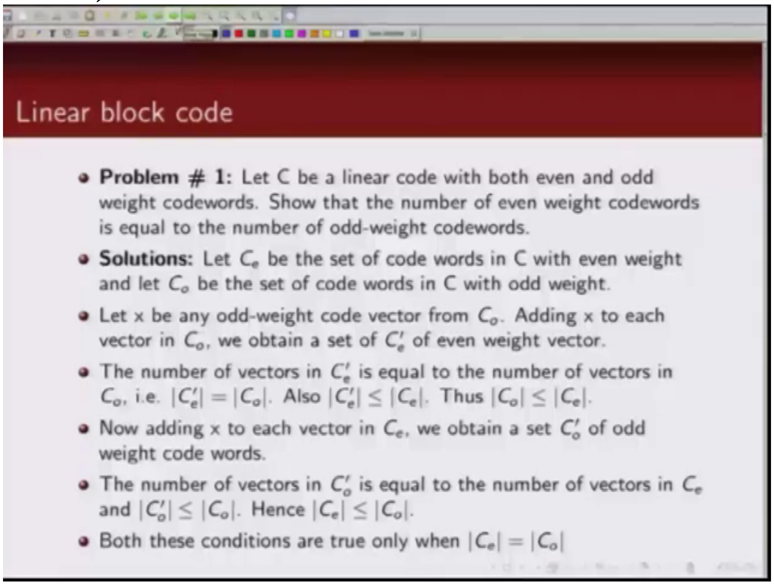
Hence we prove that in a linear code with both even and odd codewords the number of even weight codewords is same as number

(Refer Slide Time 06:21)



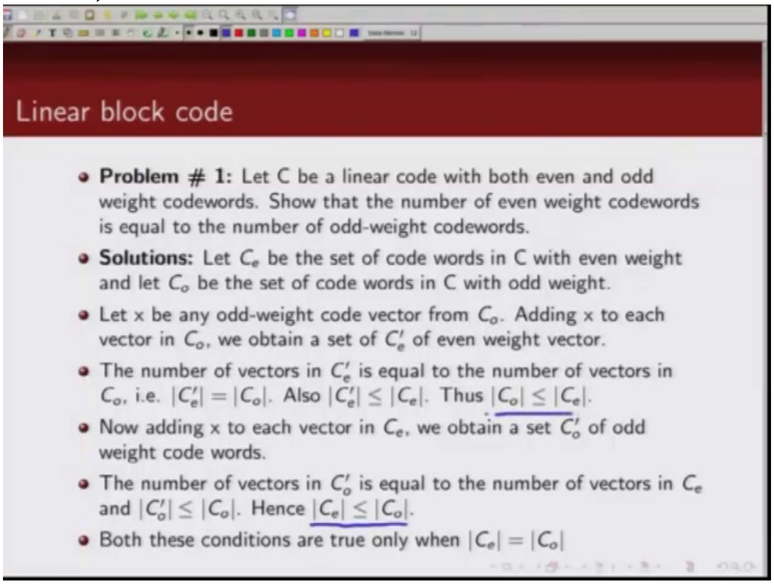
of odd weight codewords.

(Refer Slide Time 06:24)



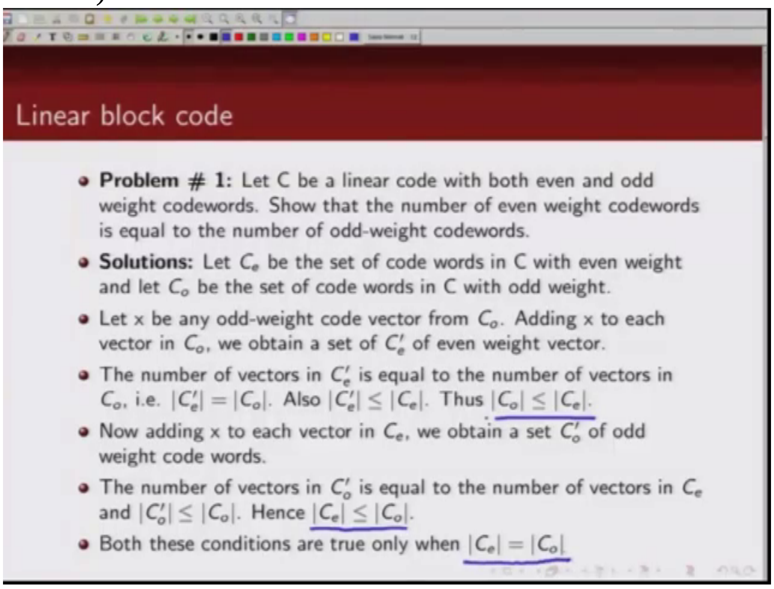
So I repeat, this condition and this condition

(Refer Slide Time 06:29)



will be simultaneously satisfied only when

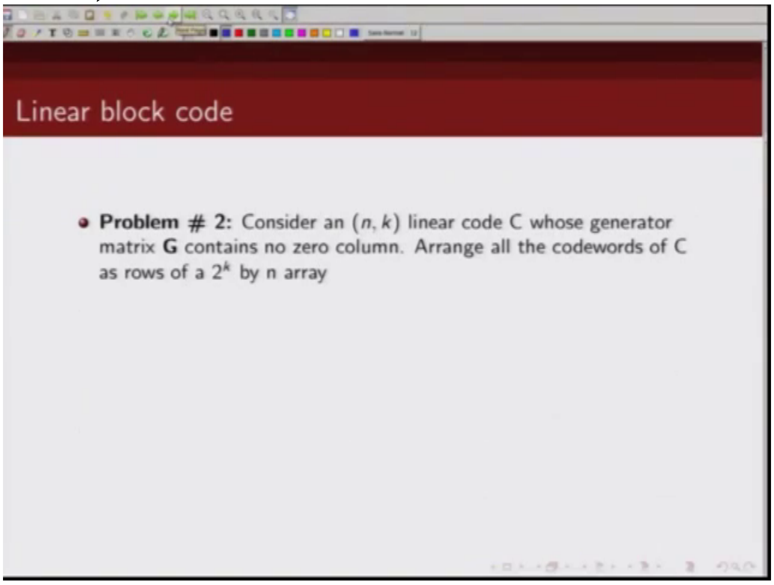
(Refer Slide Time 06:35)



the set of number of even codewords is same as set of number of odd codewords. And this proves our result.

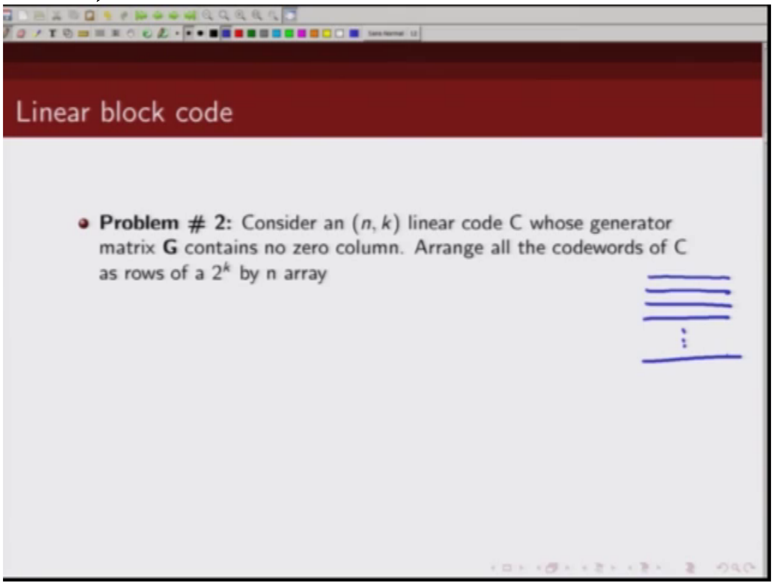
The next problem

(Refer Slide Time 06:48)



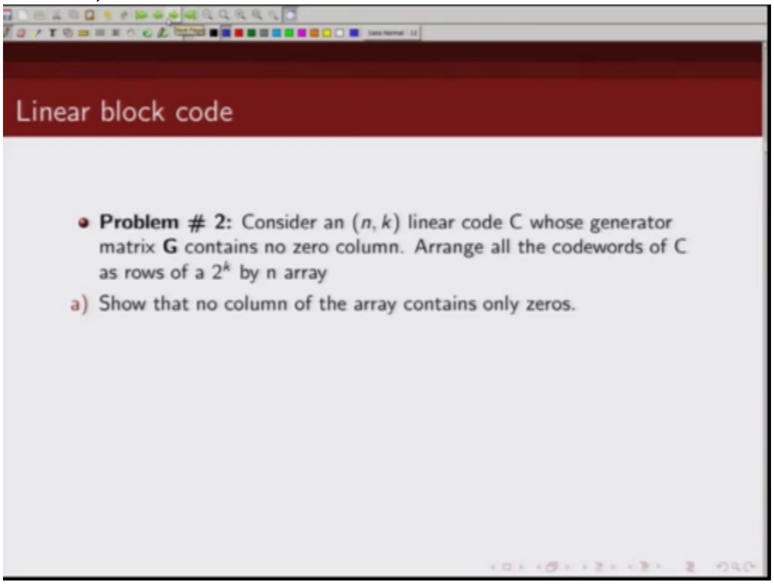
we will look out is as follows. Let us consider a linear n k code C whose generator matrix contains no zero column; now arrange all codewords of this linear code as rows of two raised to power k by n array. So what we are doing is we are arranging the 2 k codewords like this, in an array. So this array has dimensions 2 k cross n because total number of codewords

(Refer Slide Time 07:28)



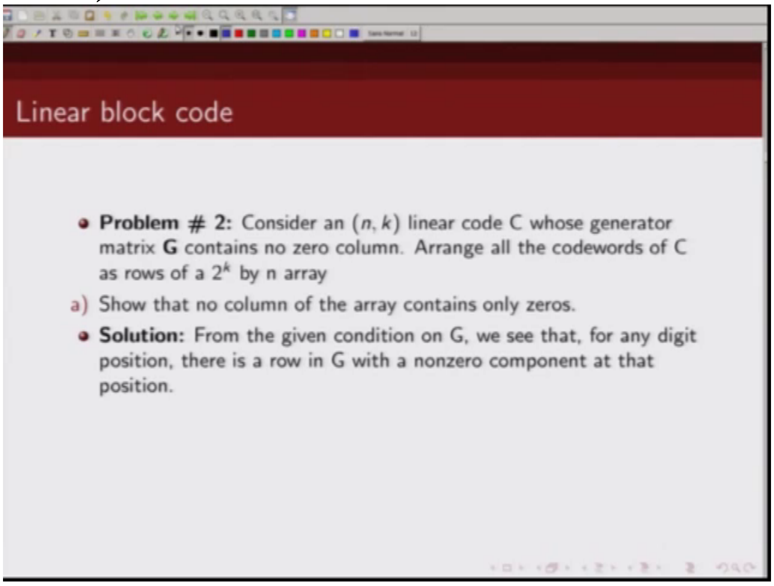
are 2 raised to power k for a n k binary code and they are all n-bit. The first result that we are

(Refer Slide Time 07:37)



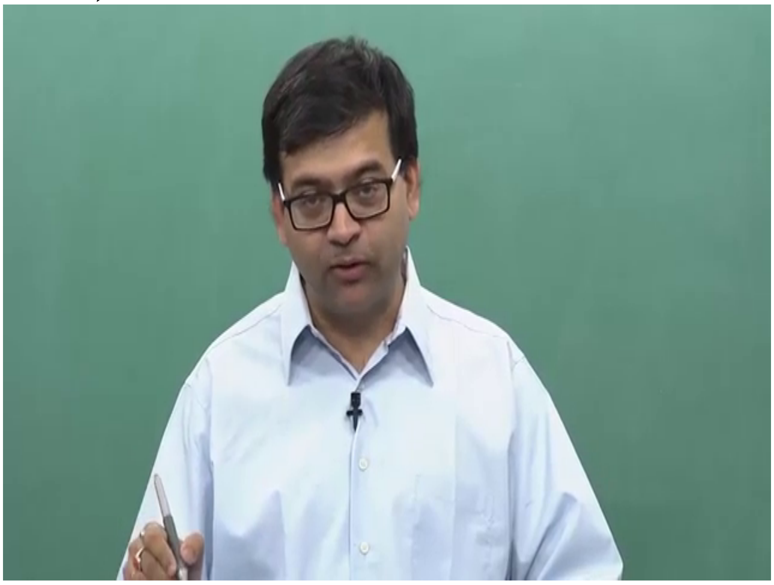
going to show is no columns of this array contain 0. Now please note that we have

(Refer Slide Time 07:48)



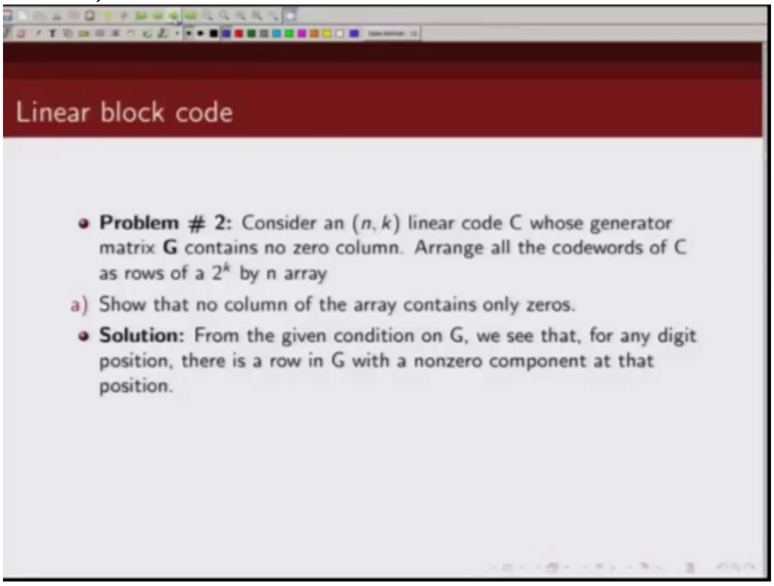
been given that this generator G does not contain any zero column ok so from the given condition on G we can see that for any position of any bit position there is a row in G which has non-zero component at that particular

(Refer Slide Time 08:12)



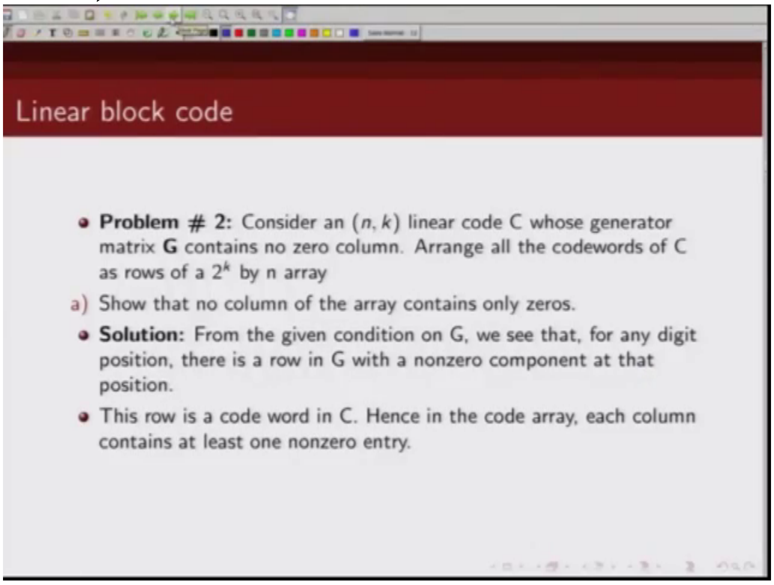
bit location and

(Refer Slide Time 08:14)



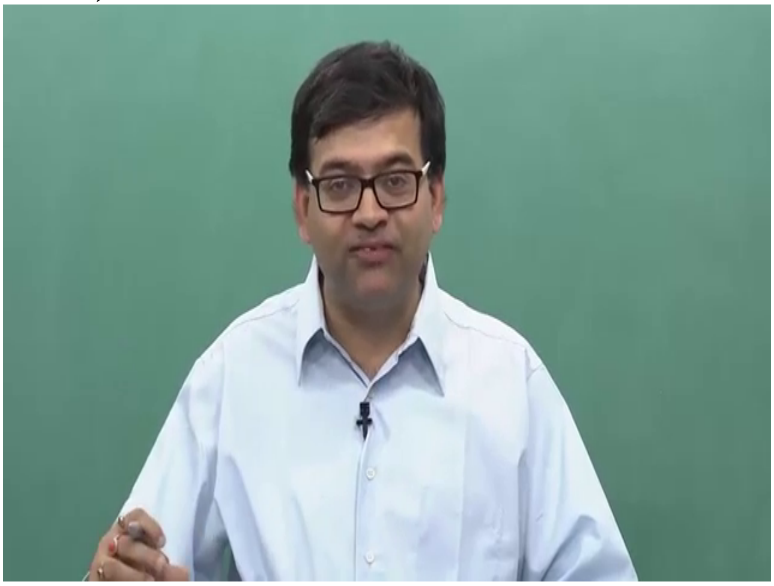
if this is

(Refer Slide Time 08:15)



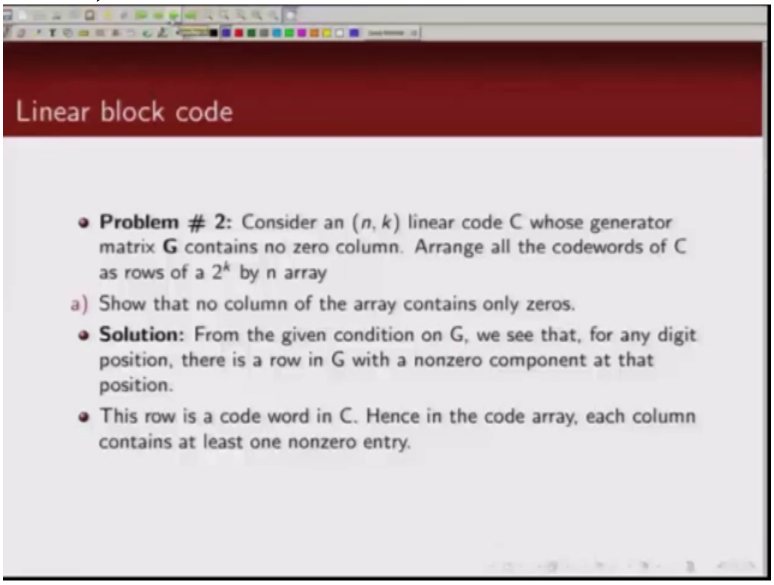
true what are the rows of, how do we generate the codewords? We generate the codewords by linear combination of these

(Refer Slide Time 08:22)



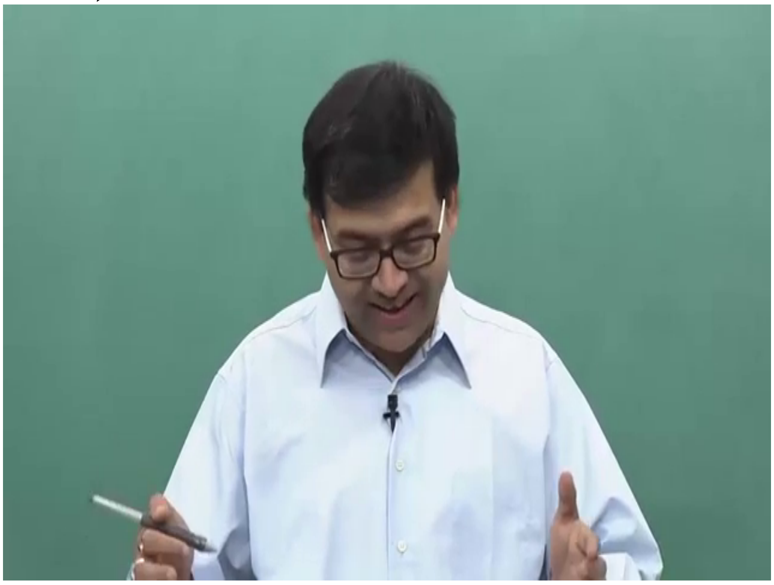
rows of the generator matrix. And since the generator matrix does not contain

(Refer Slide Time 08:29)



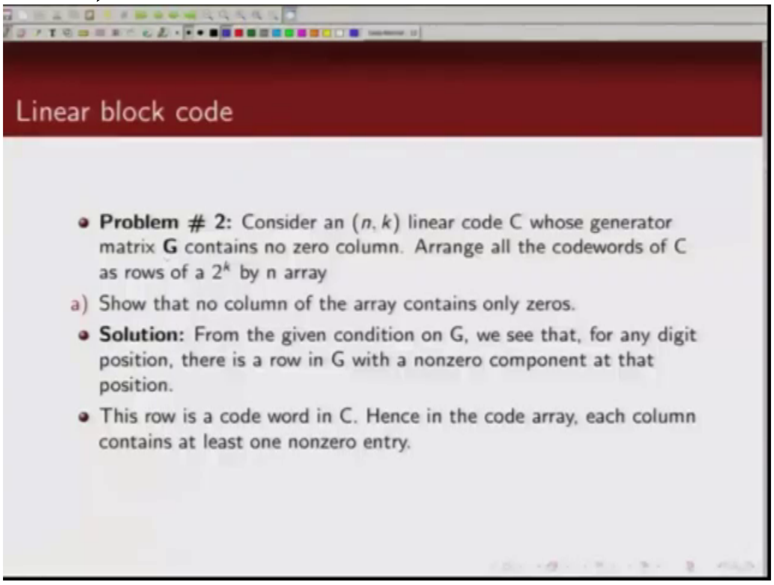
any zero column so each of these rows can be looked up as codeword in C so when we generate the codewords using this generator matrix in this code array, each column will have at least one

(Refer Slide Time 08:46)



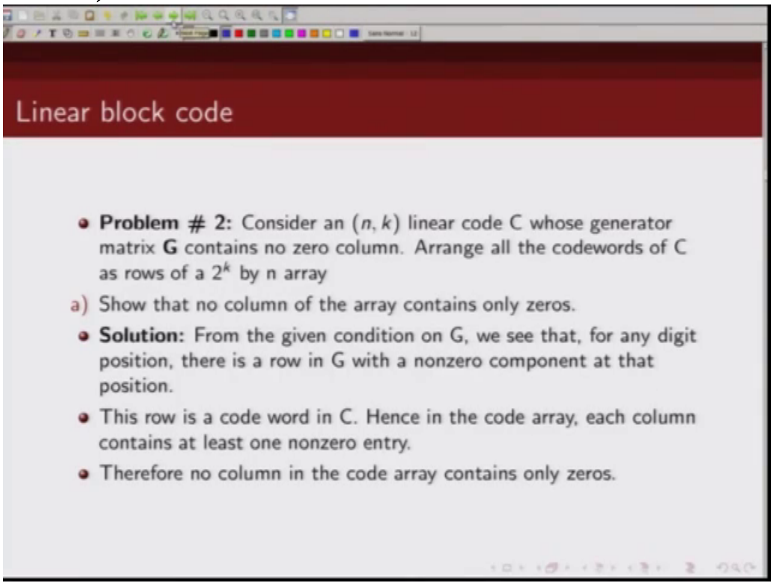
non zero entry. So this follows from the fact

(Refer Slide Time 08:51)



that our generator matrix G does contain any zero column

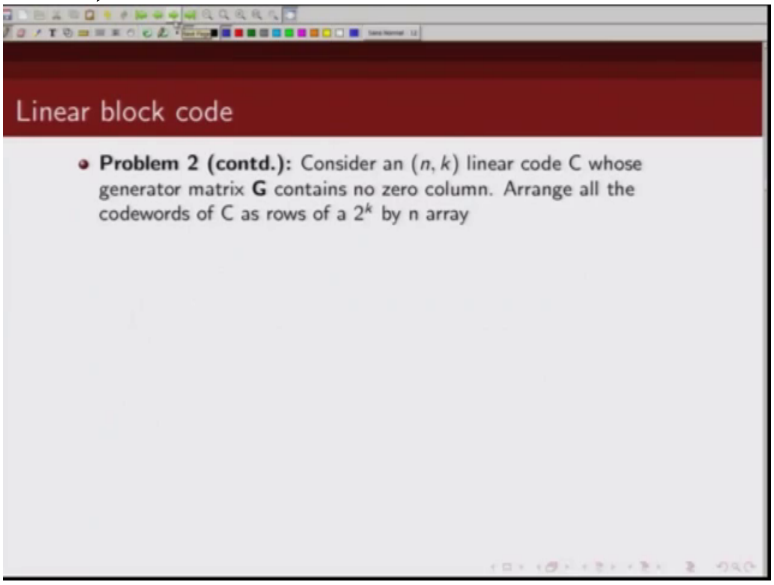
(Refer Slide Time 08:59)



and hence no column in this code array will have zeroes.

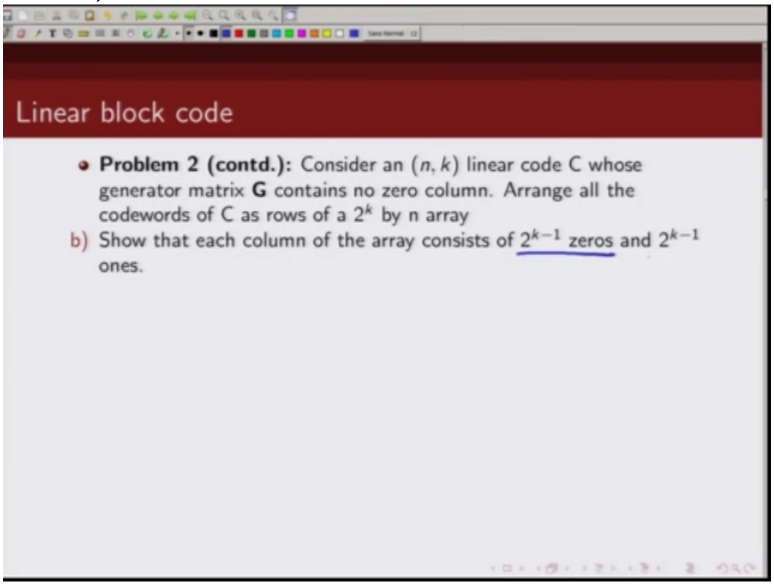
The next result that we

(Refer Slide Time 09:08)



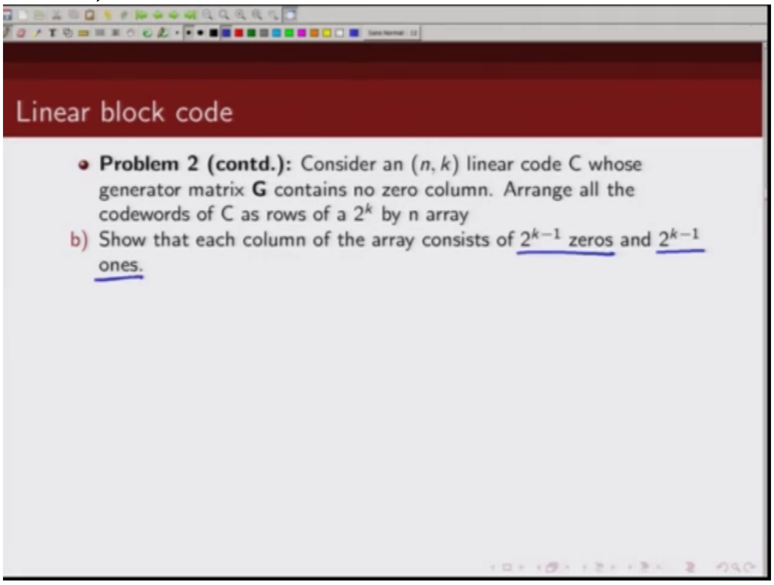
are going to show is in this array, in this 2 raised to power k cross n array each column consists of equal numbers of

(Refer Slide Time 09:21)



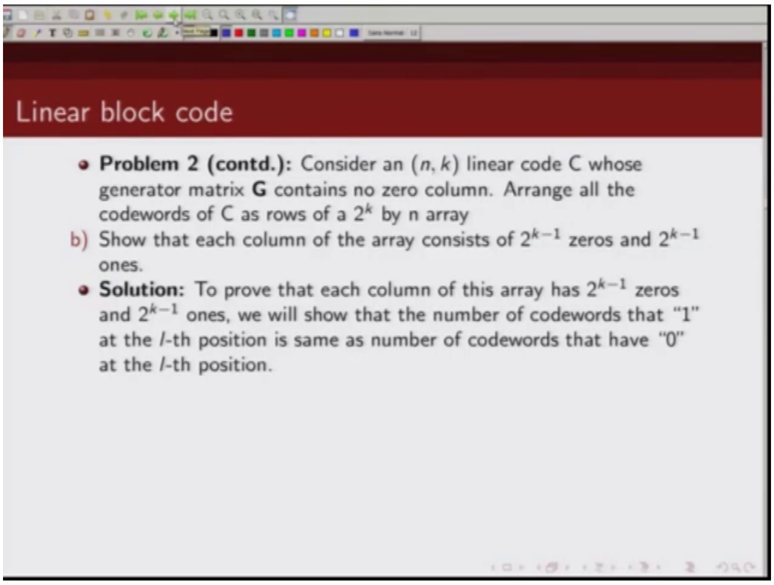
0s and 1s.

(Refer Slide Time 09:25)



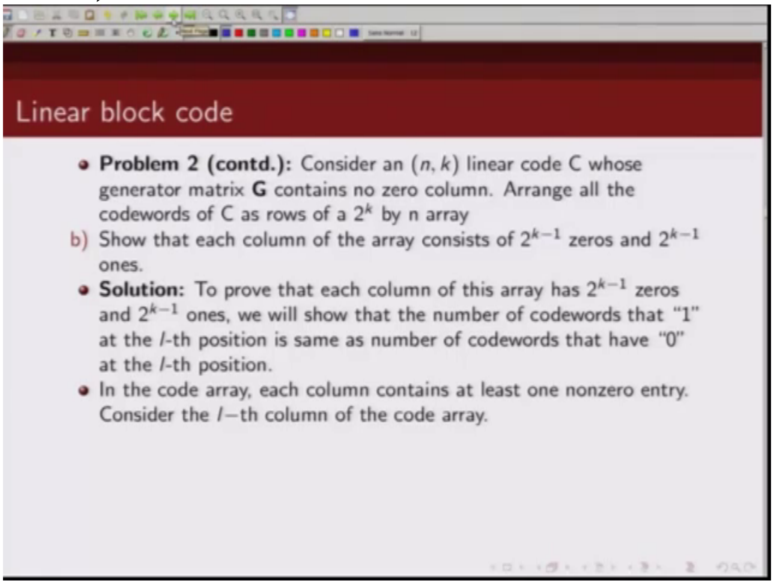
There are total 2 raised to power k minus 1 0s and 2 raised to power k minus 1 1s. So to prove

(Refer Slide Time 09:36)



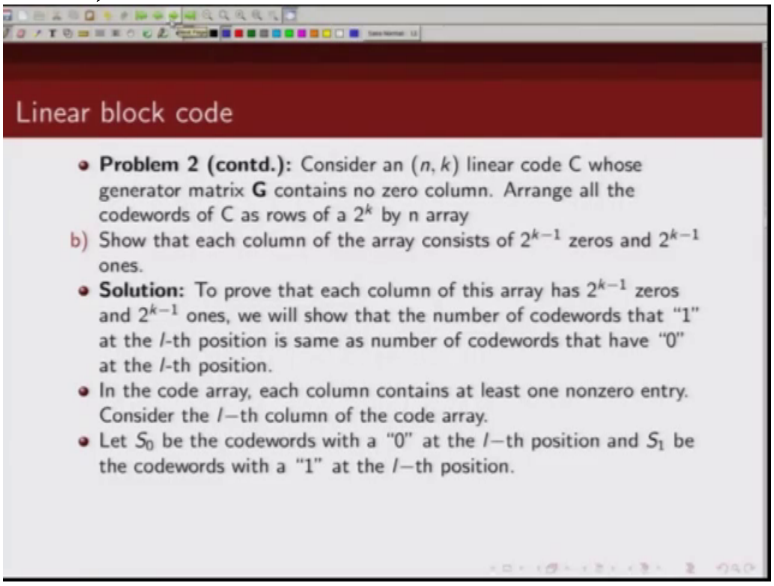
this what we will do is we will show that number of codewords that have 1 at lth location is same as number of codewords that have 0 at lth location. And in this way we will prove that this array has same number of 0s and 1s. So in this code array

(Refer Slide Time 10:01)



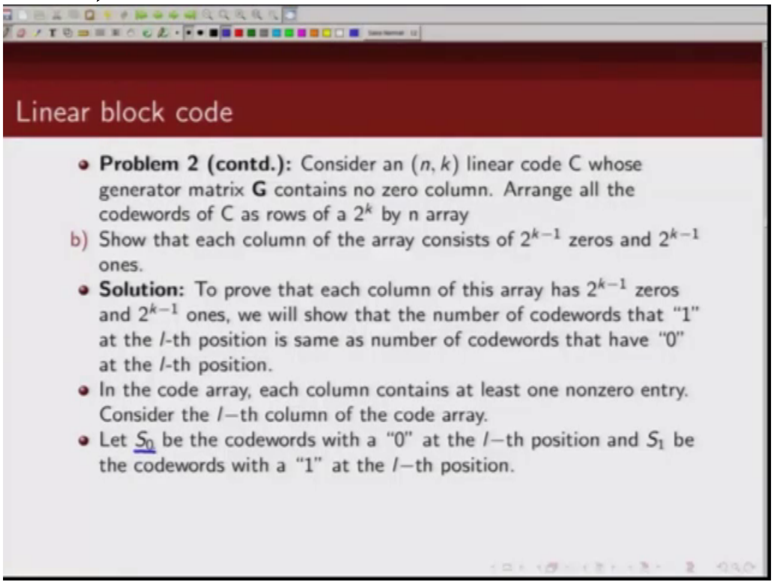
we know that each column will have at least one non-zero entry that we proved in the earlier result. So consider the lth column of this code array.

(Refer Slide Time 10:12)



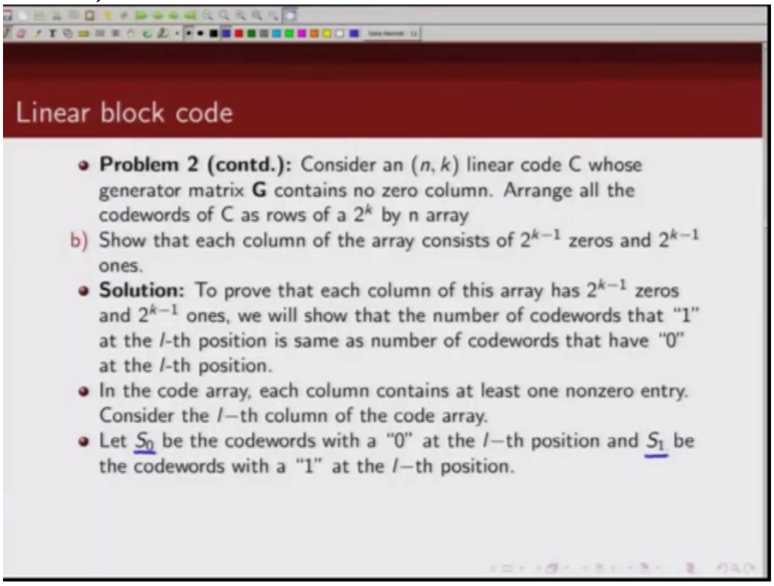
. Let us denote by S 0

(Refer Slide Time 10:17)



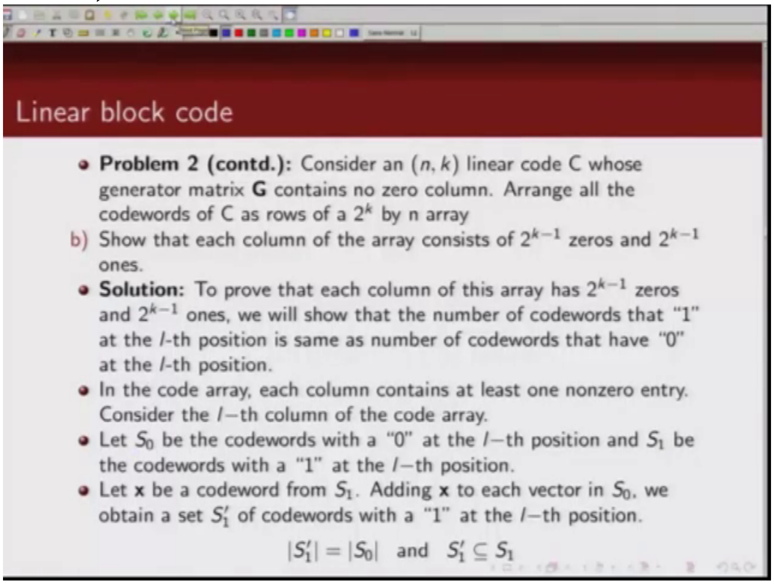
the set of codewords that has 0 at the lth location. And let us denote by S 1

(Refer Slide Time 10:27)



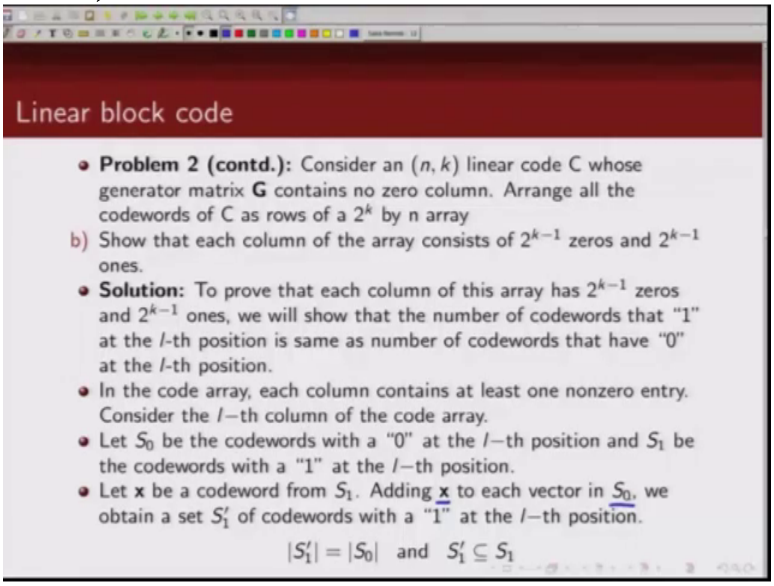
the set of codewords that have 1 at the lth location. Now we pick up a

(Refer Slide Time 10:36)



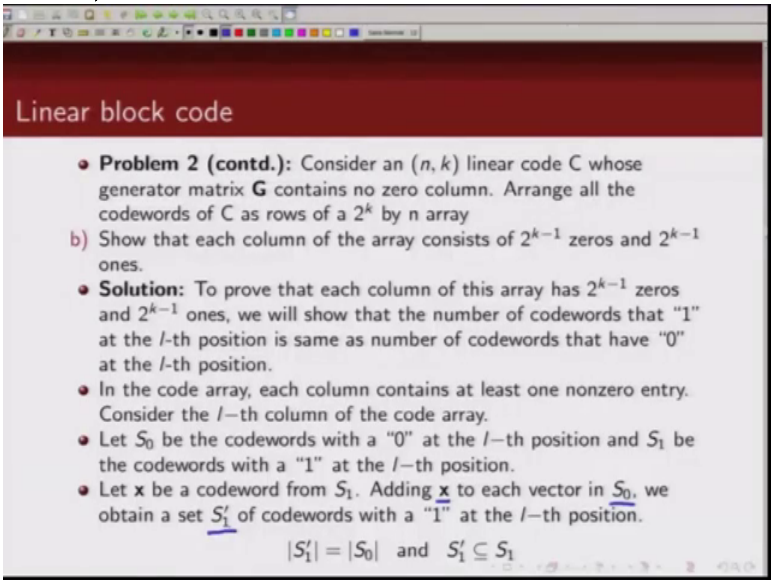
codeword x from the set S 1 that means x has 1 at lth location. Now if we add this codeword x to all the elements in the set S 0 what do we get?

(Refer Slide Time 10:58)



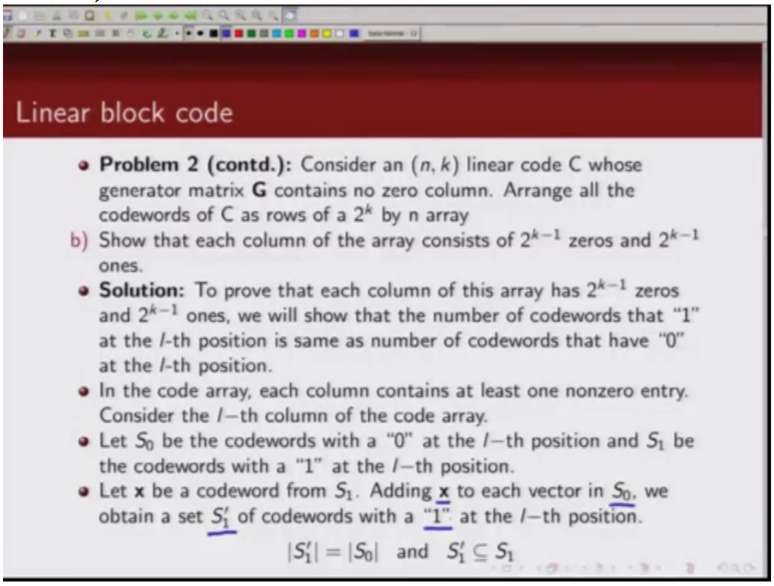
What we will get is a set containing 1 at lth location. Why, because S 0 is a set that has 0 at the lth location and x has, x is taken from the set S 1 so x has 1 at lth location. So if we add x to S, elements in S 0 what we will get is there will be a 1 at the lth bit location. So we denote this class

(Refer Slide Time 11:32)



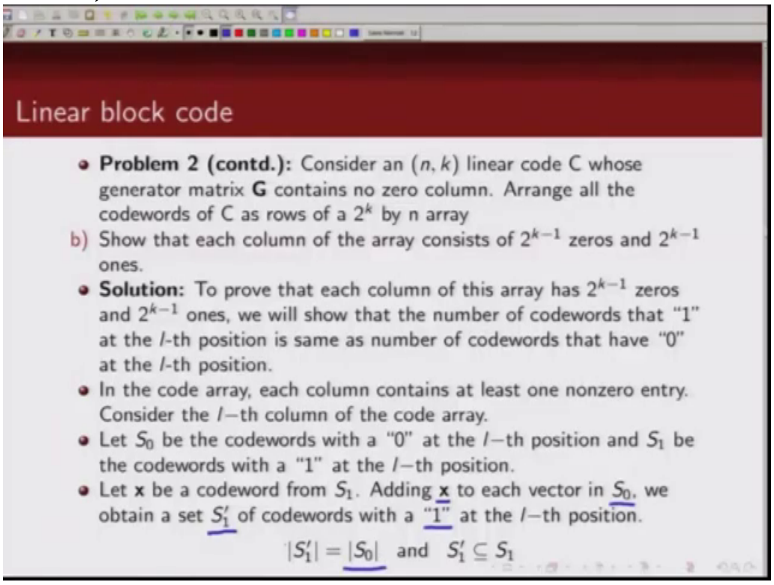
of codeword by S 1 prime and this S 1 prime will have 1 at the

(Refer Slide Time 11:40)



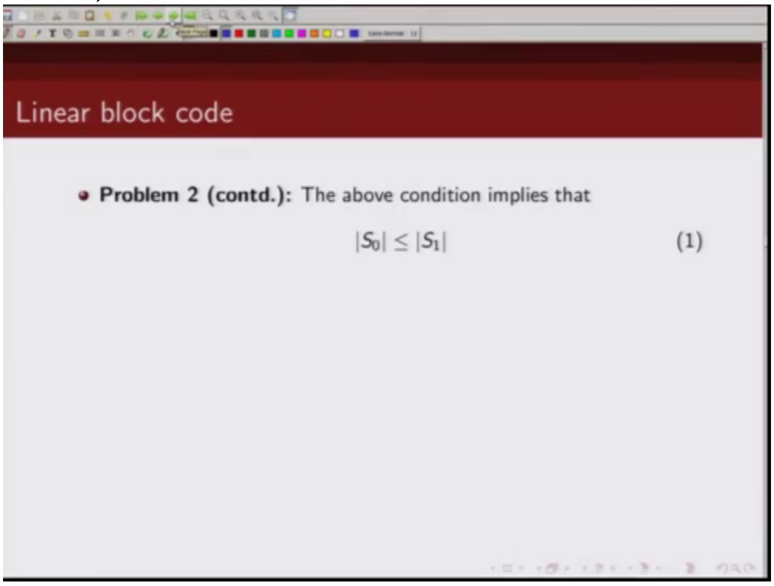
lth location and since this S 1 prime is generated by adding x to this set of vectors in S 0 so number of elements in S 0 is going to be

(Refer Slide Time 11:52)



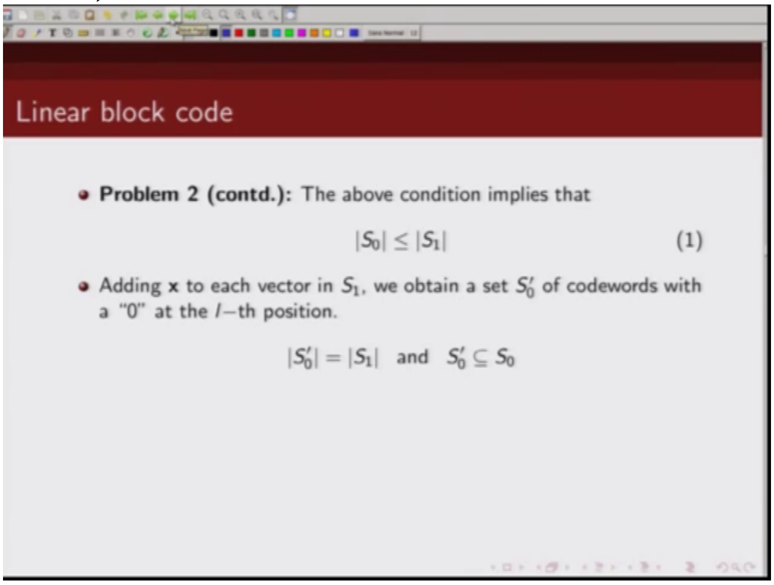
same as number of elements in S 1 prime and S 1 prime is a subset of S 1 which is a set of all codewords which have 1 at lth location.

(Refer Slide Time 12:14)



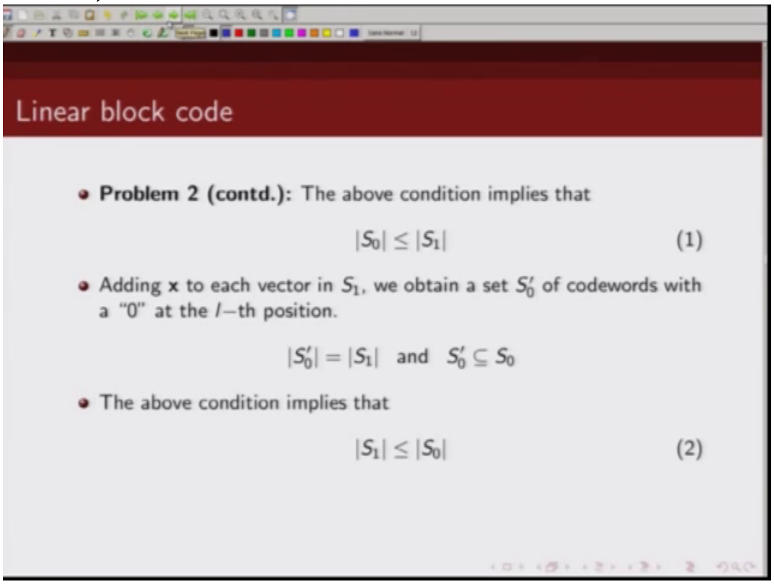
So from this we get this condition that set of codewords which has 0 at lth location is less than or equal to set of

(Refer Slide Time 12:28)



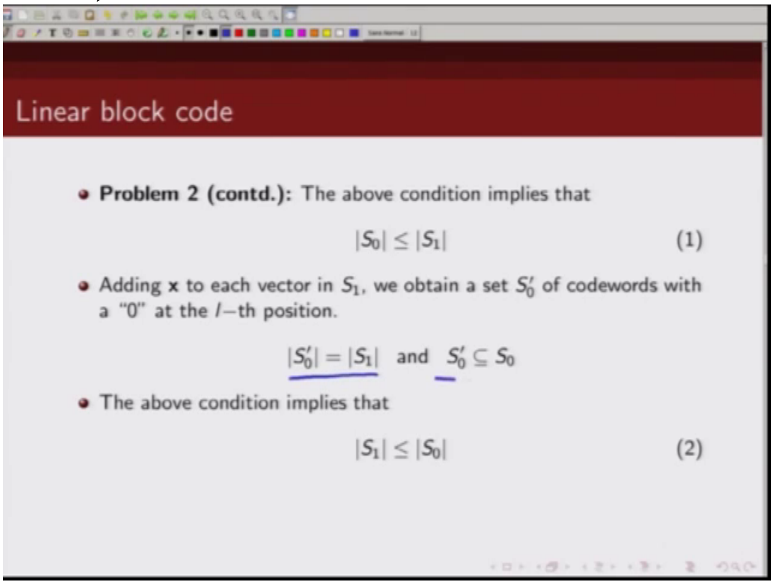
codewords which has 1 at lth location. Now add the same vector x which has 1 at lth location to all the elements in S 1. When we do that what we get is a new set of vectors which has 0 at lth location. We denote this set by S 0 prime. So S 0 prime is a set of codewords which are obtained by adding x to the set of vectors set of odd vectors which have 1 at lth location. So then we can write as set of vectors in S 0 prime is same as set of vectors in S 1 and since S 0 prime is a subset of S 0 what we can write then is,

(Refer Slide Time 13:26)



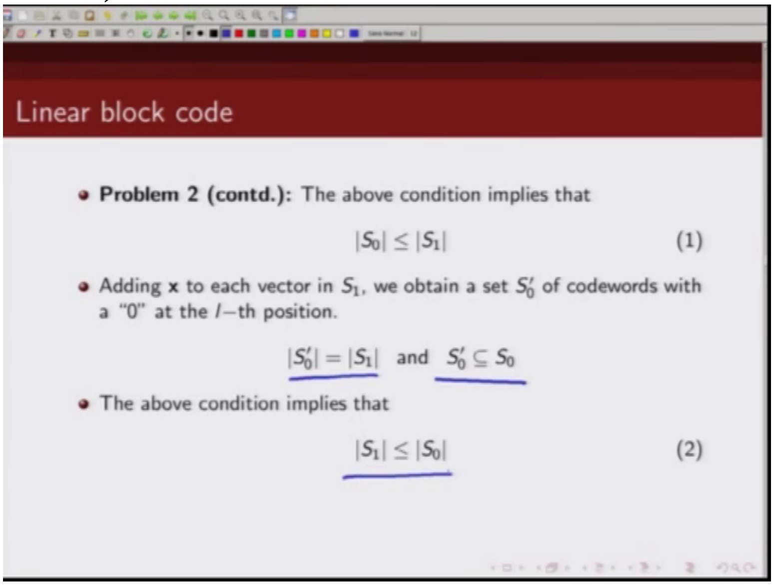
from this relation and this relation we

(Refer Slide Time 13:30)



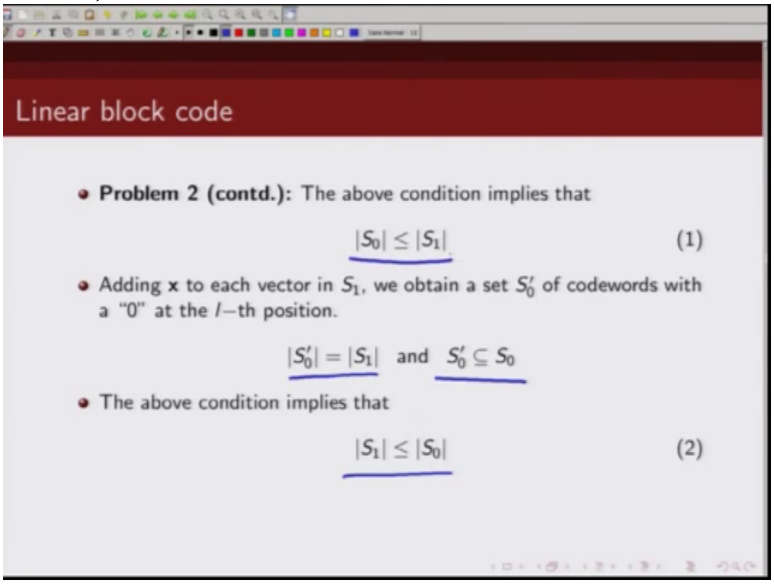
can write

(Refer Slide Time 13:32)



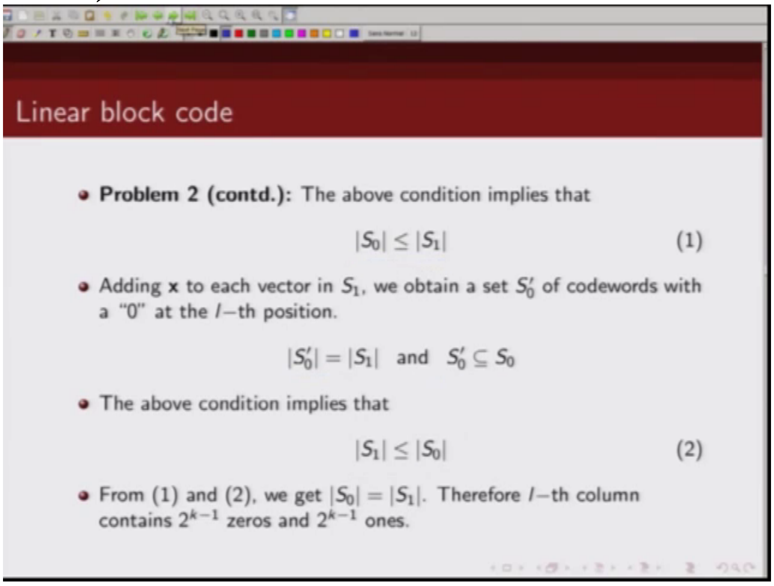
set of codewords which have 1 at lth location is less than set of codewords which has 0 at lth location. Now equation 1 and 2, they are going to be simultaneously

(Refer Slide Time 13:49)



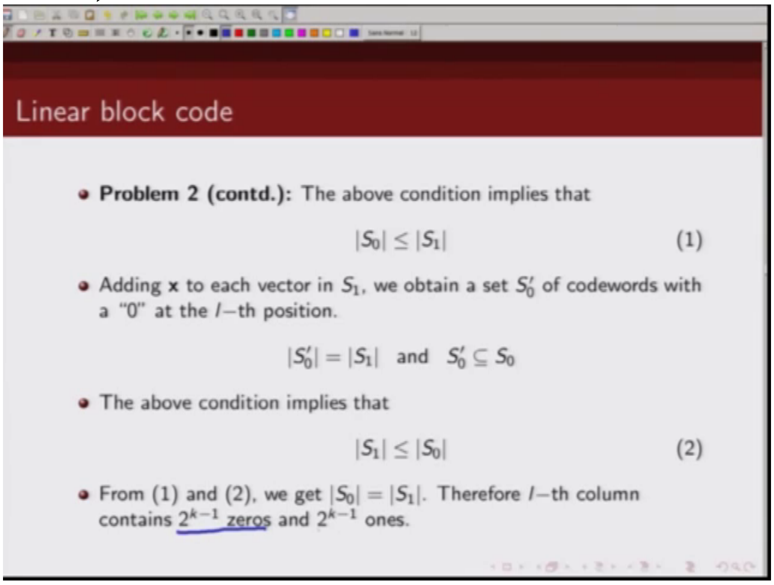
satisfied only when this is satisfied with equality. Then what it shows here is

(Refer Slide Time 13:57)



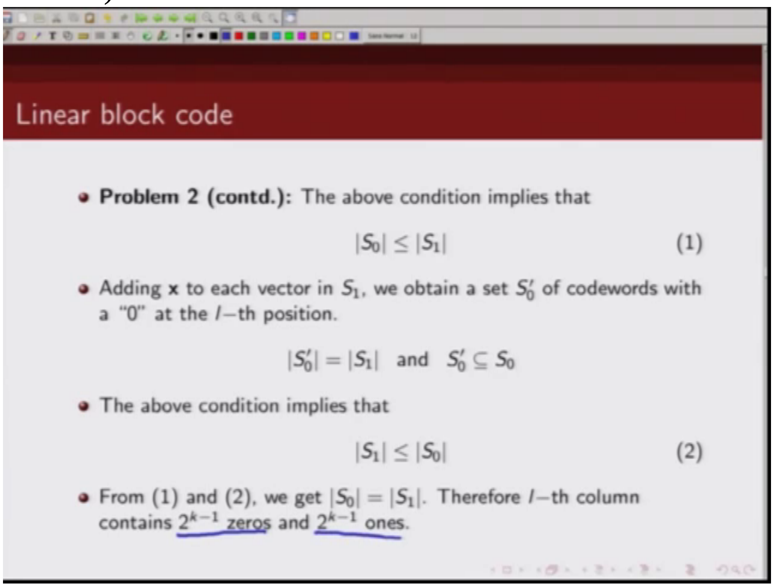
that at any lth location number of codewords which have 0 at lth location is same as number of codewords which have 1 at lth location. So basically each column will then have same number of 0s and

(Refer Slide Time 14:17)



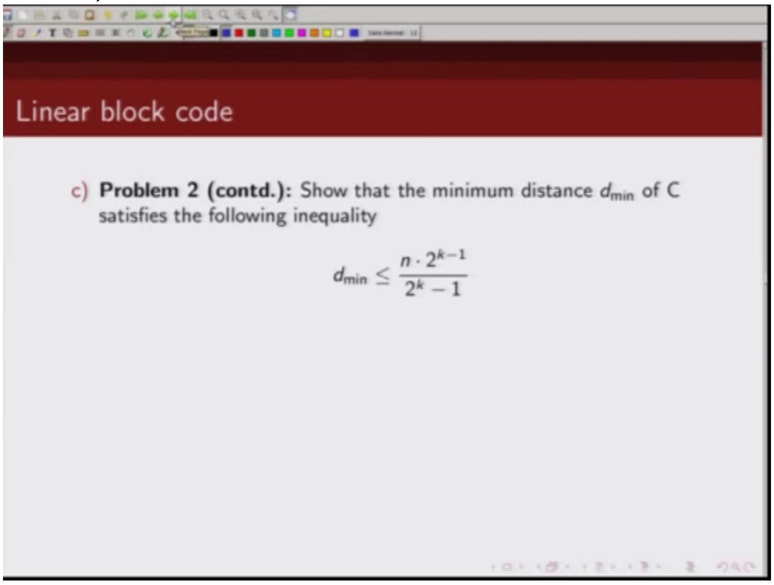
same number of 1s.

(Refer Slide Time 14:18)



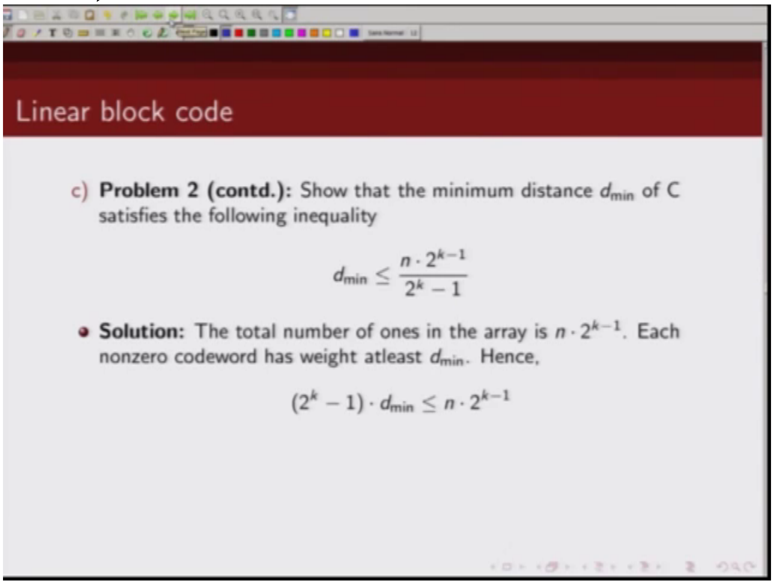
Now we

(Refer Slide Time 14:20)



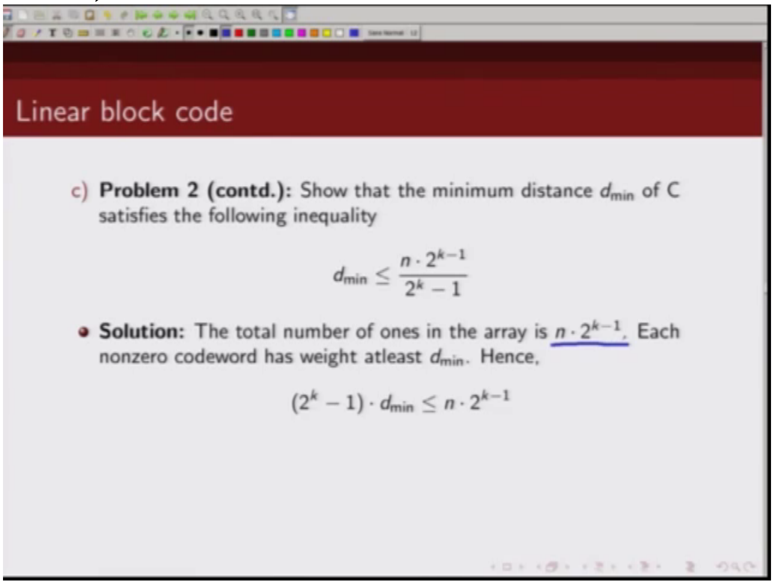
prove another result. We show that minimum distance of the code is upper bounded by this quantity and to prove this result we are just going to use the result we just proved in the previous section. So

(Refer Slide Time 14:41)



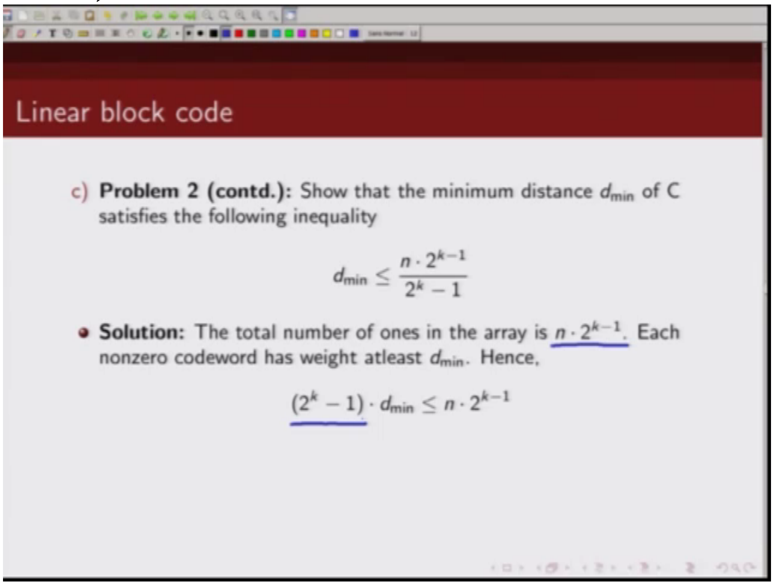
in the previous section what we did was we arranged these 2 k codewords in an array two k cross n array and we showed that each column of this array has 2 raised to power k minus 1 1s and 2 raised to k minus 1 0s. So in this whole array which has n columns total number of 1s is given by this,

(Refer Slide Time 15:11)



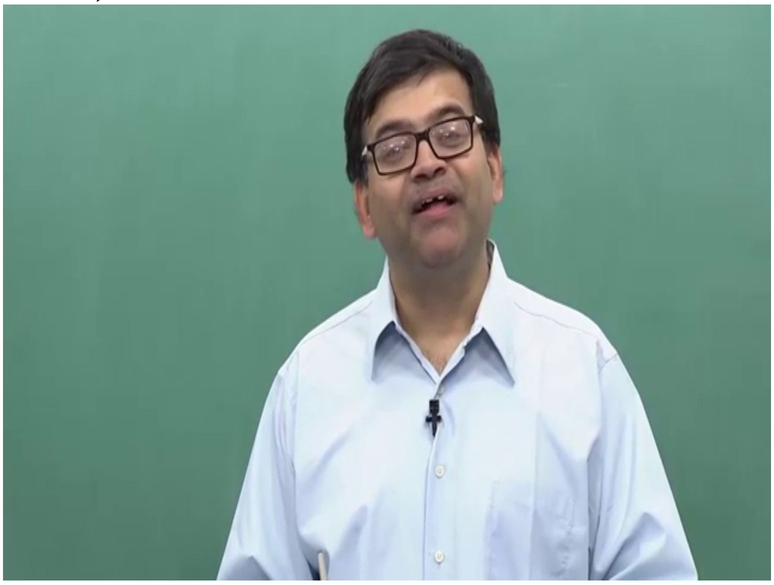
n times 2 raised to power k minus 1. Now since each non-zero codeword will have minimum distance at least d min and how many total codewords we have 2 raised to power k one of them is all zero codeword so how many non-zero codewords we have, that is given by 2 k minus 1

(Refer Slide Time 15:37)



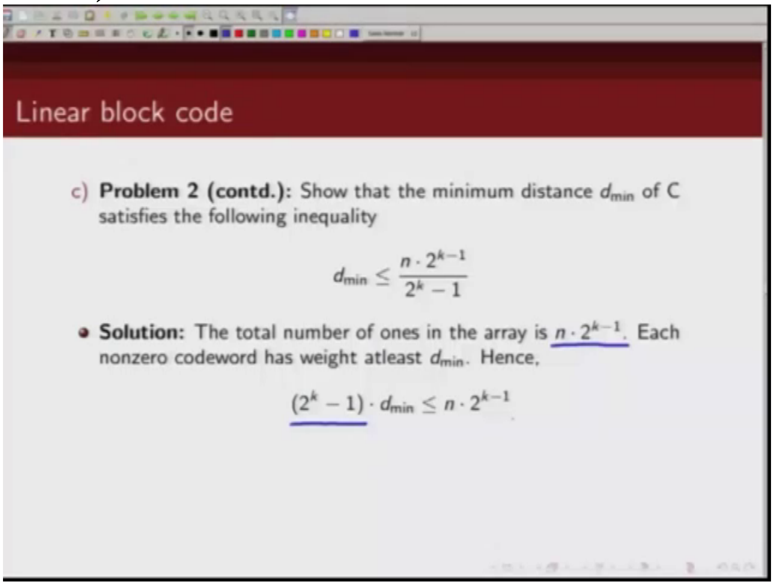
and each of these non-zero codewords have minimum distance at least d min so total number of non-zero codewords multiplied by d min must be

(Refer Slide Time 15:53)



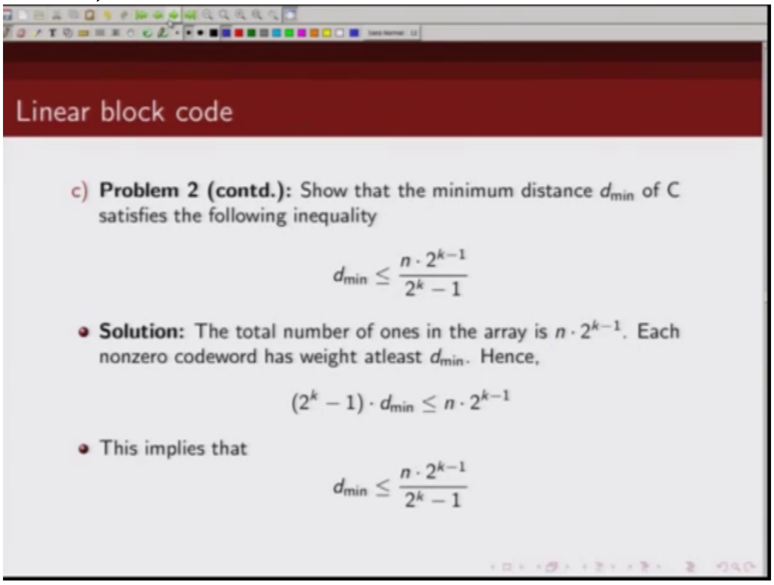
less than equal to total number of 1s

(Refer Slide Time 15:56)



in this code array which is given by n times 2 k minus 1, 2 raised to power k minus 1. So from this relation then we can then write

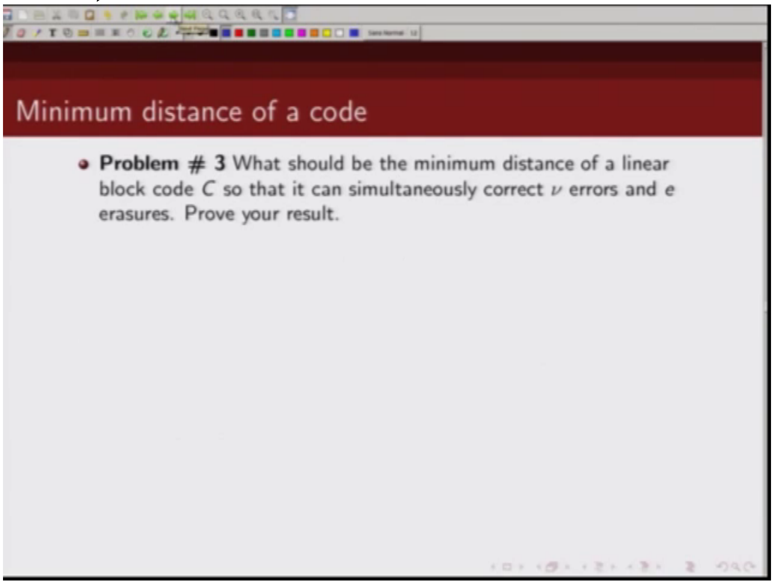
(Refer Slide Time 16:09)



that minimum distance of a code is upper bounded by this relationship.

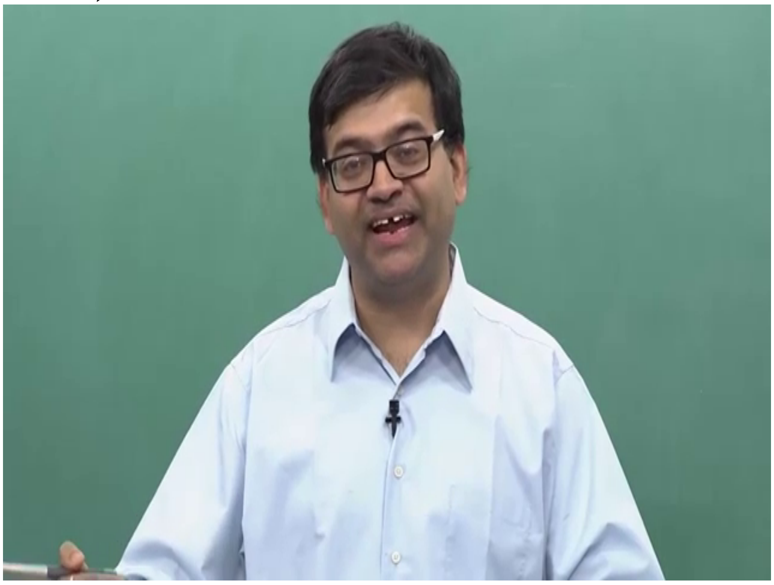
So

(Refer Slide Time 16:16)



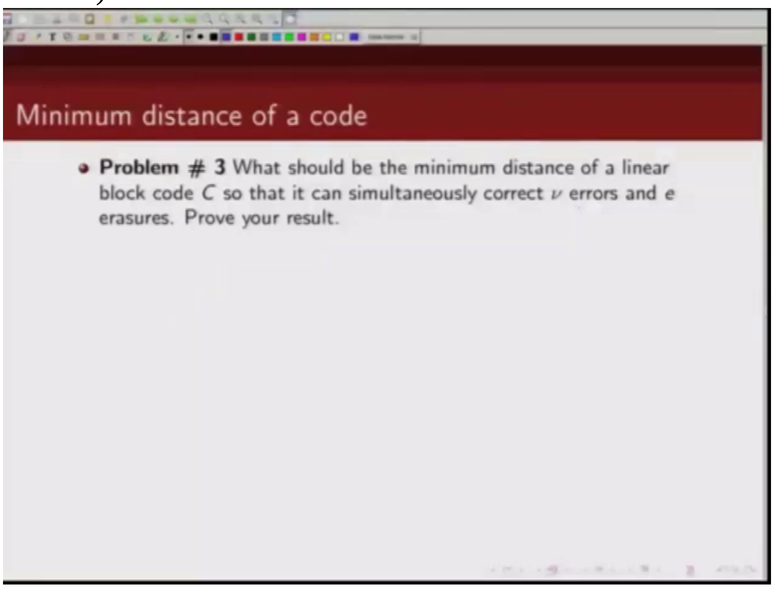
next problem that we will look at is what is a minimum distance of linear block C that can simultaneously correct mu errors and e erasures? Now just recall what do we mean by error correction and error

(Refer Slide Time 16:36)



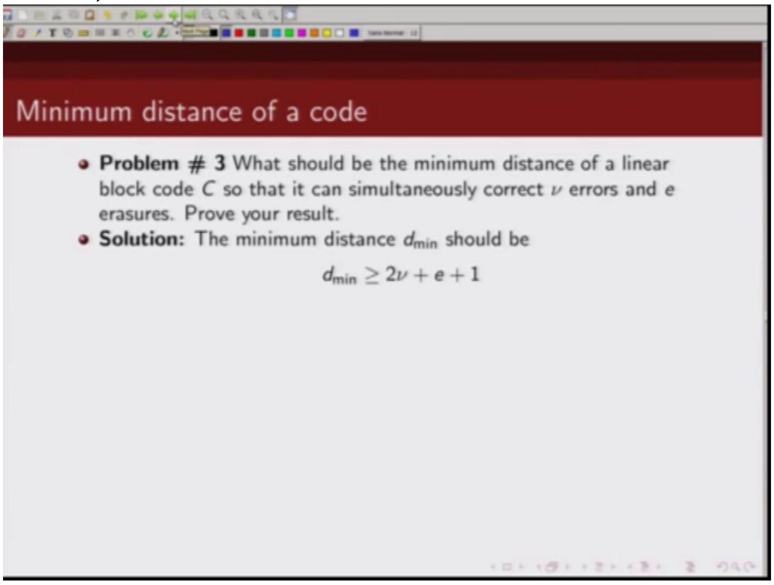
erasure correction. Basically, so erasure is basically, some of the bits are getting erased. So you send n bits. If e bits are getting erased what you are receiving is n minus e bits. And error correction you are familiar with, basically you want to correct errors that have happened in so many bit locations. So the question is

(Refer Slide Time 17:02)



what should be the minimum distance of a linear block code that can simultaneously correct mu errors as well as e erasures?

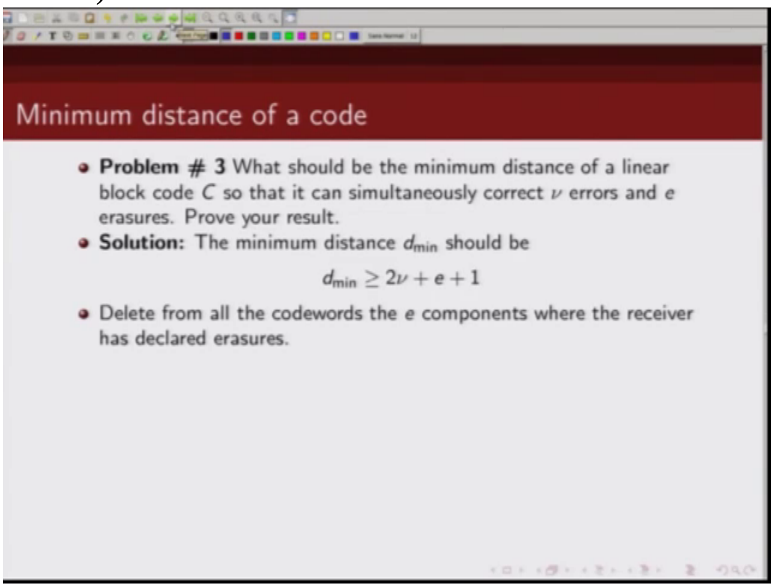
(Refer Slide Time 17:15)



Now if the minimum distance of the code is at least two mu plus e plus 1 then it can simultaneously correct mu errors and e erasures. We are going to next prove this result.

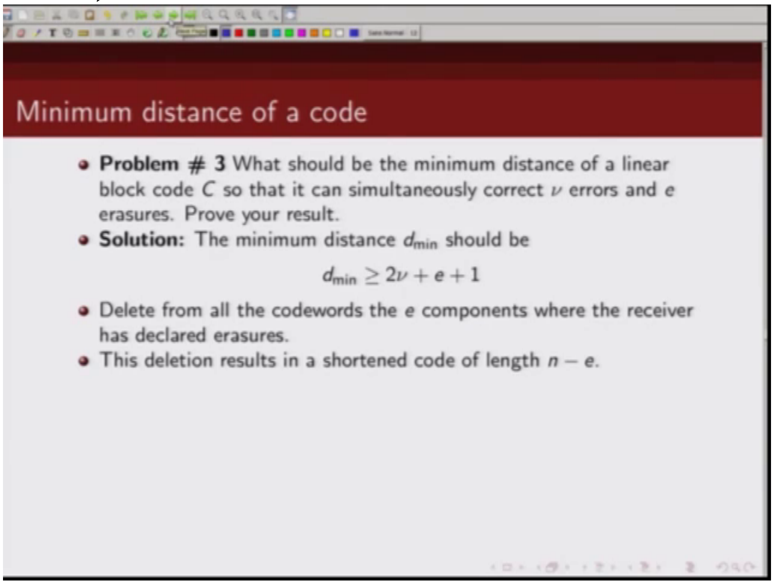
So

(Refer Slide Time 17:31)



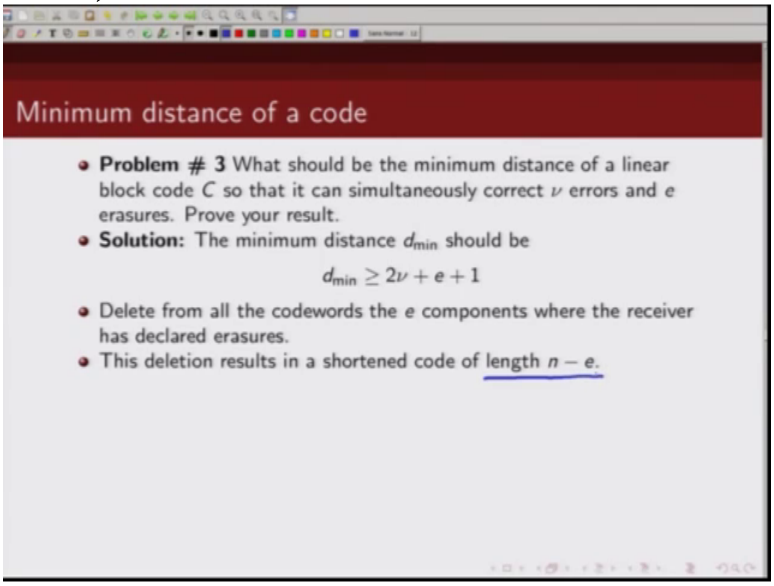
delete from all codewords e components which got erased. If we delete these e components what we are left is n minus e

(Refer Slide Time 17:46)



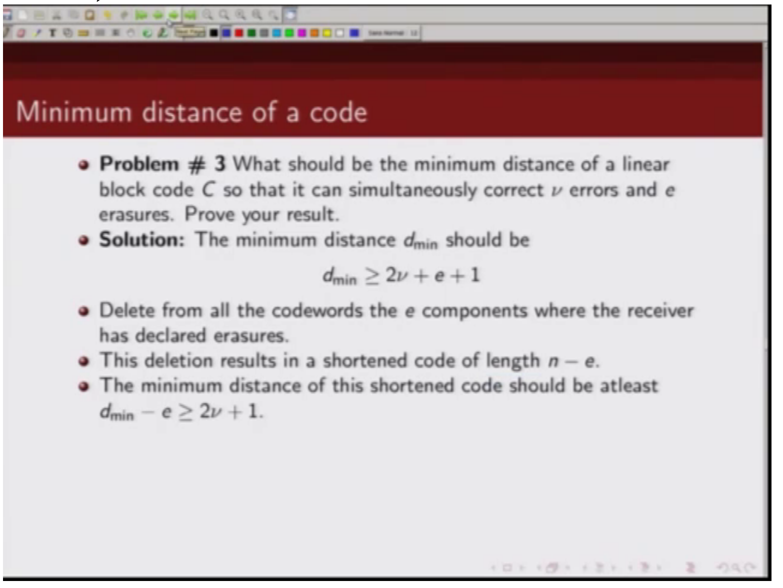
length shortened codeword. So this deletion of e component results in a shortened code of length n minus e.

(Refer Slide Time 17:56)



Now we

(Refer Slide Time 17:59)

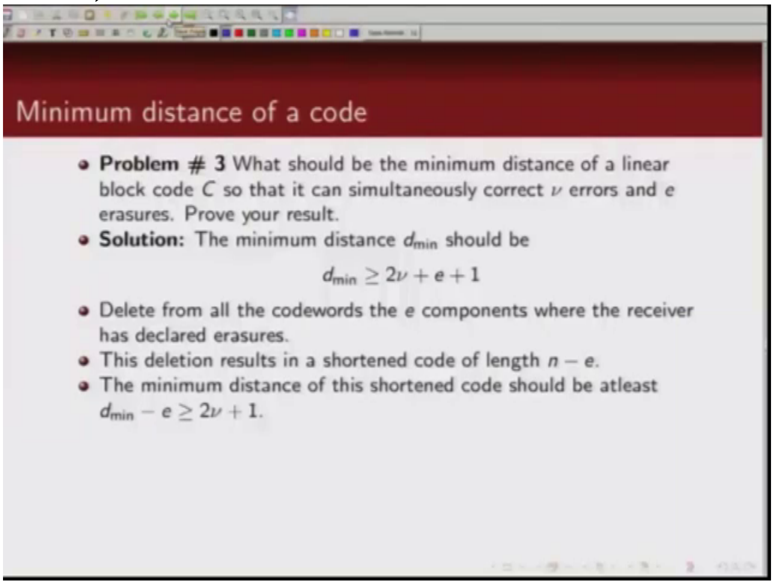


know that if we want to correct t errors, what should be minimum distance of the code? It should be at least

(Refer Slide Time 18:06)

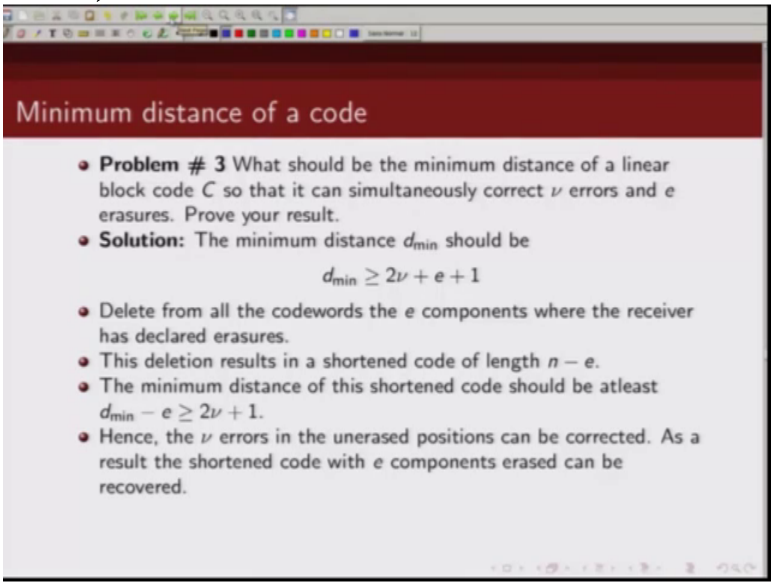
2 t plus 1. So this code

(Refer Slide Time 18:11)



basically, if we want to correct mu errors the minimum distance of the code after these e erasures

(Refer Slide Time 18:48)

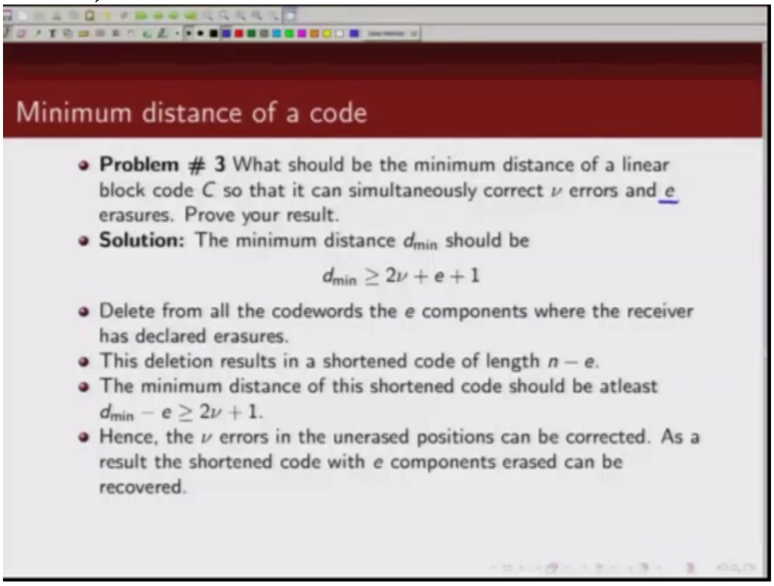


should be greater than equal to two mu plus 1. So if minimum distance of the code after these e erasures, the minimum distance is still larger than two mu plus 1, then this code can correct mu errors. So we want our minimum distance of the code to be at least 2 mu plus e plus 1. Now since these mu errors are in, in the unerased positions can be corrected if this condition holds so as a result basically we would be able to correct mu errors.

(Refer Slide Time 19:08)

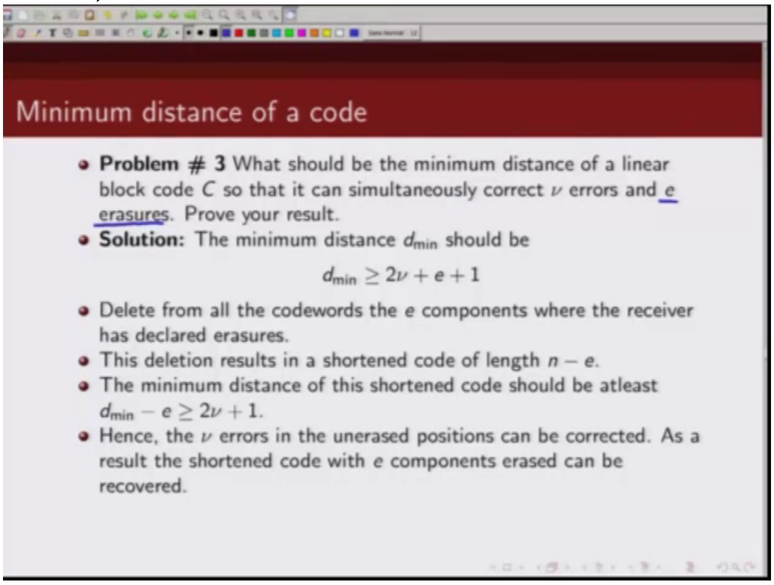
Now remember we have to simultaneously

(Refer Slide Time 19:11)



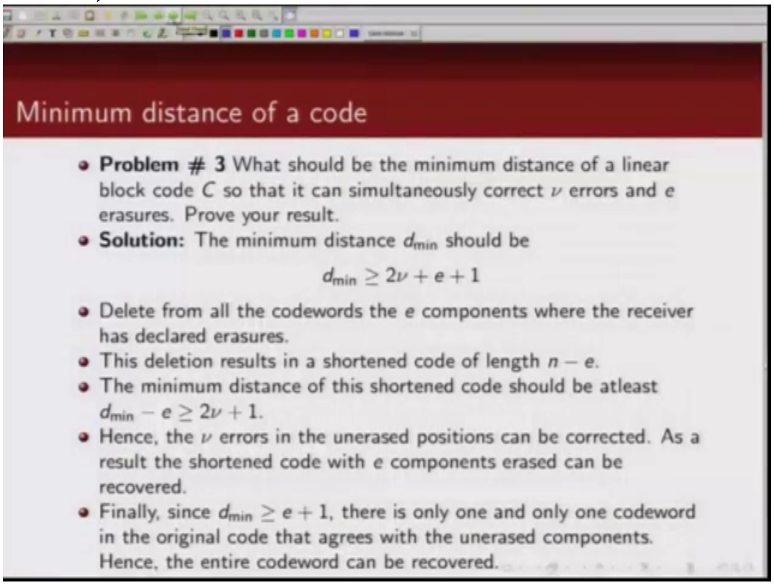
also be

(Refer Slide Time 19:14)



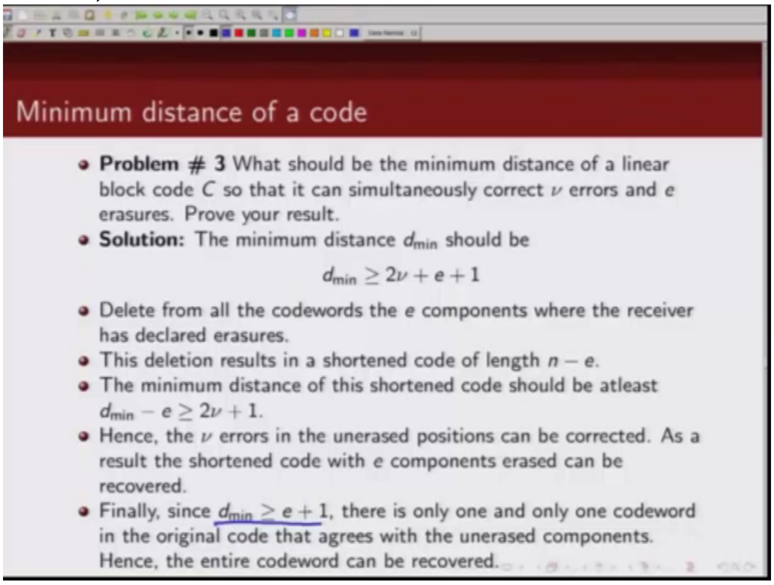
able to basically, it would be not only simultaneously correct mu errors but we have to correct e erasures also. Now what is the condition on minimum distance such that e erasures can also be corrected? The minimum distance of the code should be at least greater than number of erasures plus 1. So if the minimum distance of the code

(Refer Slide Time 19:40)



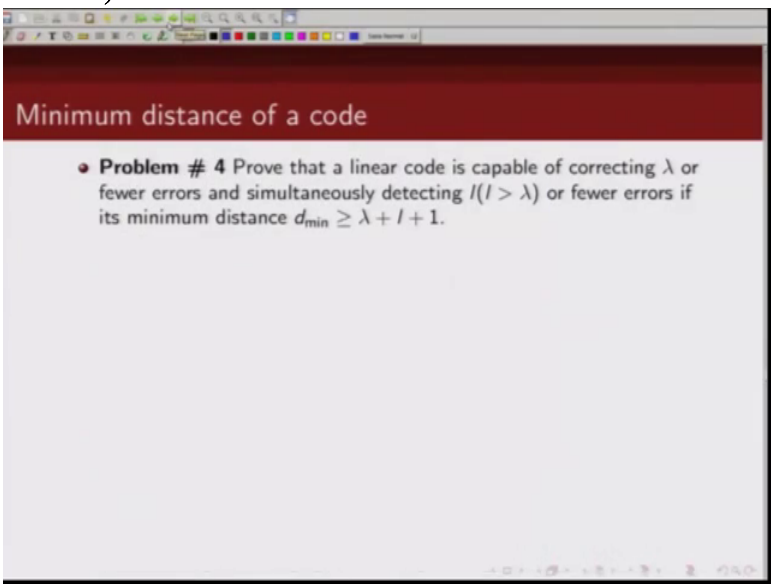
is greater than e plus 1

(Refer Slide Time 19:43)



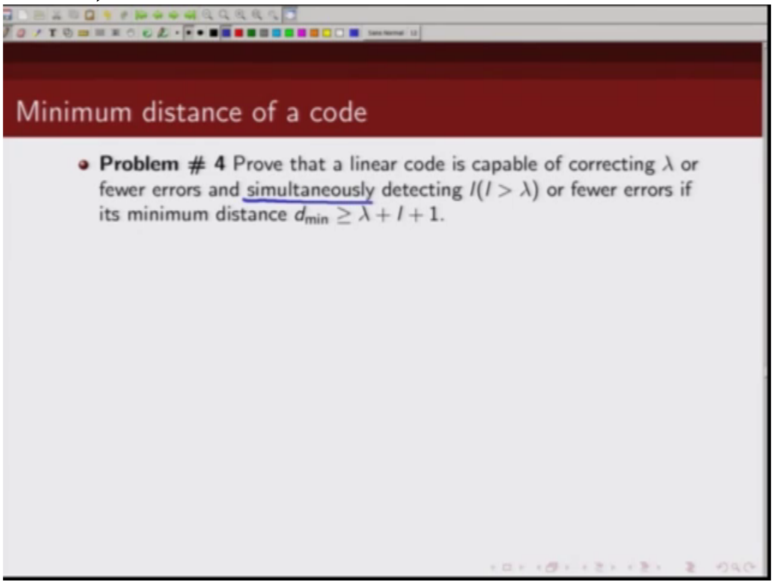
then there is only one codeword in the original code that maps to the shortened code. So as long as minimum distance of the code is greater than e plus 1, there is only one and one codeword in the original code that agrees with the unerased component. So there is as long as minimum distance of the code is greater than e plus 1, there is only one code that maps from erased shortened code to the original code. And since in this case the d minimum is already 2 mu plus e plus 1 which is greater than e plus 1, this code would be able to correct e erasures as well. So if we choose our minimum distance of the code to be greater than equal to two mu plus e plus 1, it would be able to correct mu errors as well as e erasures.

(Refer Slide Time 20:53)



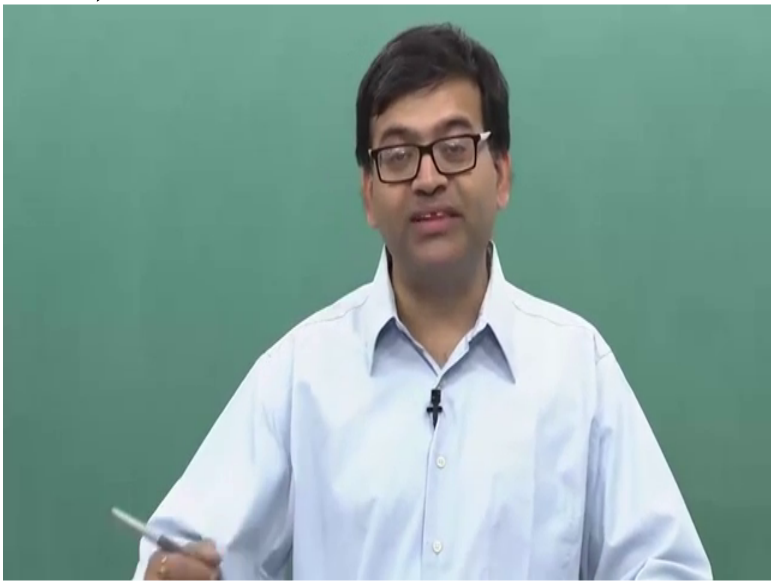
The next problem that we are going to solve is as follows. Prove that a linear block code is capable of correcting lambda or fewer errors and simultaneously detecting l where l is greater than lambda or fewer errors if the minimum distance of the code is at least lambda plus l plus 1. Please pay attention to the word simultaneously.

(Refer Slide Time 21:28)



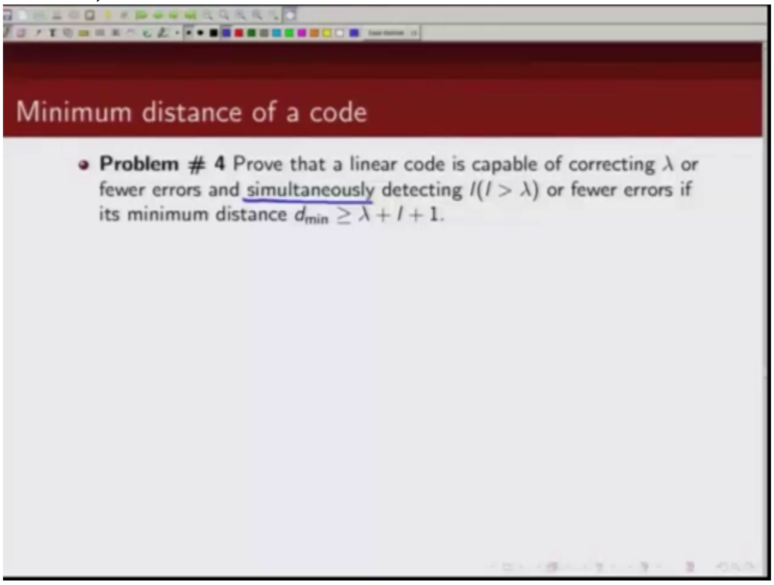
So we want not only to correct mu errors, along with that we should be able

(Refer Slide Time 21:34)



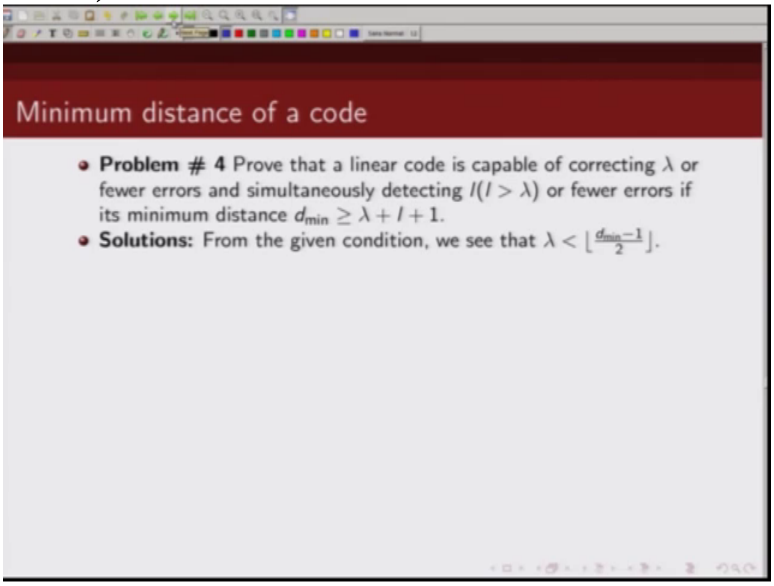
to detect l errors as well. That's what we mean by

(Refer Slide Time 21:40)



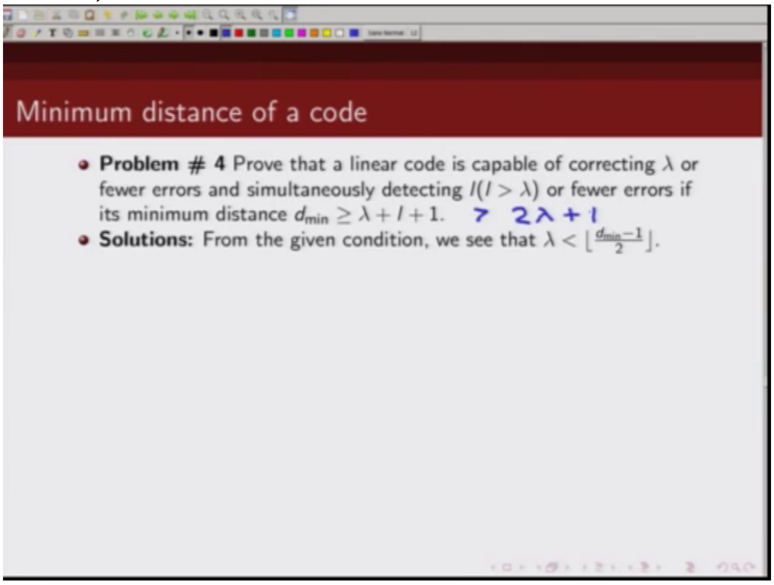
simultaneous error detection and correction. So let's prove this result. Now note

(Refer Slide Time 21:48)



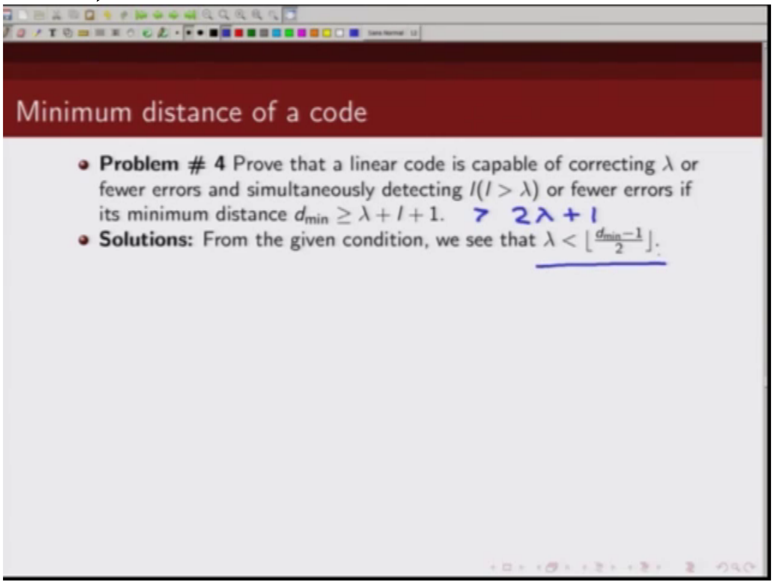
lambda is less than l. So if minimum distance is lambda plus l plus 1, this is basically greater than 2 lambda plus 1.

(Refer Slide Time 22:05)



And if the minimum distance is greater than 2 lambda plus 1, it would be able to correct lambda errors. So from this given condition that d min is at least lambda plus l plus 1 where l is greater than lambda we know

(Refer Slide Time 22:25)

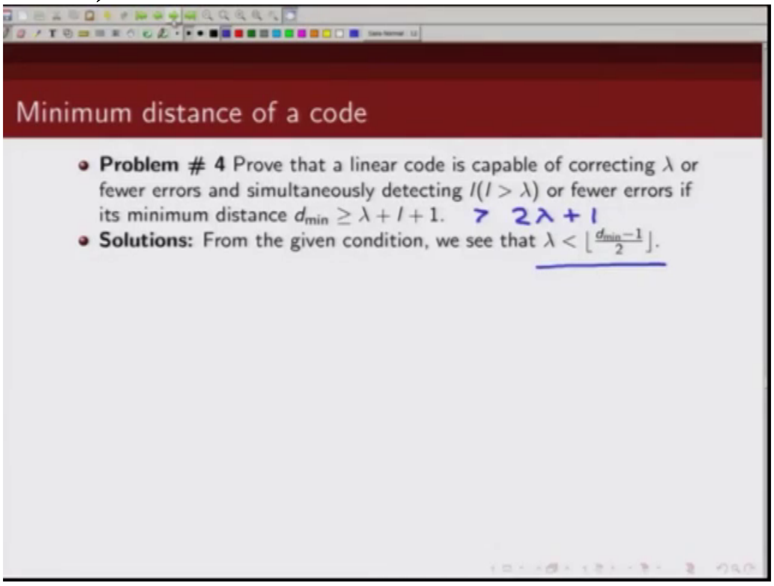


that the minimum distance is greater than 2 lambda plus 1.

(Refer Slide Time 22:30)

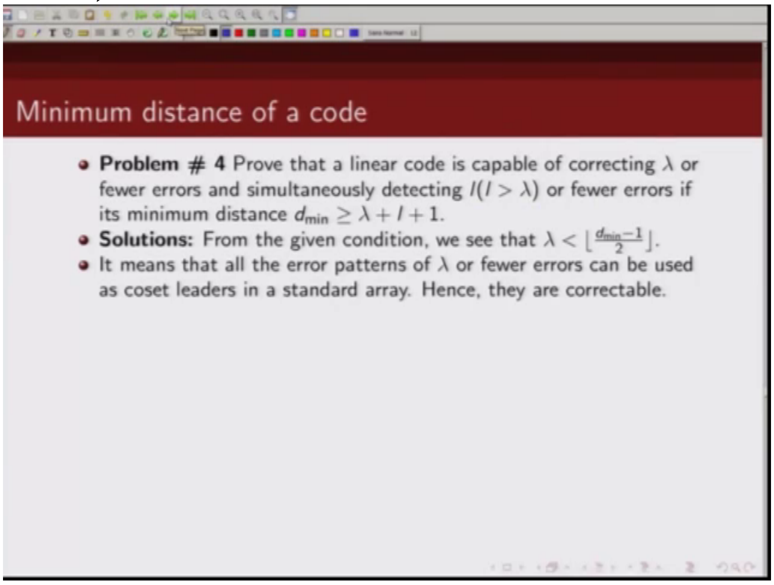
So it should be able to correct lambda errors.

(Refer Slide Time 22:34)



Now note we want to,

(Refer Slide Time 22:36)

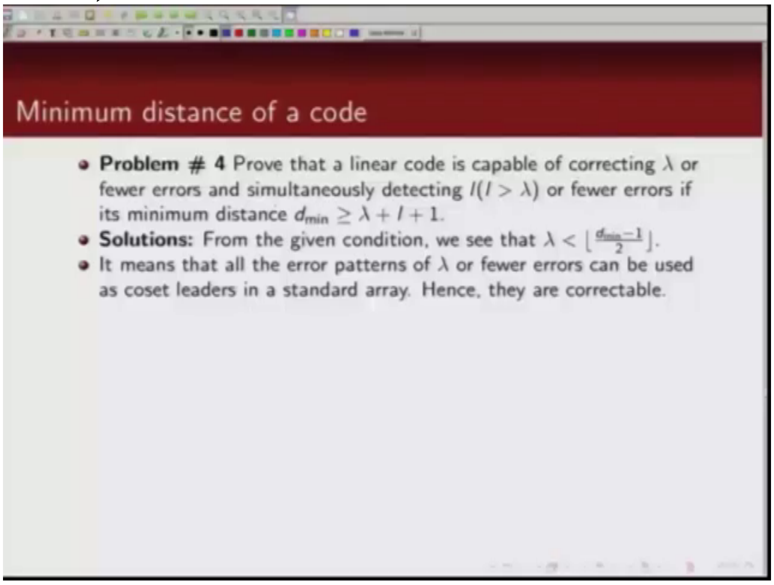


in addition to correcting lambda or fewer errors, we also want to simultaneously detect l errors. Now if we want to simultaneously detect those l or fewer errors we have to ensure that

(Refer Slide Time 22:53)

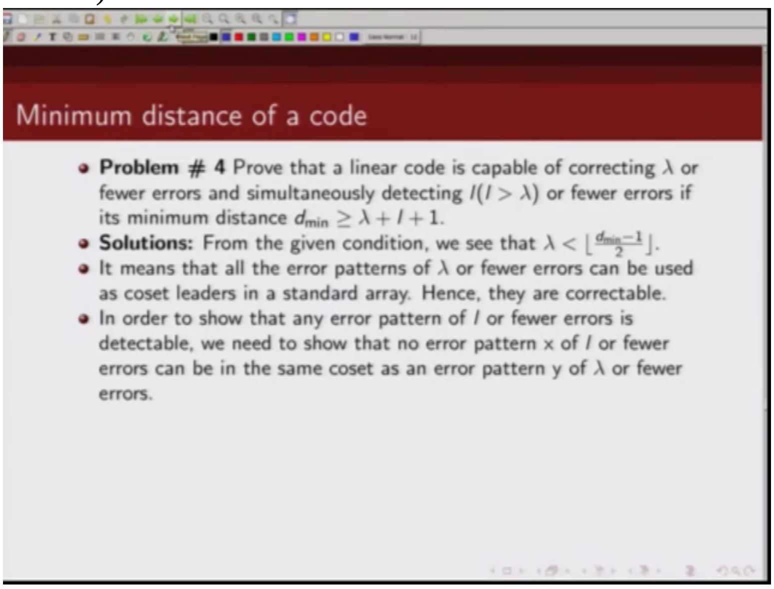
those error patterns of weight l or less are not in the same coset as the error patterns that we are

(Refer Slide Time 23:02)



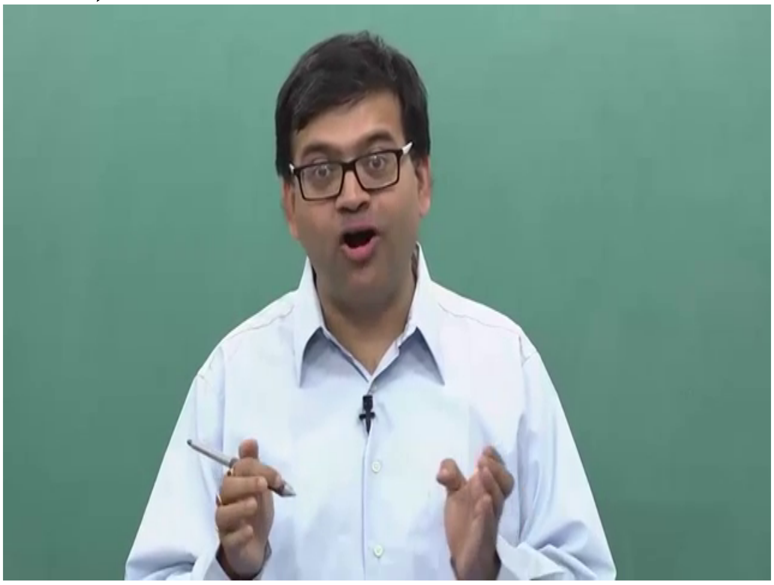
trying to correct. Now since lambda errors can be corrected we can put all error patterns of lambda or fewer errors as coset leader in our standard array and they can be correctable.

(Refer Slide Time 23:22)



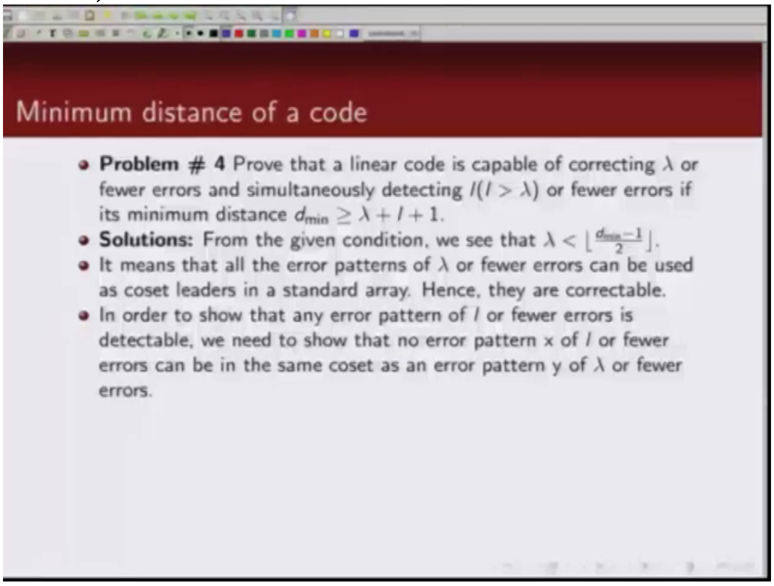
Next, to simultaneously detect l errors we have to show that none of these error patterns of weight l or less

(Refer Slide Time 23:32)



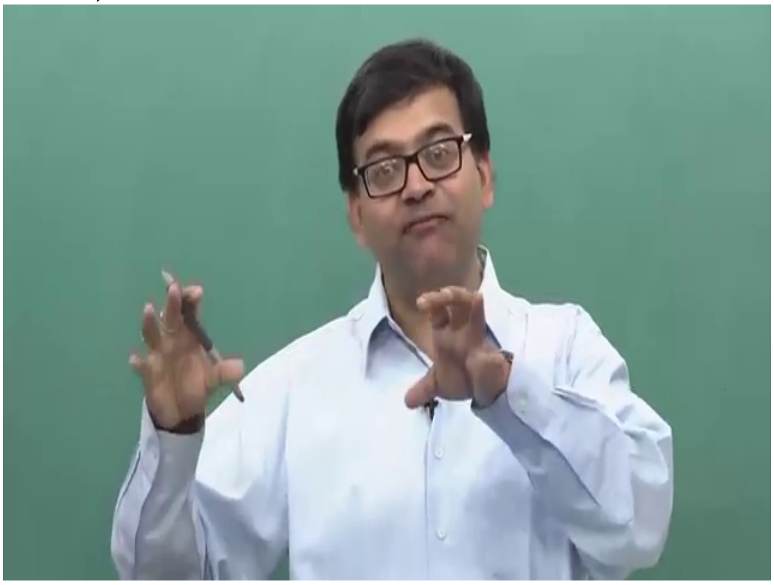
are in the same coset as these error

(Refer Slide Time 23:38)



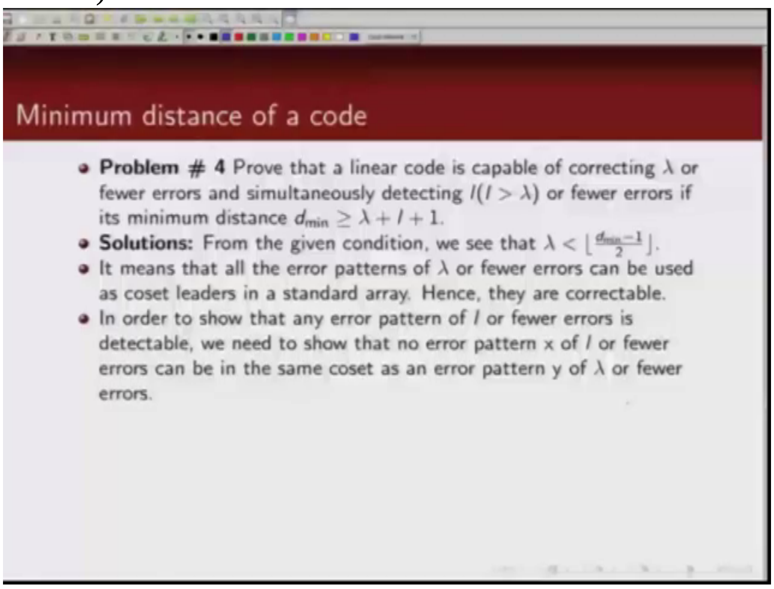
patterns of lambda or less error. So we need to show that no error pattern x of length l or fewer are in the same coset as error pattern y of lambda or fewer errors. If they are in the same coset, because we are using those coset leaders for error correction, we would not be able to detect those error patterns. So it is important that those error patterns of weight l or less,

(Refer Slide Time 24:09)



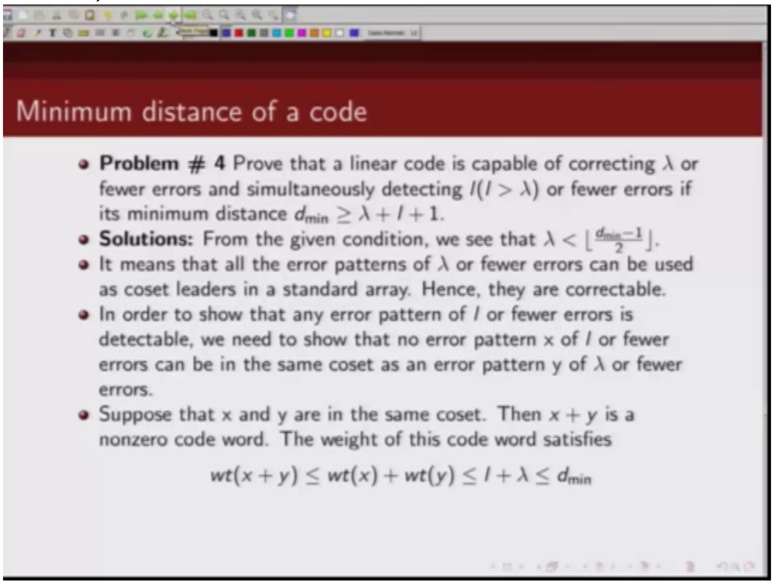
if we want to detect them, they should not be in the same coset as the correctable error patterns.

(Refer Slide Time 24:17)



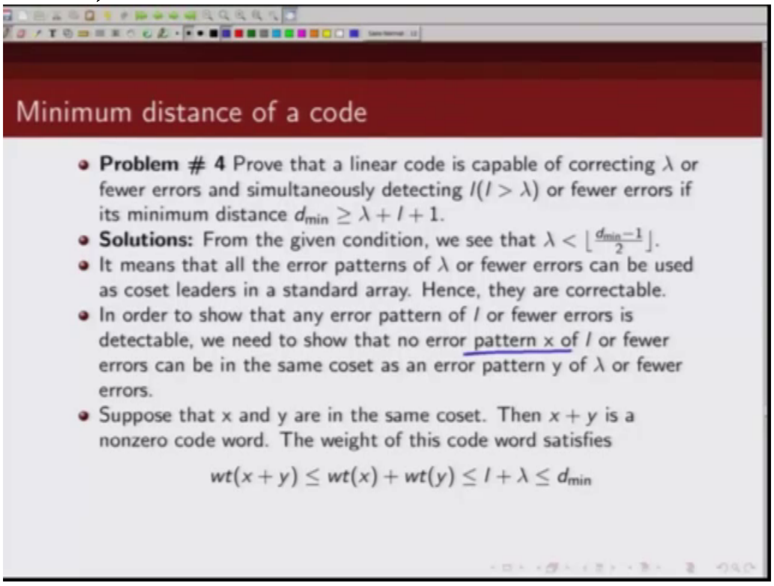
So

(Refer Slide Time 24:19)



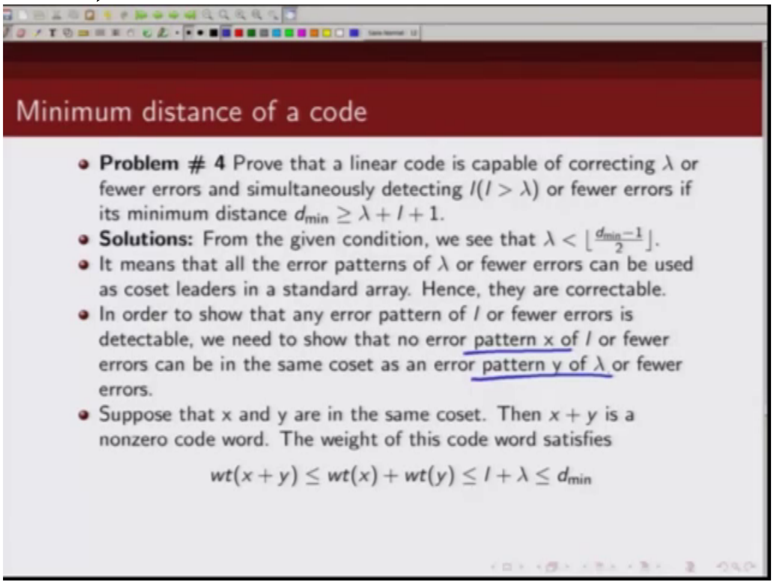
So we are now going to use method of contradiction to show that it is not possible to have these error pattern x of

(Refer Slide Time 24:29)



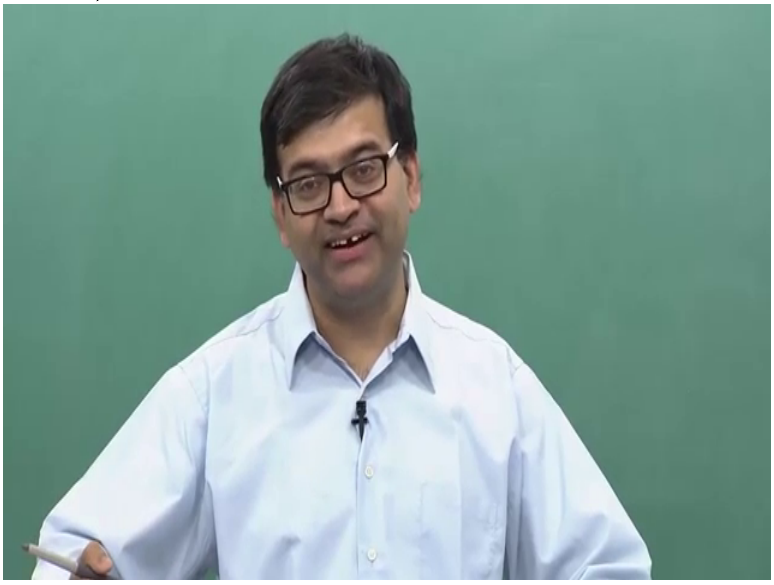
l or fewer errors in the same coset as these error patterns y

(Refer Slide Time 24:35)

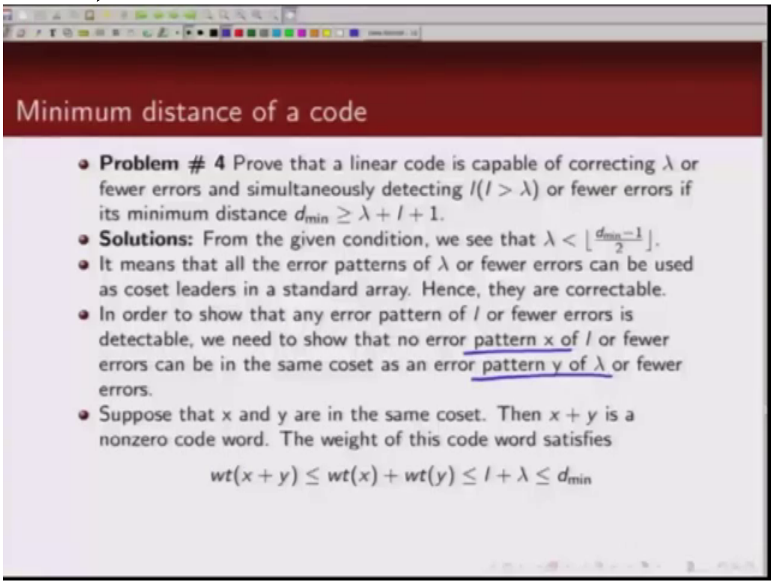


of lambda or fewer errors which we are trying to correct. So how does this method of contradiction work? We will first assume that they are in the same coset and then we will show that this is not possible. Hence our assumption that they are in the same coset is wrong.

(Refer Slide Time 24:52)



(Refer Slide Time 24:54)

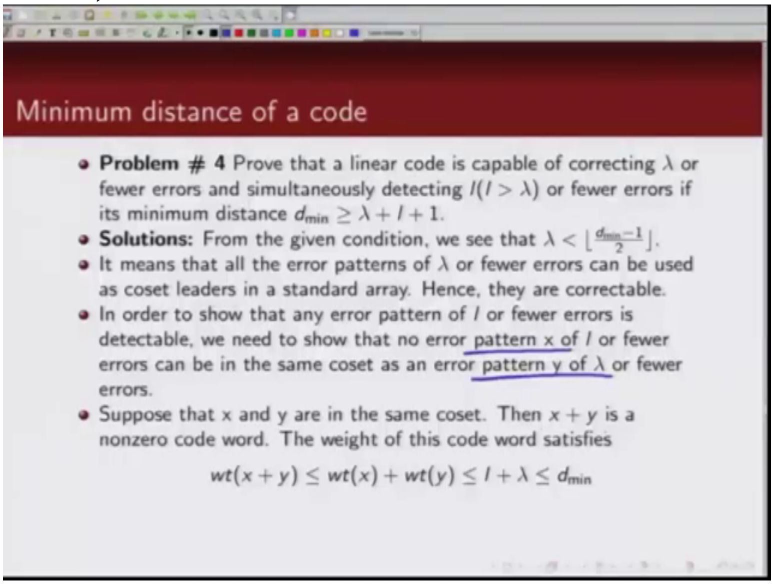


So we start our proof by saying these error pattern x of weight l or less and error pattern y of weight lambda or less, they are in the same coset. Now if x and y are in the same coset we know from our

(Refer Slide Time 25:16)

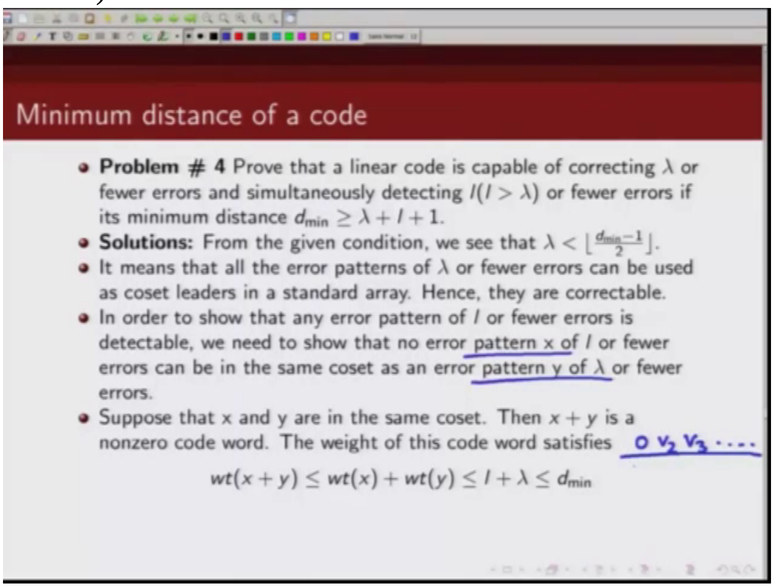
standard array that x plus y

(Refer Slide Time 25:19)



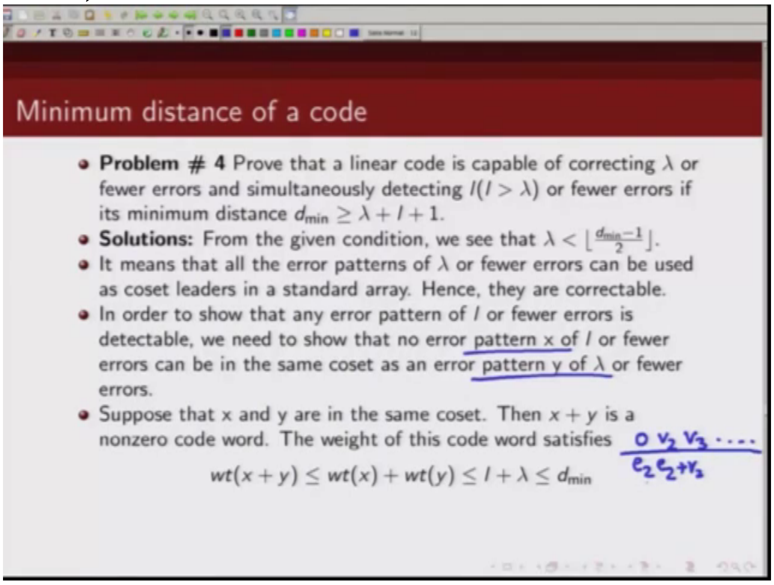
should be a non-zero codeword. If you recall the entries of standard array we have in the first column an all zero codeword and then we had other codeword v 2, v 3 ... and then what we had was error pattern

(Refer Slide Time 25:37)



e 2 and then we had basically this was e 2 plus v 2 like that we had and if you add any two elements of a coset or a row

Refer Slide Time 25:52)

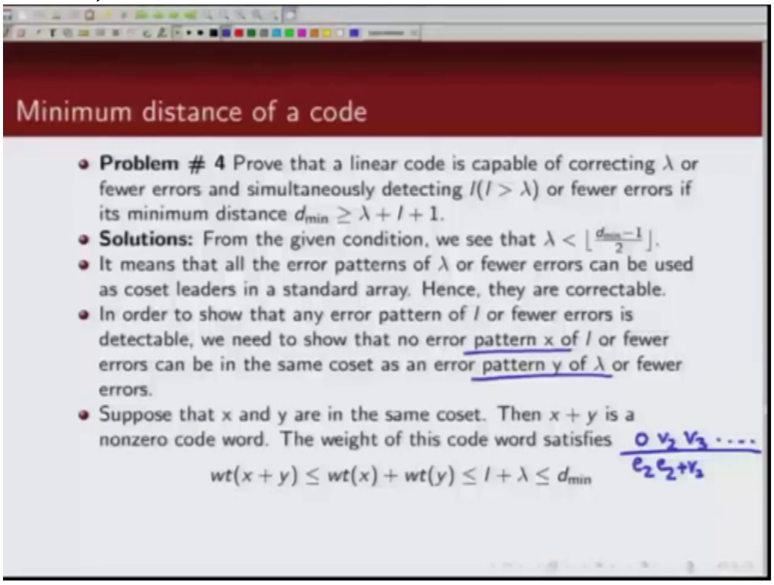


what you will notice is sum of them is a valid codeword. So if x and y are in the same coset, x plus y must be a non-zero codeword. Now let's look at what is the weight of x plus y. So weight of x plus y is less than equal to weight of x plus weight of y because it is possible that there are some common elements between x and y,

(Refer Slide Time 26:25)

that's why the weight of x plus y

(Refer Slide Time 26:27)

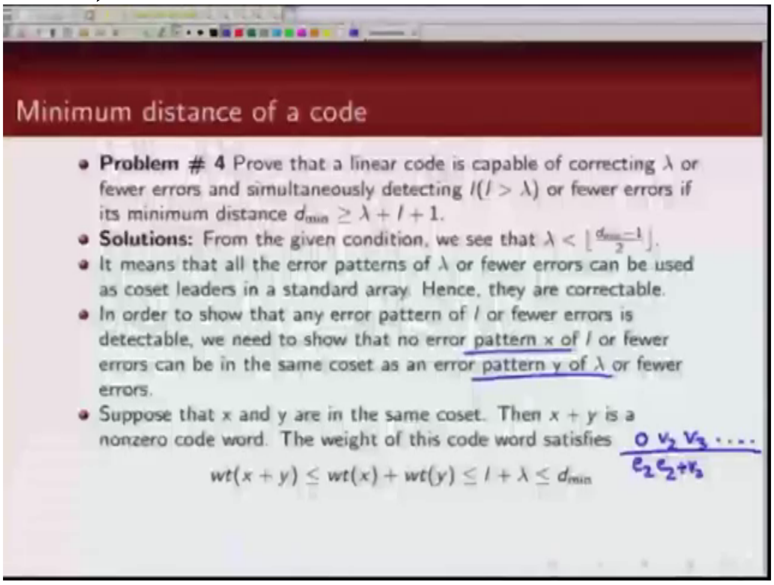


is less than equal to weight of x plus weight of y. And what is weight of x? x are the error pattern of weight l or less. So the maximum weight of x is l. Similarly maximum weight of y is lambda. So weight of x plus y is then less than equal to lambda plus l. And what is the minimum distance? Minimum distance code is at least lambda plus l plus 1. So weight of x plus y is then less than d min. So what we have shown is the weight of x plus y, x plus y should have been a codeword, is a codeword if they are in the same coset. If x and y are in the same coset, x plus y is a valid non-zero codeword but what we have shown here is weight of x plus y is less than d min so x plus y is a valid codeword. Its minimum weight

(Refer Slide Time 27:38)

should be at least d min. So

(Refer Slide Time 27:42)

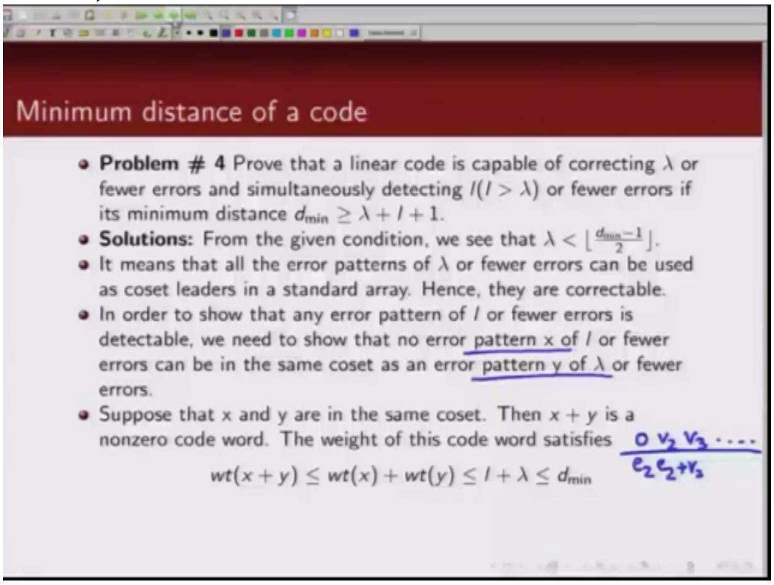


from here, basically what we get is it is not possible to have x plus y in the same coset. Because if they were in the same coset x plus y would have been a valid codeword and its weight of x plus y should have been more than d min but here in this case it is coming out to be less than d min. Hence our assumption that x and y are in the same coset

(Refer Slide Time 28:13)

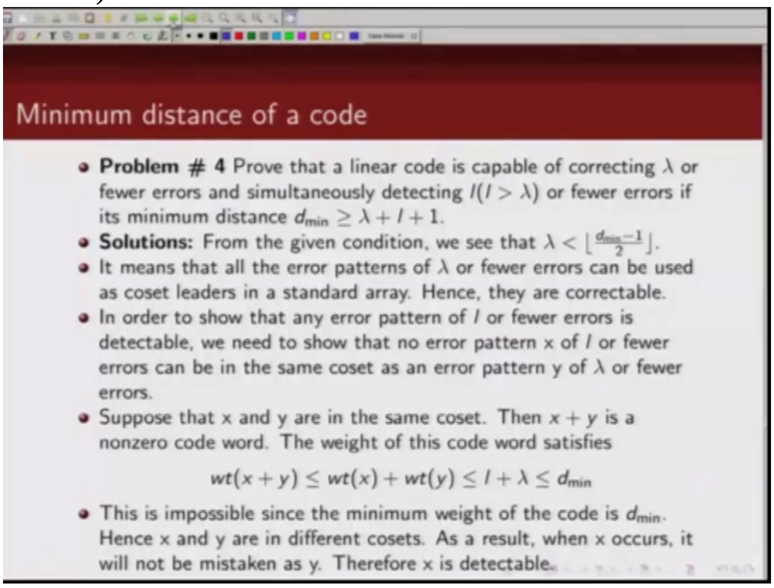
is wrong. Now if

(Refer Slide Time 28:17)



x and y are not in the same coset

(Refer Slide Time 28:20)

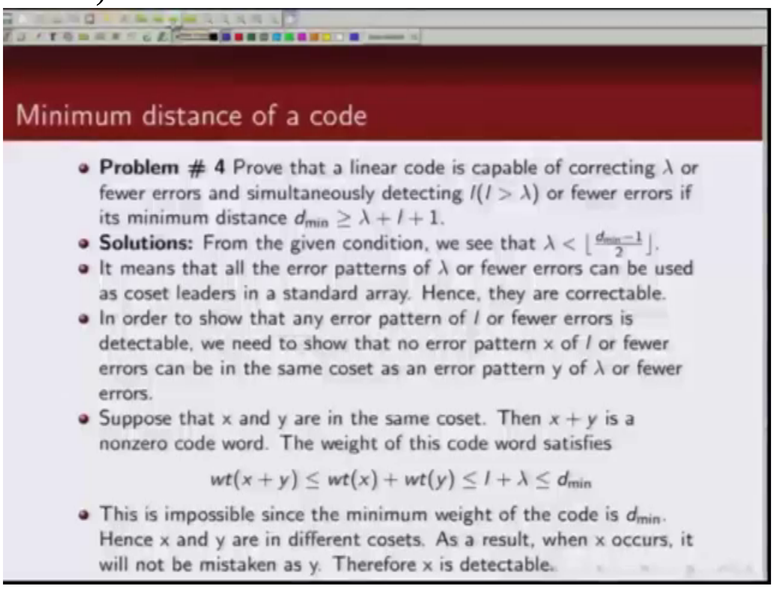


then we can always put those error patterns of

(Refer Slide Time 28:26)

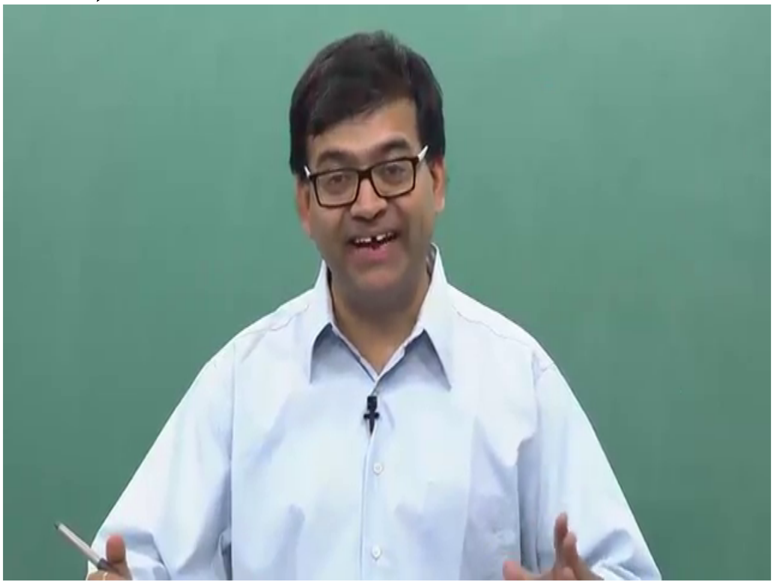
y and x in different cosets and hence we can simultaneously

(Refer Slide Time 28:33)



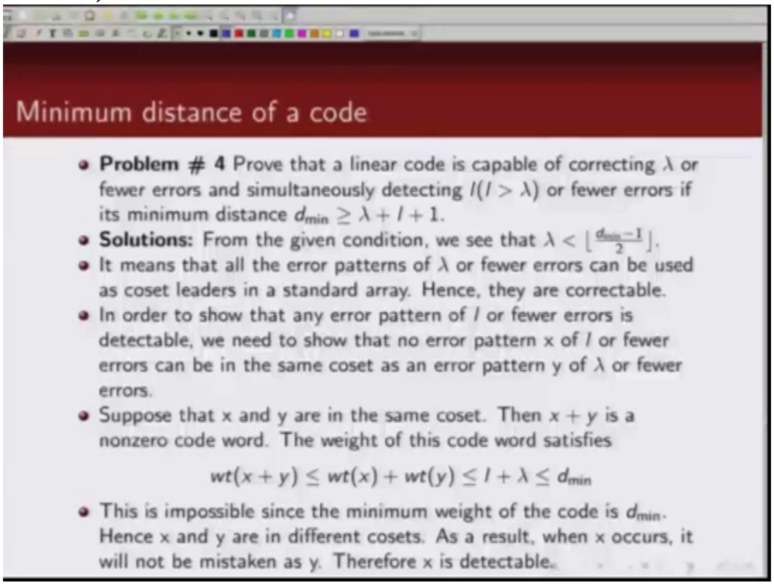
detect and correct errors. So we can simultaneously correct lambda errors while detecting also l errors, Ok. So again to recap, basically we prove this result by showing that if you want to simultaneously correct and detect errors

(Refer Slide Time 28:56)



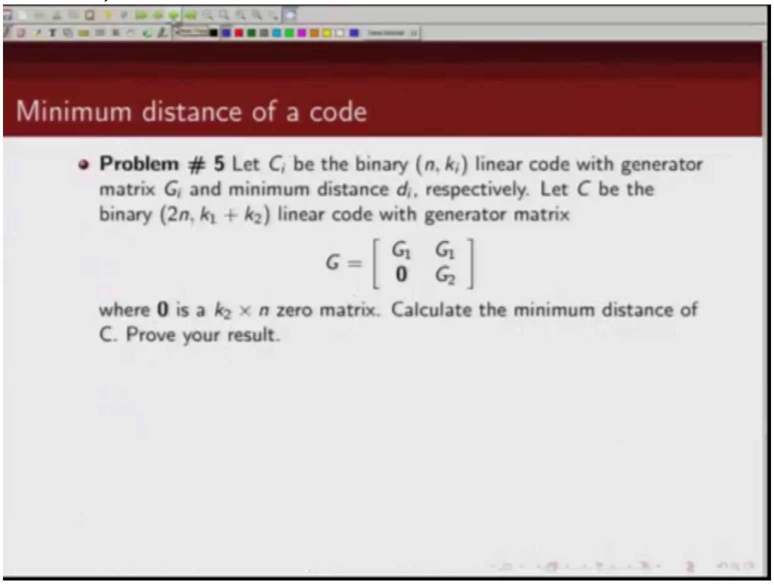
those error patterns should be in the different cosets and hence we can

(Refer Slide Time 29:05)



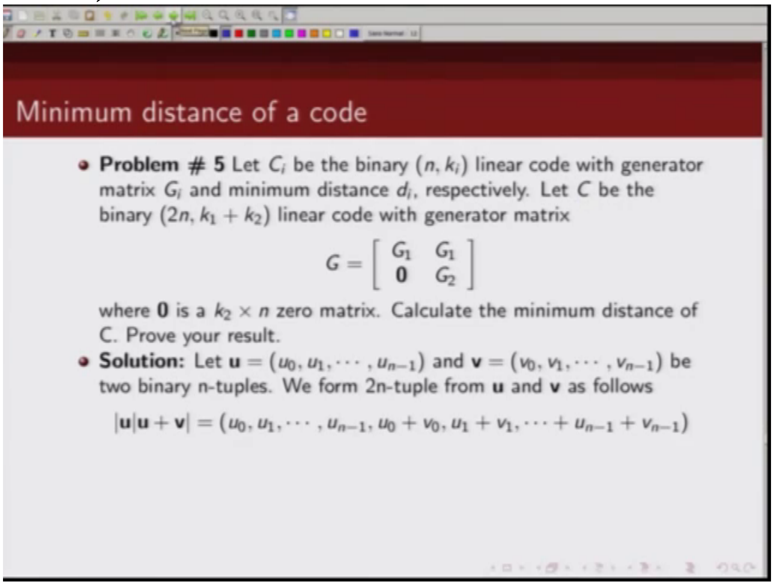
simultaneously correct and detect those error patterns.

(Refer Slide Time 29:10)



The next problem that we are going to solve is as follows. Let C i be a binary linear code with code parameter given by n, k i with generator matrix G i and minimum distance d i. And let us consider a new code C, a new binary code linear code of length 2 n and message bit length k 1 plus k 2 whose generator matrix is given by this expression. Now what is the minimum distance of this new code C?

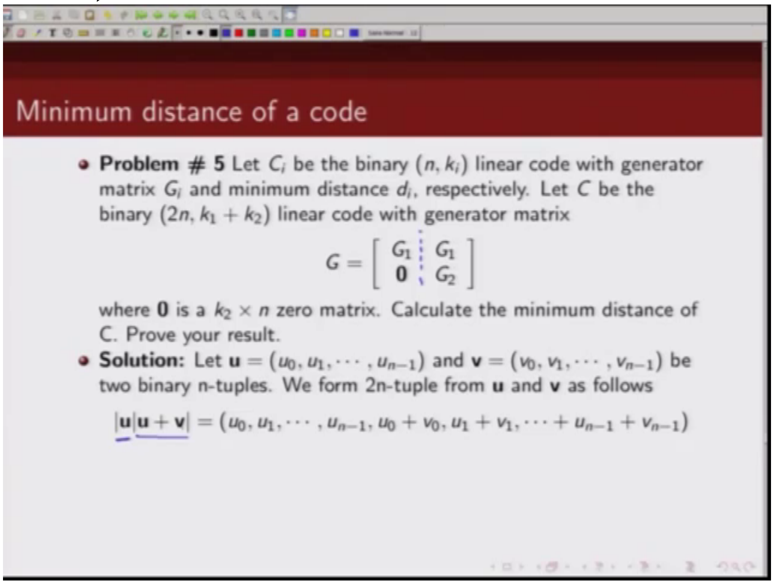
(Refer Slide Time 29:58)



So to find out the minimum distance, so let's consider let u and v are two binary n-tuples. And we form a 2 n-tuples as follows. If we look at this codeword v how is this codeword generated? So it's one n-bit codeword, another n-bit codeword. First n-bit codeword is generated by using u times v 1 and second one you get basically u times G 1 plus v times G 2.

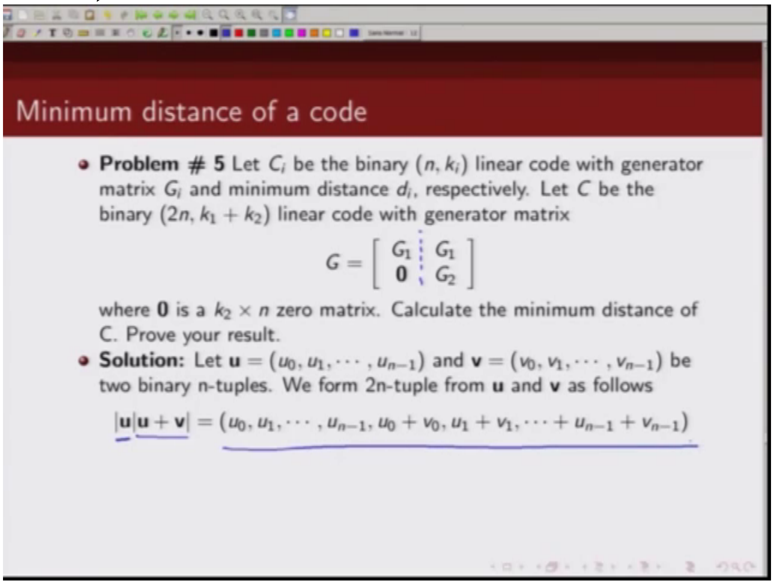
So essentially the way you are generating this codeword, the first part contains n-bit codeword u and the second part is n-bit codeword which is

(Refer Slide Time 30:42)



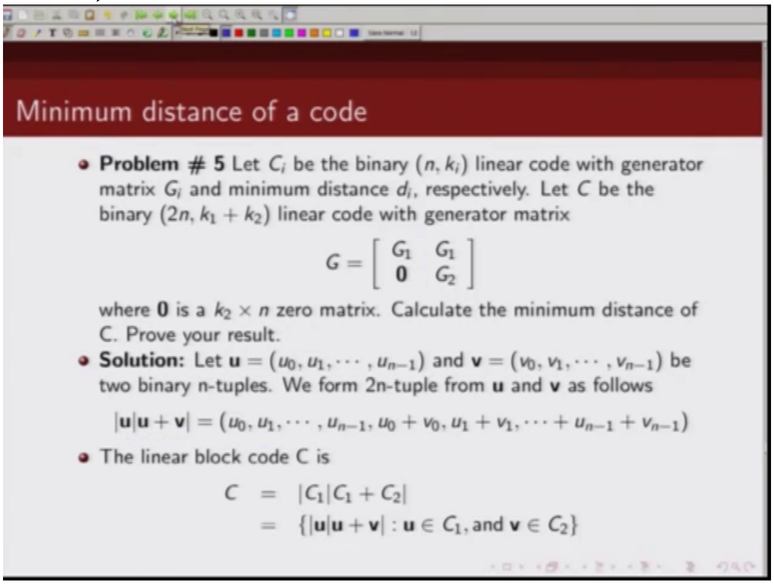
u plus v. So this 2 n length codeword will be of the form like this

(Refer Slide Time 30:52)



where the first 10 bits are u 0, u 1, u 2, u n minus 1 and then next n bits are of the form u 0 plus v 0, u 1 plus v 1 and like that, so

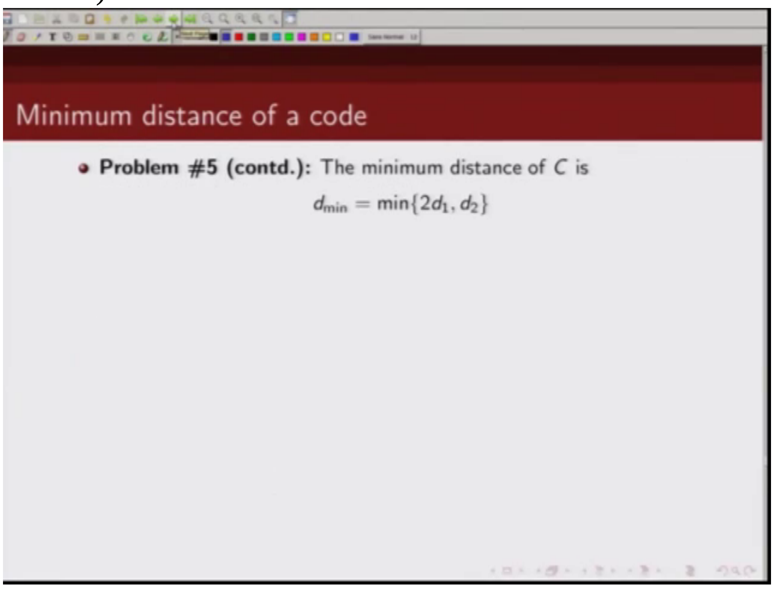
(Refer Slide Time 31:07)



as I said our linear block code, see the new code of length 2 n can be written of this form where you have a code u of length n which belongs to C 1 and then the second part which is n bit part is u plus v where u belongs to C 1 and v belongs to C 2.

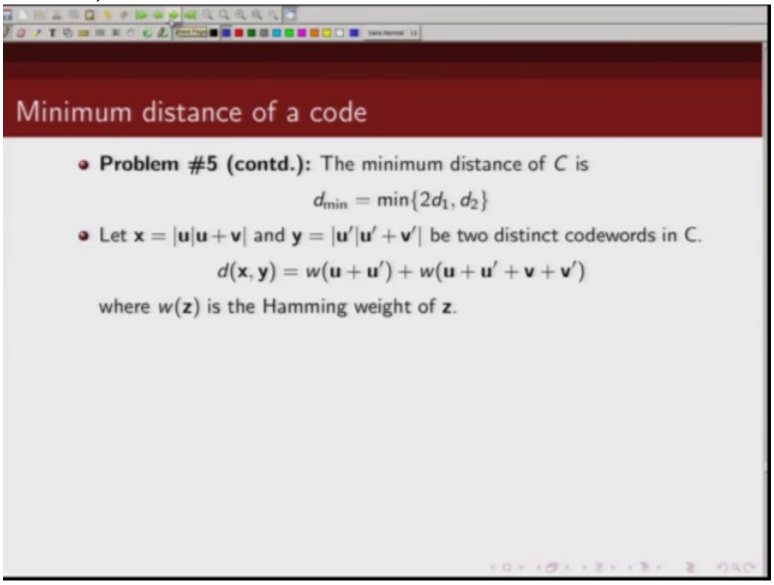
Now we will

(Refer Slide Time 31:33)



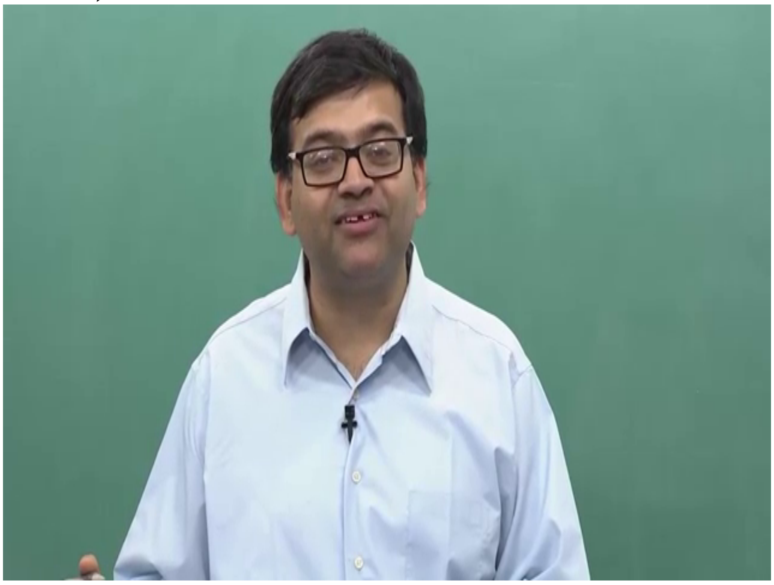
show that minimum distance of the code is minimum of two d 1 or d 2 where d 2 is the minimum distance of the code n k 2 and d 1 is the minimum distance of the code n k 1. So

(Refer Slide Time 31:51)



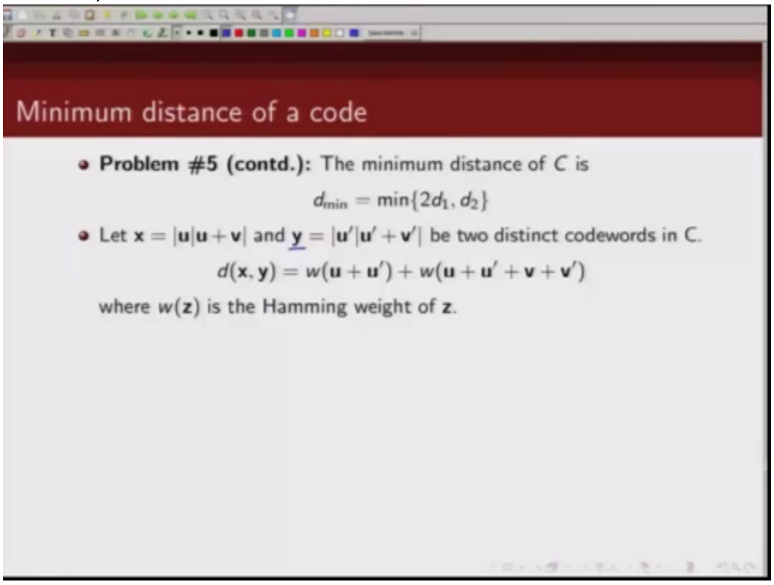
let us consider two distinct codewords x and y. So x we denote as concatenation of u and u plus v and this is u prime, u prime plus v prime. Let x and y be two distinct codewords in C. Now what is the Hamming distance

(Refer Slide Time 32:16)



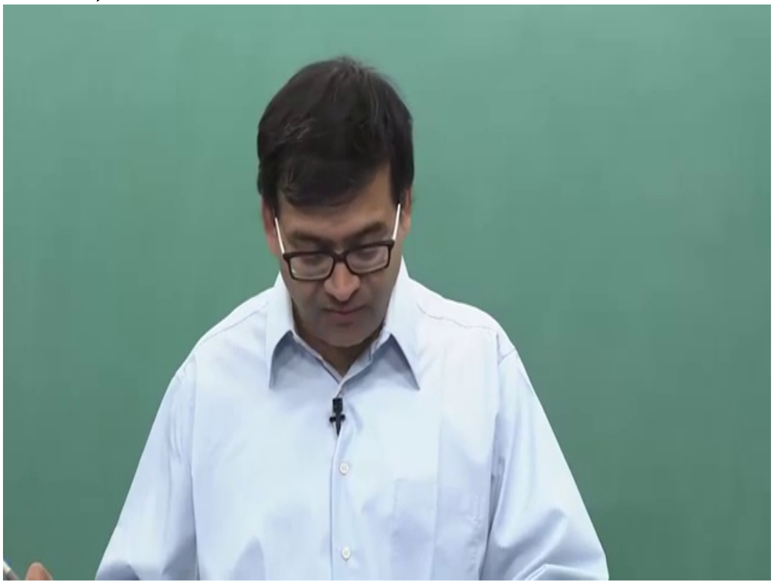
between x and y? We can write down the Hamming distance between x and y as

(Refer Slide Time 32:23)

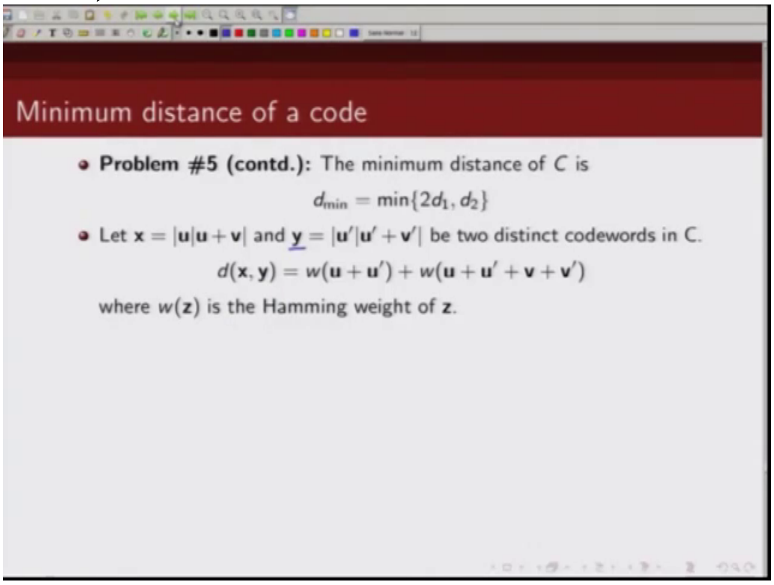


the Hamming weight between u plus u prime plus Hamming weight between u plus v plus v hat plus u hat. So the Hamming distance between x and y can be written as Hamming weight of u plus u hat plus Hamming weight of u plus u hat plus v plus v hat.

(Refer Slide Time 32:50)

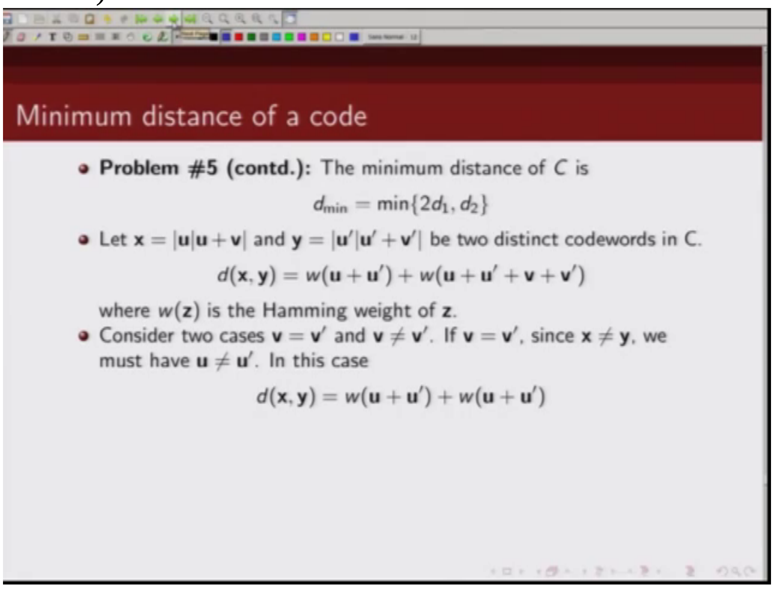


(Refer Slide Time 32:52)



Now note

(Refer Slide Time 32:54)

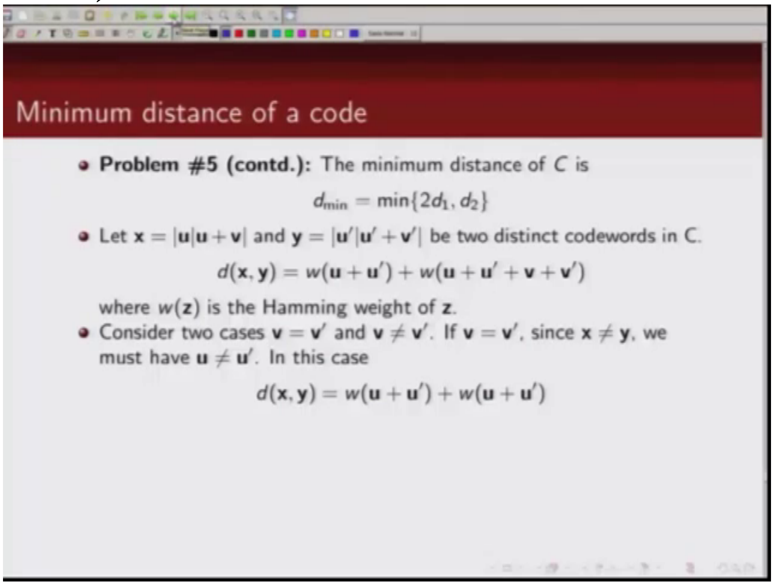


x and y are distinct codewords. So let us consider two scenarios. In first case we will consider v is same as v prime. In second case we will consider v is not same as v prime. So if we consider v as same as v prime, since x and y are distinct codewords what we will have is u is not same as

(Refer Slide Time 33:22)

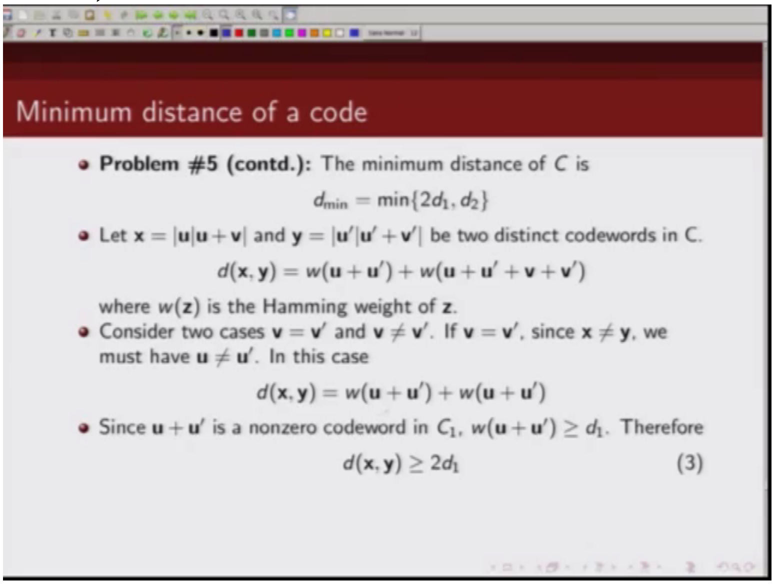
u prime. So in this case the Hamming

(Refer Slide Time 33:26)



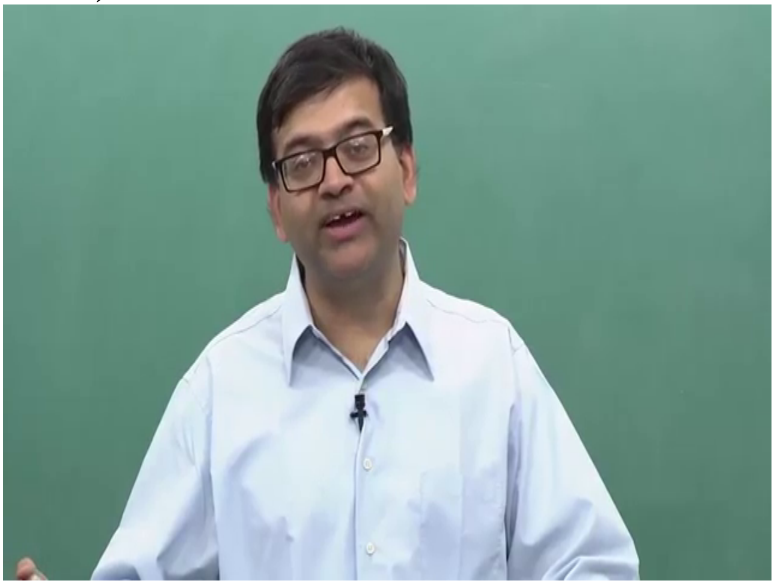
distance between x and y will be given by Hamming weight of u plus u prime plus, now since v and v prime are same this will be zero so this will be same as Hamming weight of u plus u prime. So then in this case when v is same as v prime we can write the Hamming distance between x and y as Hamming weight of u plus u prime plus Hamming weight of u plus u prime. And since what is u plus u prime?

(Refer Slide Time 34:03)



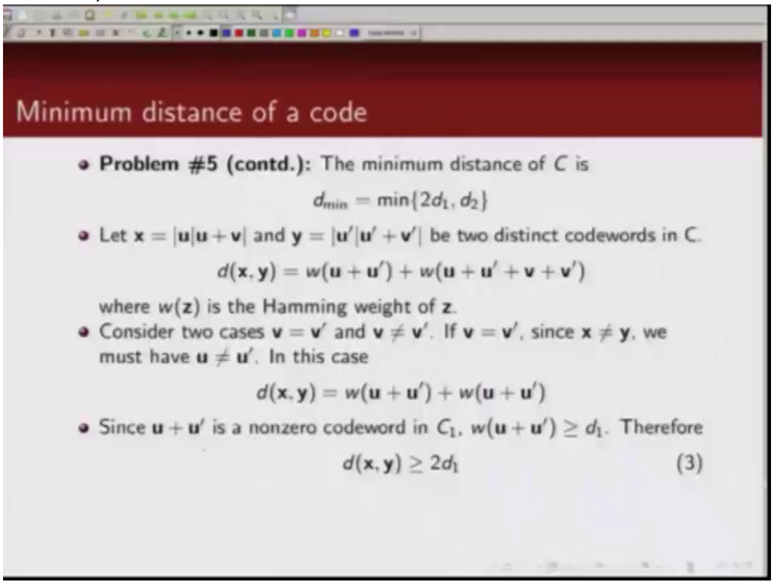
u and u prime are two codewords belonging to

(Refer Slide Time 34:08)



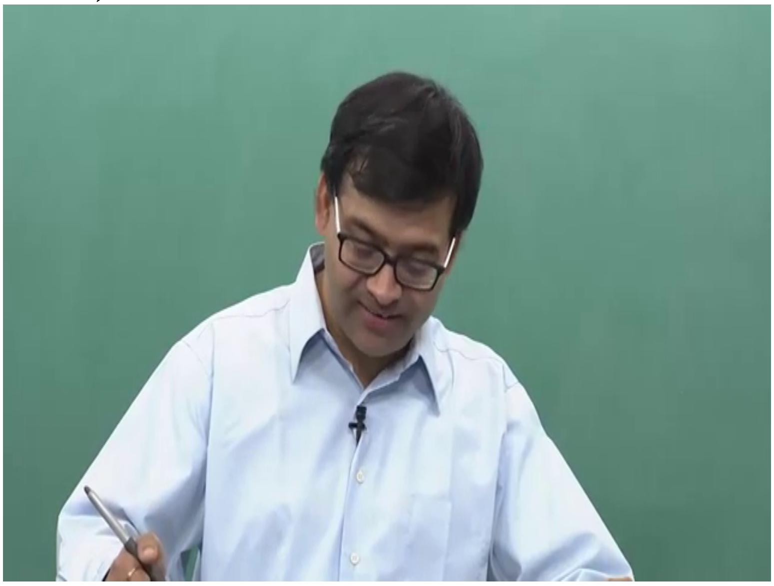
C 1. So sum of two codewords for a linear block code is another valid codeword. So u plus u prime is going to be

(Refer Slide Time 34:18)



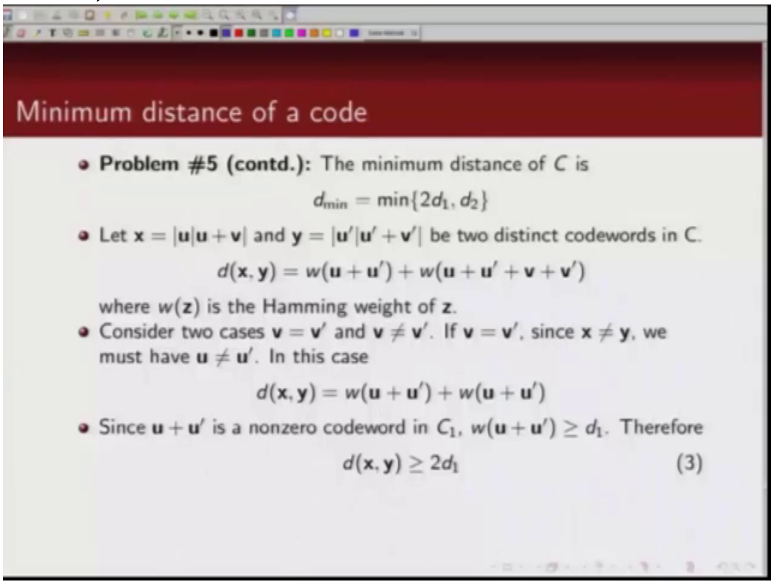
another valid codeword. So then, so what would be the minimum distance of u plus u prime? It would be at least the minimum distance of the code C 1 which is d 1.

(Refer Slide Time 34:31)



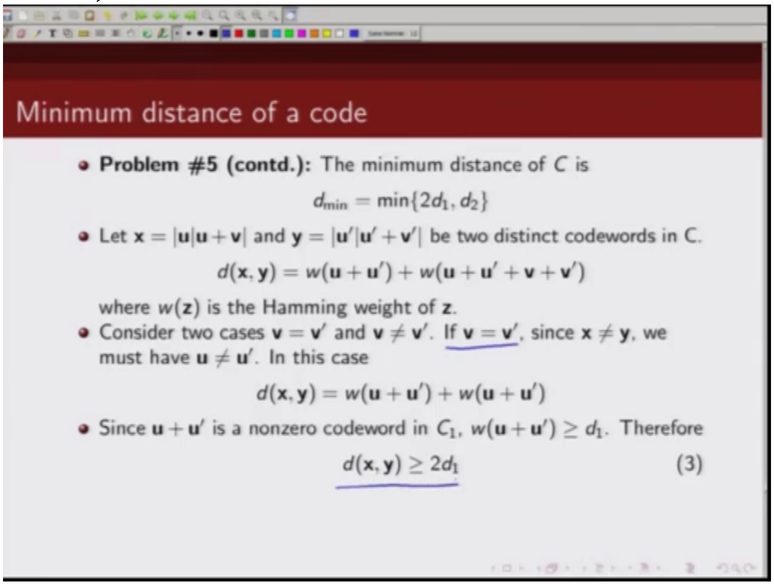
So then Hamming distance between x and y would be

(Refer Slide Time 34:36)



greater than equal to 2 times d 1. So for the case when v is equal to v prime we have shown that minimum distance

(Refer Slide Time 34:49)

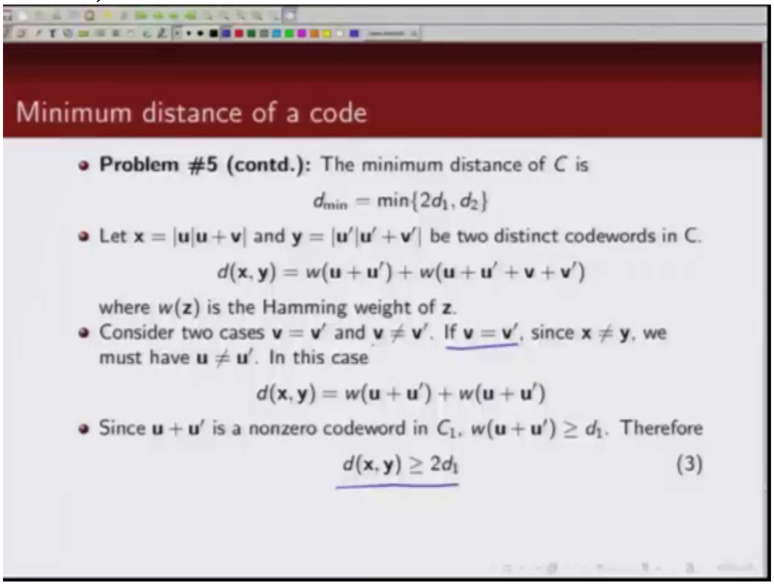


should be at least 2 times d 1.

(Refer Slide Time 34:53)

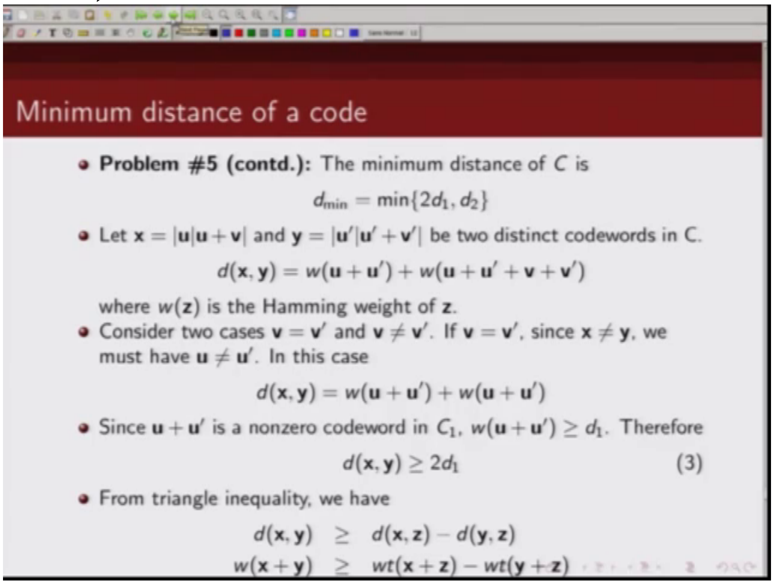
Now let us consider the case when

(Refer Slide Time 34:58)



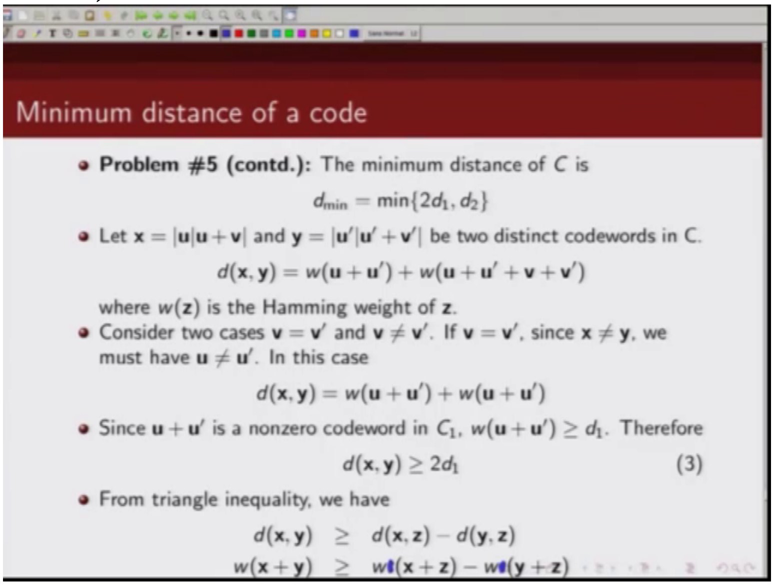
v is not equal to v prime. So before that we will just

(Refer Slide Time 35:05)



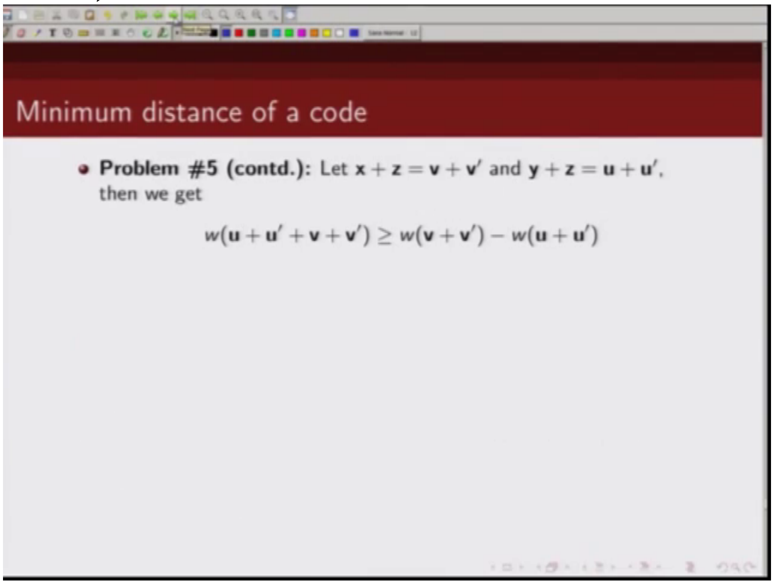
state again the triangular inequality that we are going to use; so from the triangular inequality we know that Hamming distance between x and y is greater than equal to Hamming distance between x and z minus Hamming distance between y and z. And we know the Hamming distance is nothing but Hamming weight of x plus y, Hamming weight of x plus y and Hamming weight of x plus z minus Hamming weight of

(Refer Slide Time 35:36)



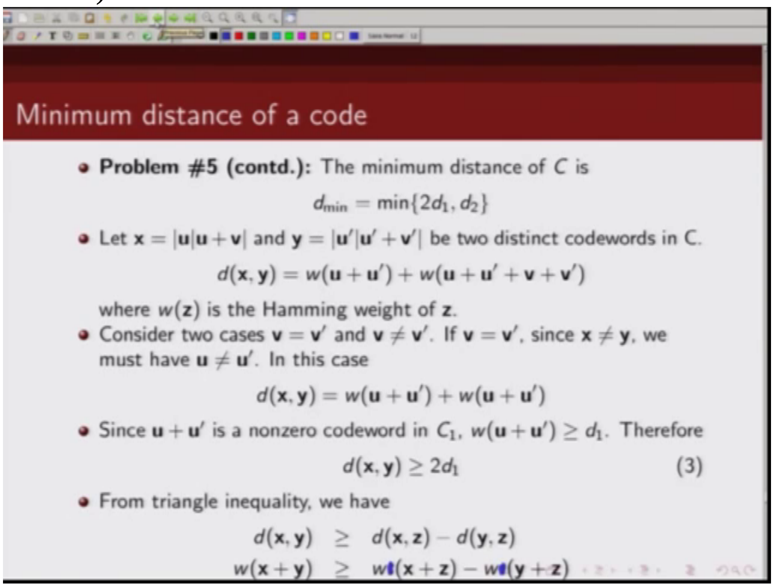
y plus z. So we can write this expression in terms of Hamming distance or we can write in terms of Hamming weight. Now let us take x plus z to be

(Refer Slide Time 35:48)



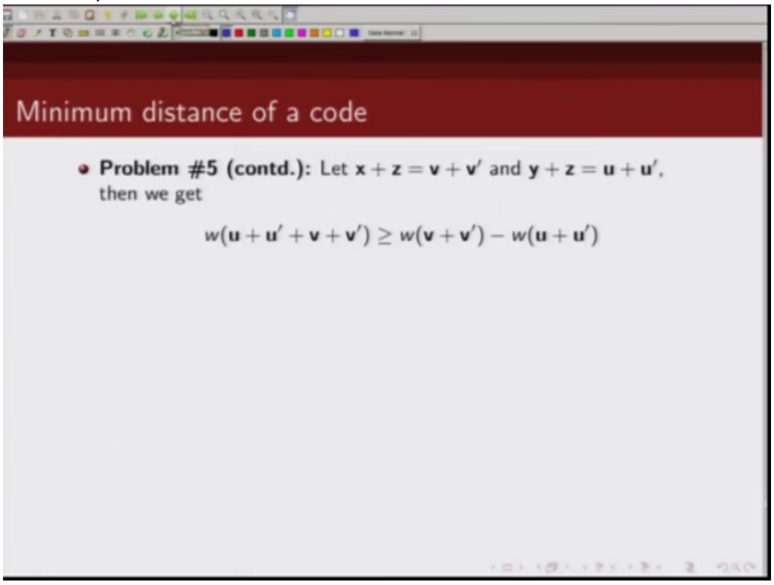
equal to v plus v prime and y plus z as u plus u prime. And we put these values of x

(Refer Slide Time 35:58)



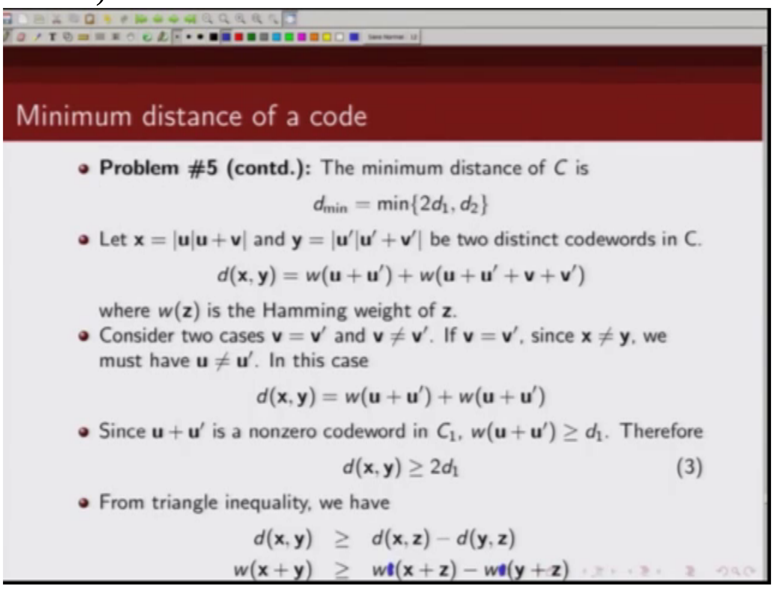
and x plus y and x plus z and y plus z, we put these values

(Refer Slide Time 36:05)



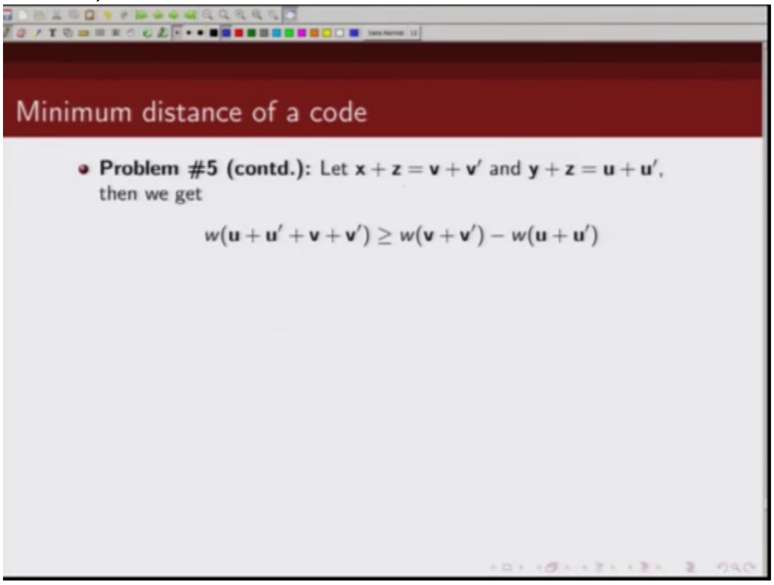
in this expression.

(Refer Slide Time 36:07)



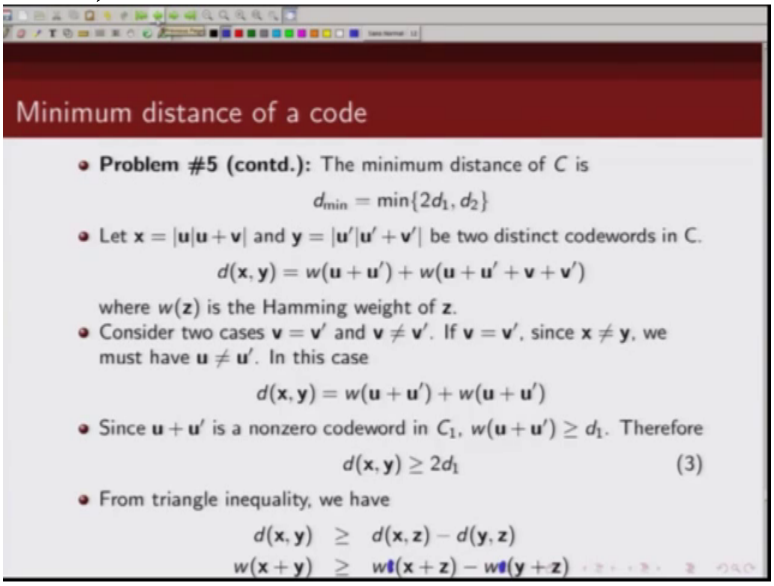
So what is x plus y? x plus y

(Refer Slide Time 36:12)



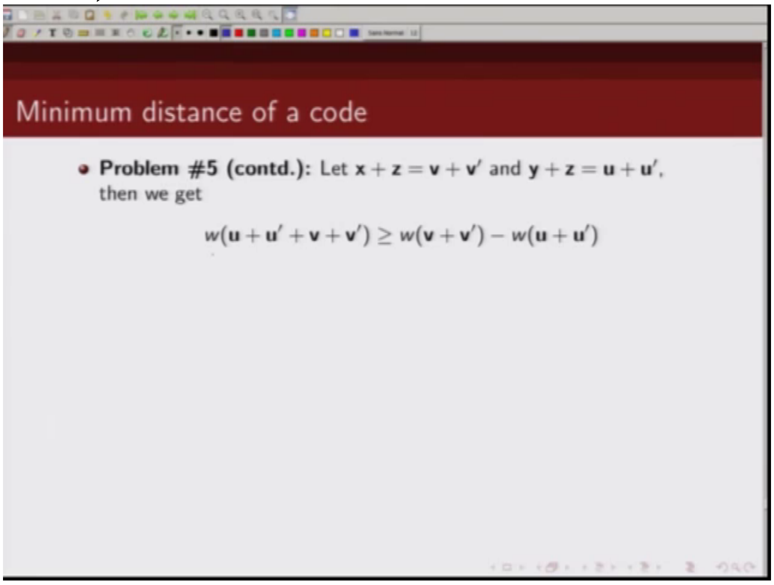
would be v plus v prime plus u plus u prime so x plus y is basically u plus u prime plus v plus v prime. So from here weight of

(Refer Slide Time 36:27)



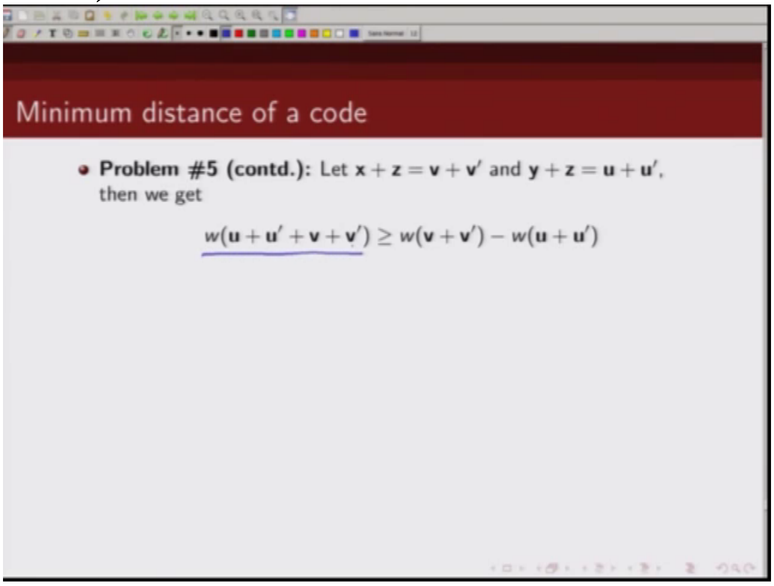
x plus y is given

(Refer Slide Time 36:30)



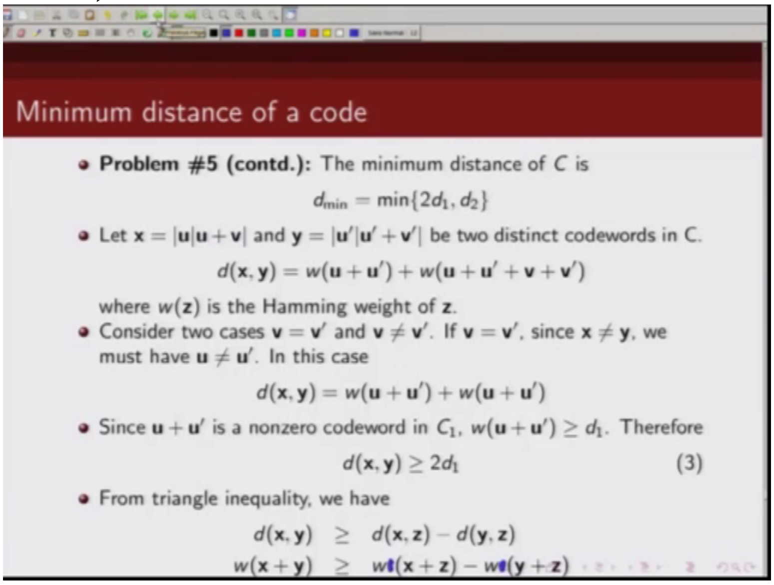
by this.

(Refer Slide Time 36:33)



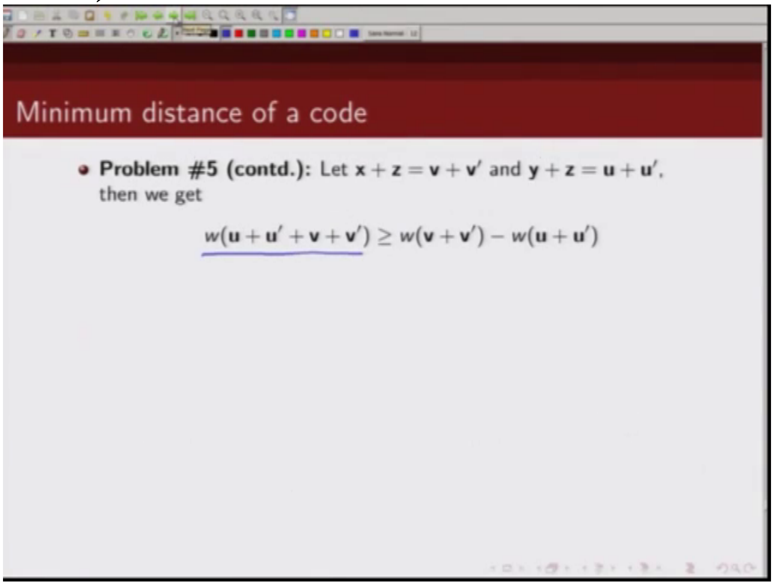
Next what we had was weight of

(Refer Slide Time 36:36)



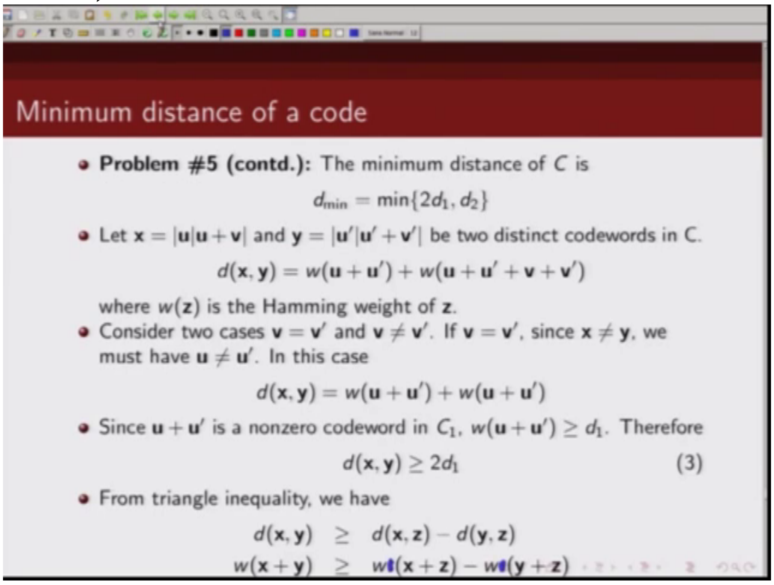
x plus z. What is weight of x plus z?

(Refer Slide Time 36:40)



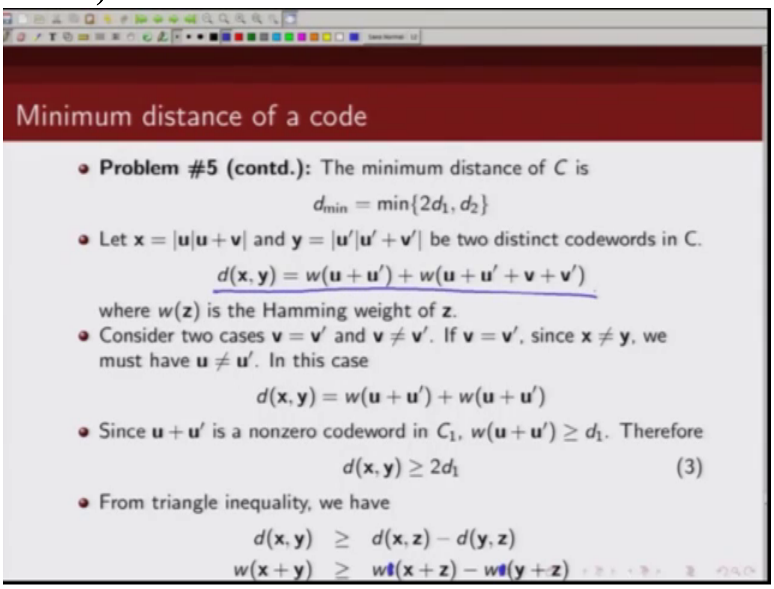
Weight of x plus z is weight of v plus v prime and similarly weight of y plus z is given by this, Ok. So this is upper bounded, this is lower bounded by this quantity. This is lower bounded by this quantity. Now go back and

(Refer Slide Time 37:06)



see what is our minimum distance between x and y? Minimum distance between x and y is given by

(Refer Slide Time 37:14)

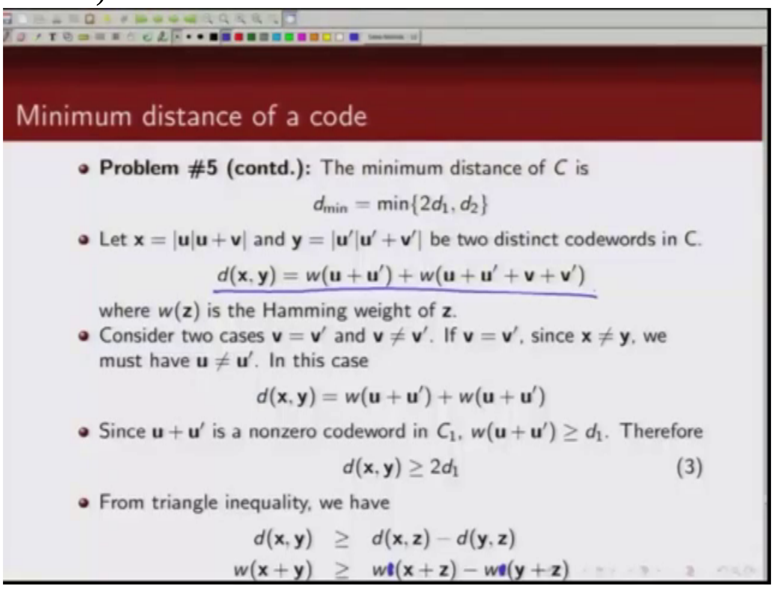


this expression. It's Hamming weight between u and u prime

(Refer Slide Time 37:22)

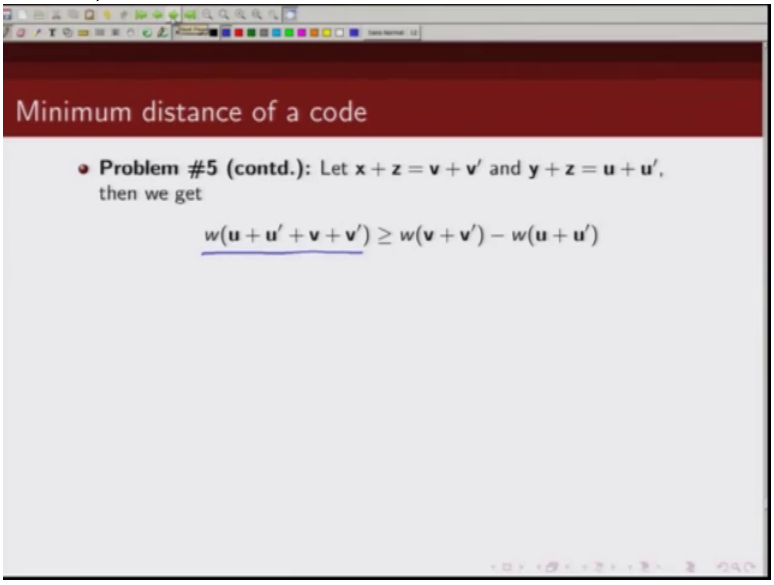
plus Hamming weight of u plus u prime plus

(Refer Slide Time 37:26)



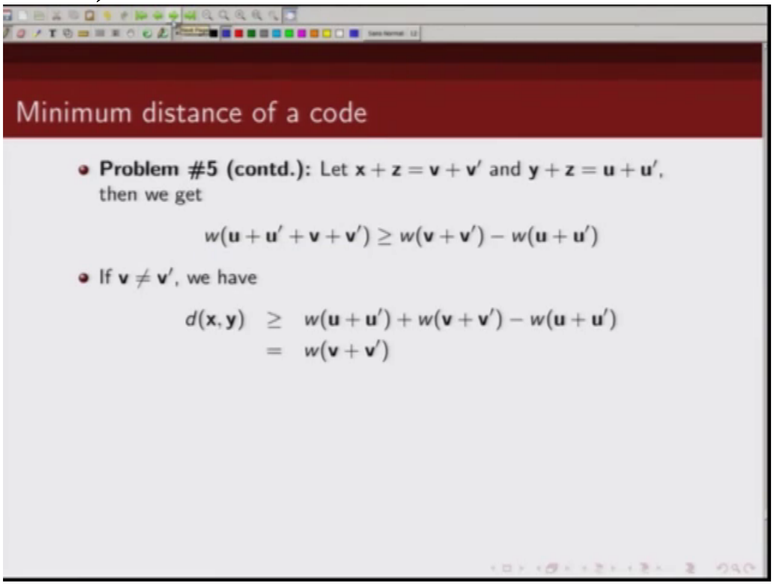
v plus v prime. And what we did just now is we lower bounded this. So then Hamming distance between x and y

(Refer Slide Time 37:40)



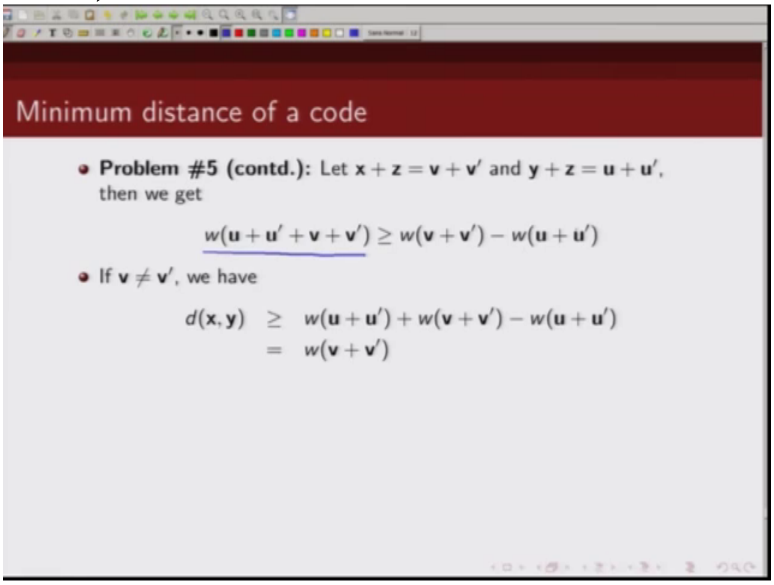
can be, so

(Refer Slide Time 37:42)



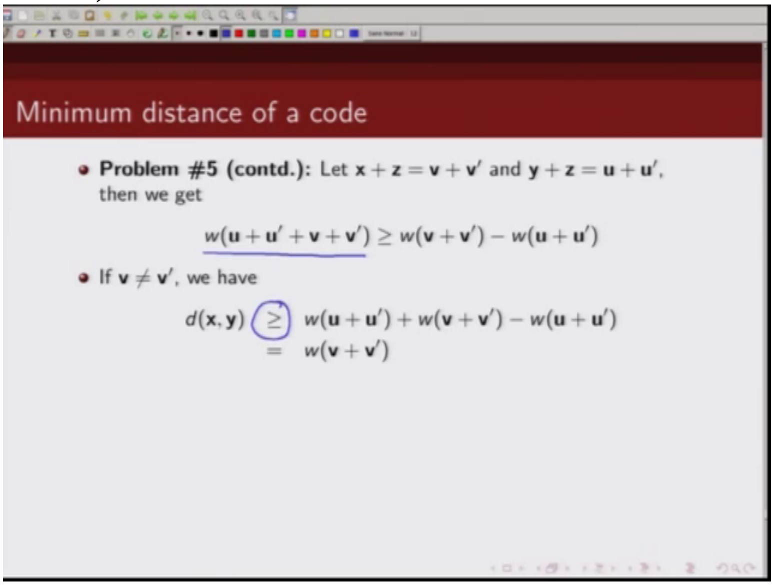
this basically this we lower bounded by this quantity.

(Refer Slide Time 37:45)



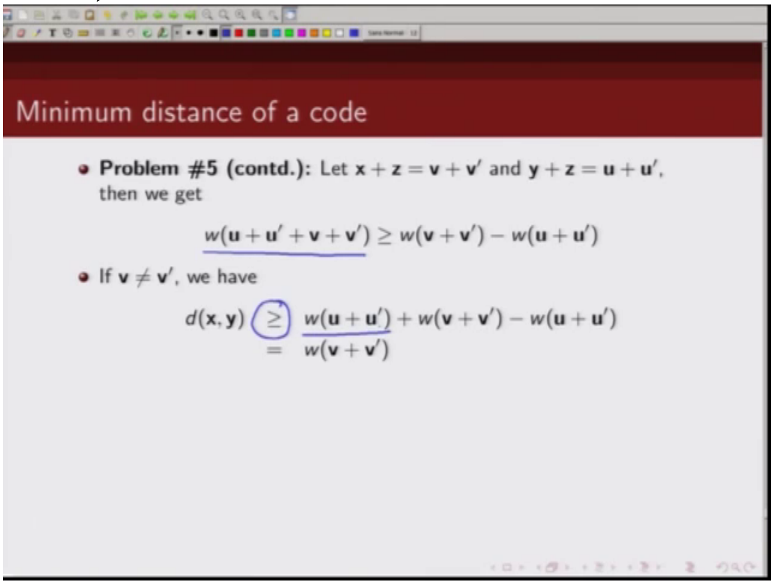
If we plug that in here, what we get here is greater than equal to.

(Refer Slide Time 37:51)



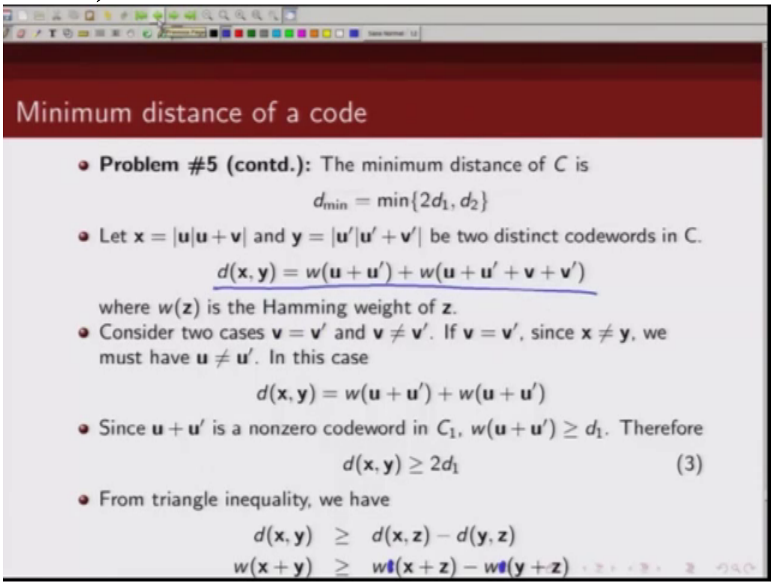
So what we can write is the Hamming distance between x and y is then greater than or equal to this term comes

(Refer Slide Time 38:02)



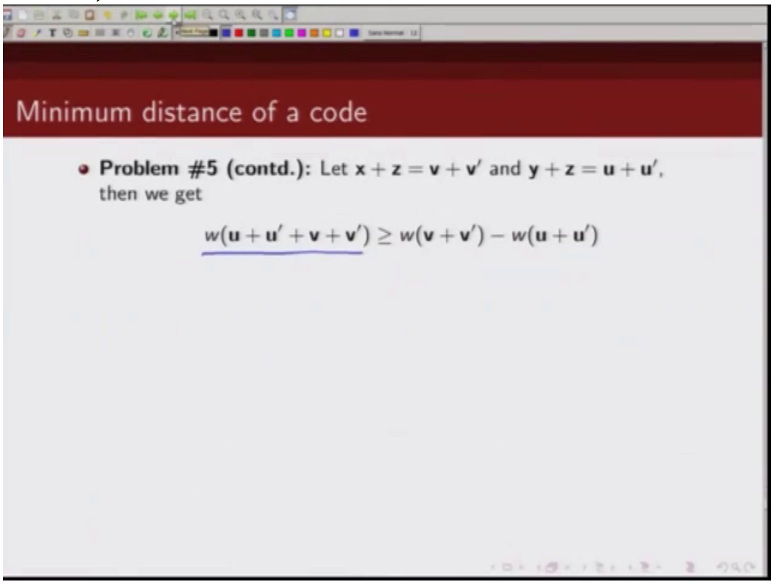
from here, this term

(Refer Slide Time 38:05)



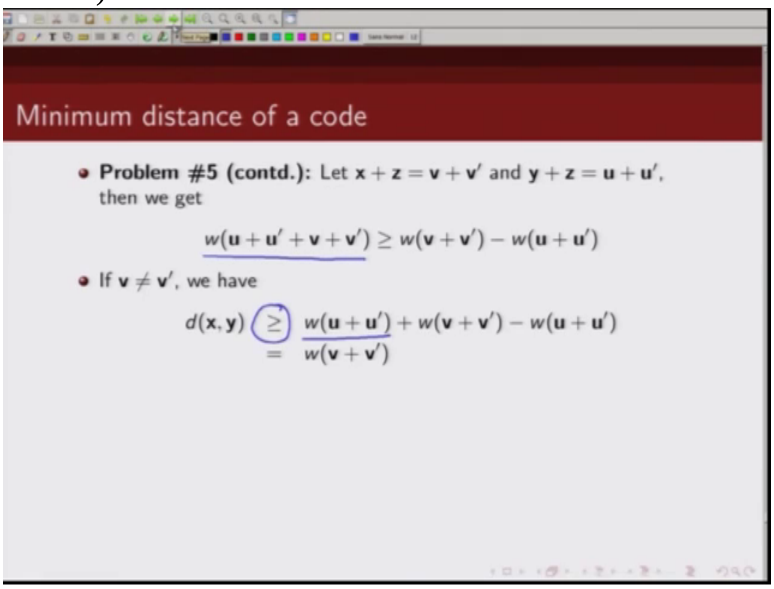
and this term is lower bounded by this term

(Refer Slide Time 38:09)



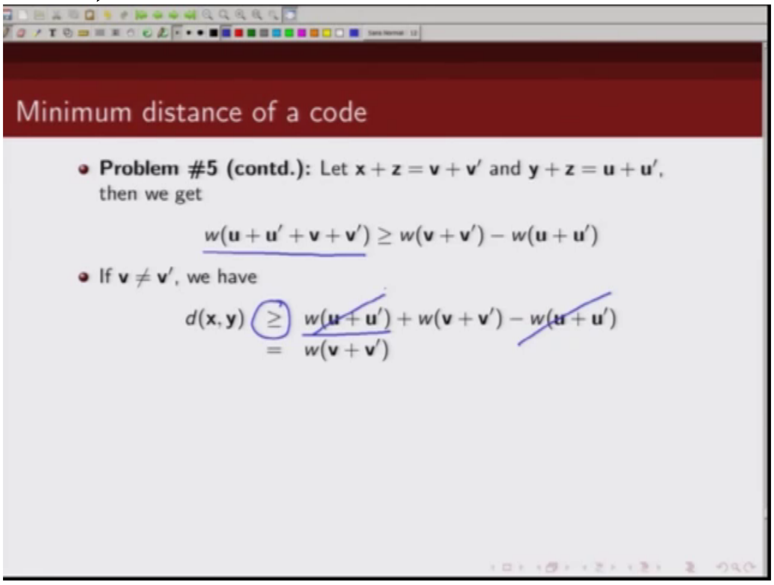
here. So we write it

(Refer Slide Time 38:12)



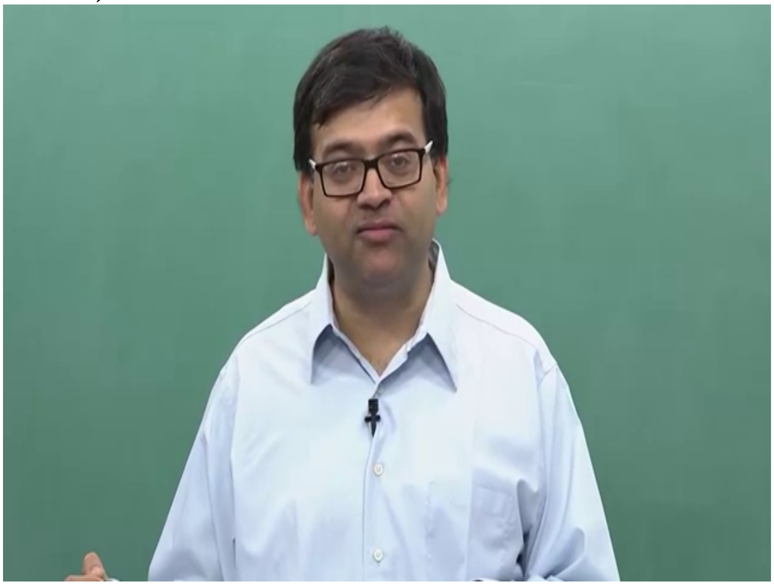
here, now this can be further written as Hamming weight of v plus v prime because these two cancel out. So what

(Refer Slide Time 38:23)



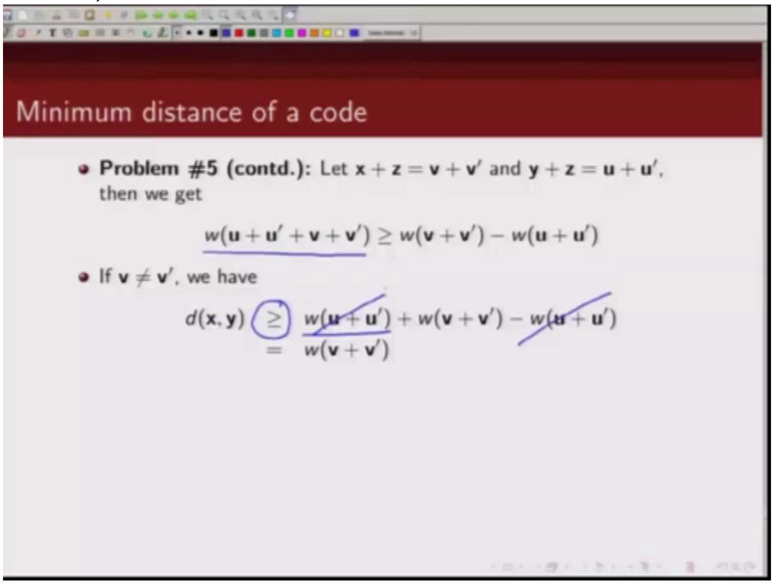
we have shown is when v is not same as v prime the Hamming distance between

(Refer Slide Time 38:30)



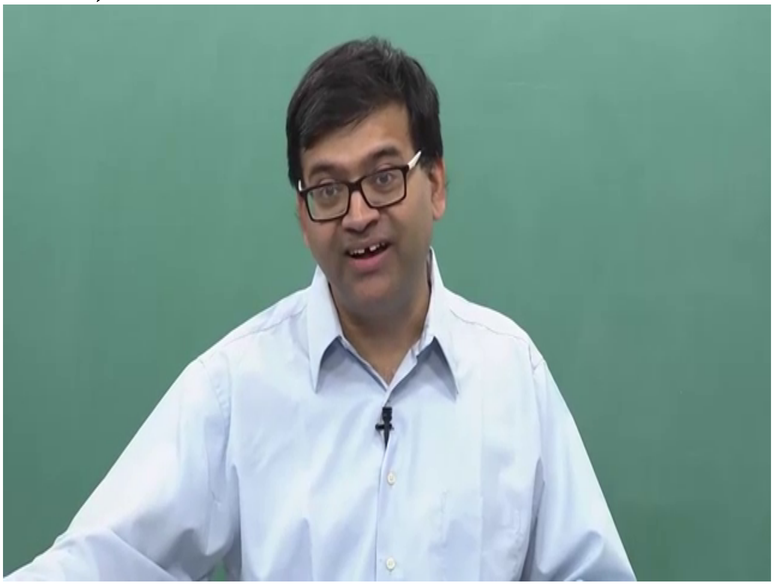
x and y is greater than equal to Hamming weight

(Refer Slide Time 38:36)



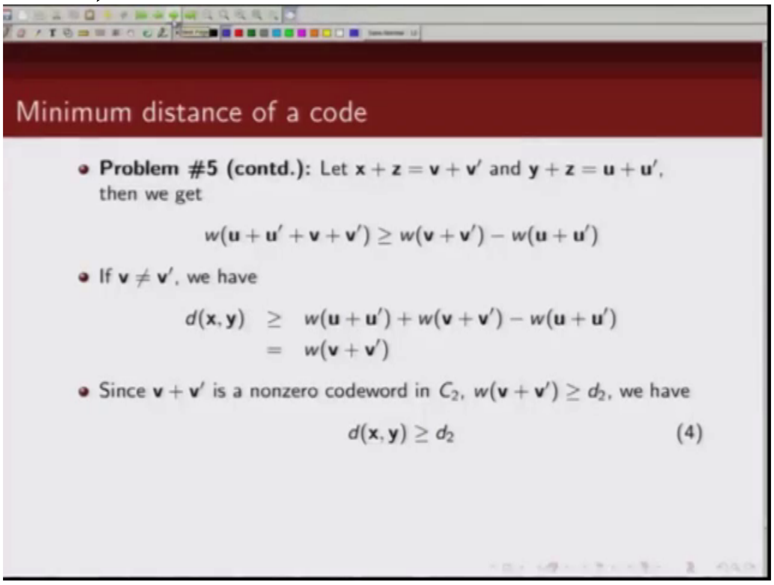
of v plus v prime. And is v plus v prime? v and v prime are valid codewords in linear block code C 2 with minimum distance d 2 so v plus v prime will be another valid codeword in C 2

(Refer Slide Time 38:53)



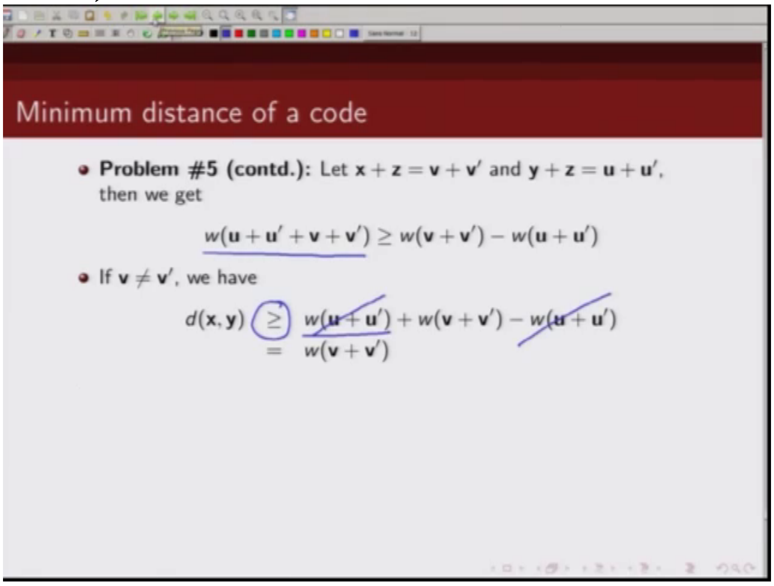
whose minimum distance is d 2. So then we can

(Refer Slide Time 38:58)



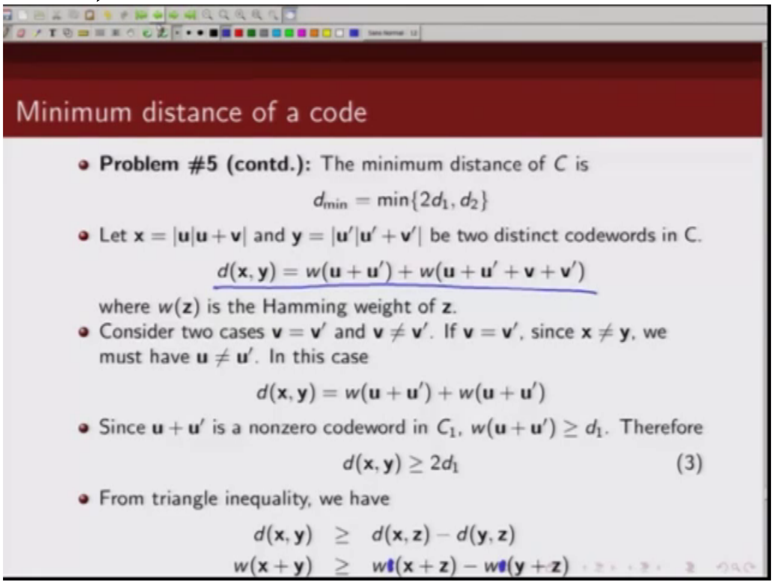
write this as Hamming distance between x and y is greater than equal to d 2. So we look, comparing equation number 4 and equation

(Refer Slide Time 39:11)



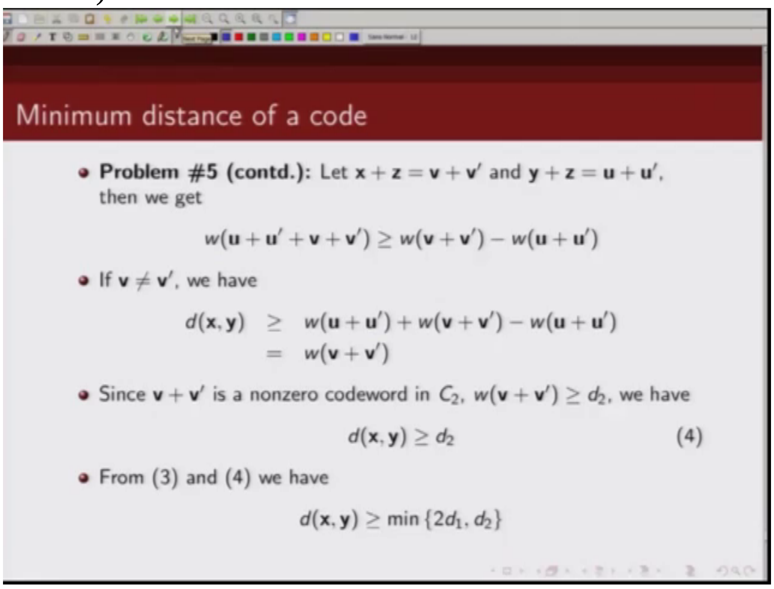
number 3,

(Refer Slide Time 39:12)



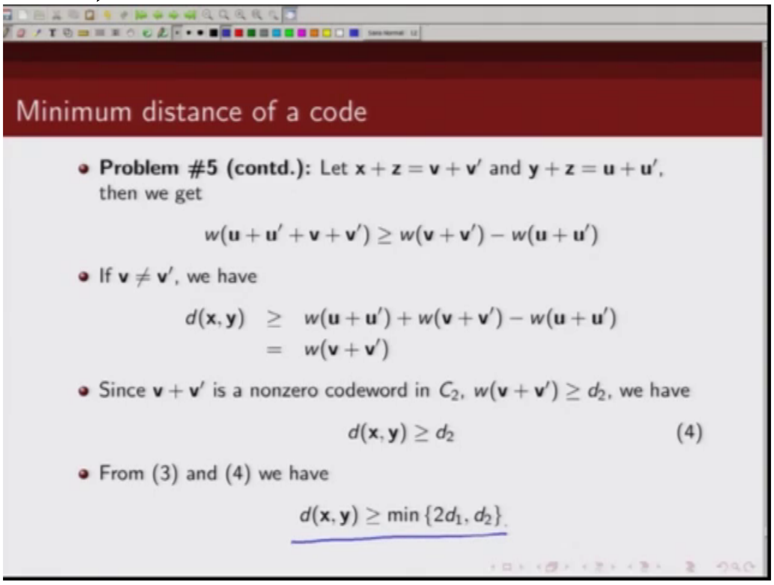
if we compare these two equations we can write that minimum distance of the code is minimum of 2 d 1 or d 2, Ok.

(Refer Slide Time 39:28)



Now let us show that

(Refer Slide Time 39:31)

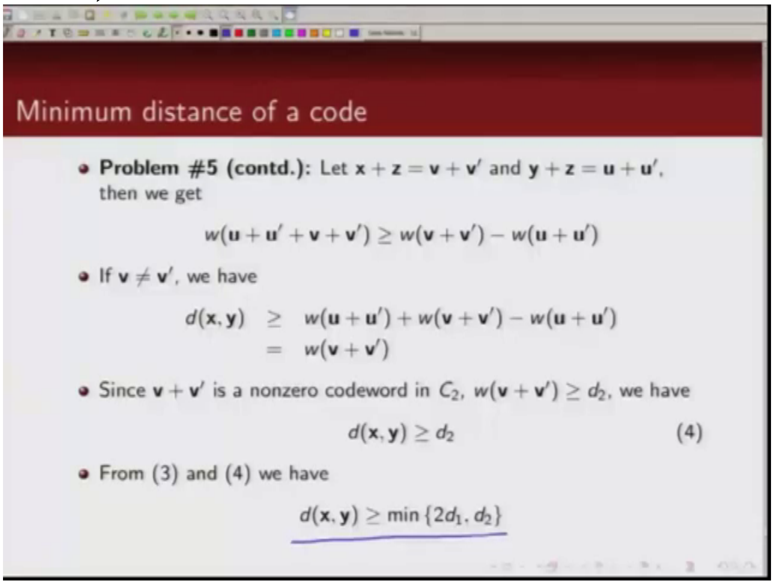


there exists a code

(Refer Slide Time 39:33)

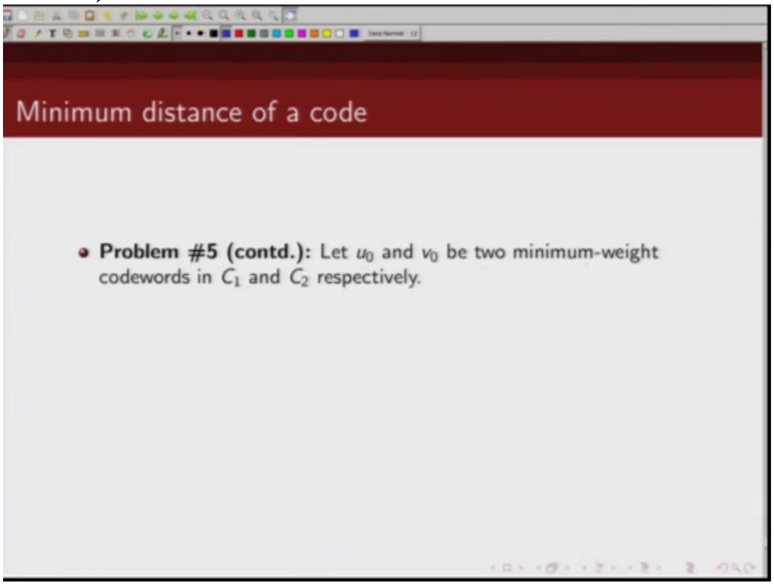
with, minimum distance of code is indeed

(Refer Slide Time 39:36)



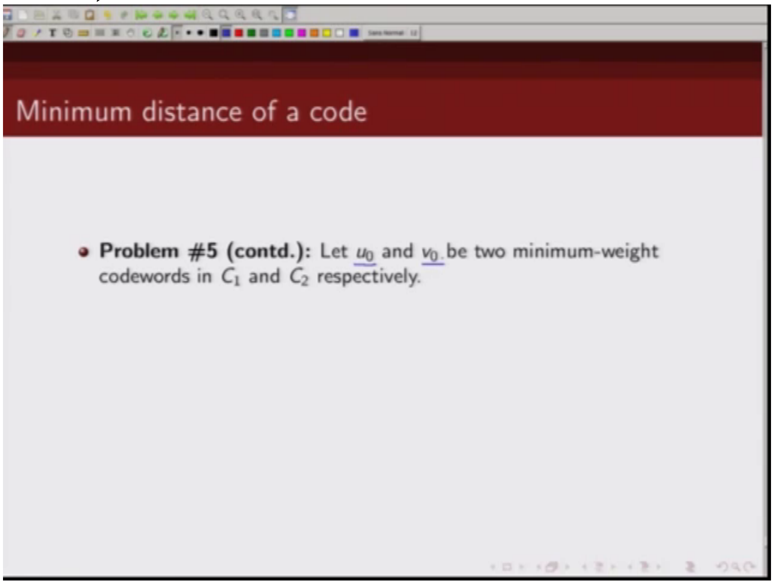
equal minimum of 2 d 1 or d 2. So let us take two minimum weight codewords

(Refer Slide Time 39:44)



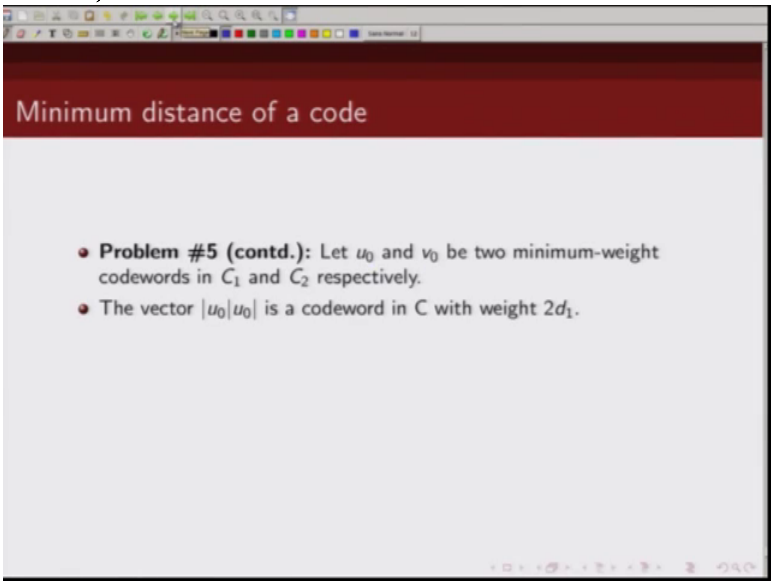
in C 1 and C 2. Let us call them u 0

(Refer Slide Time 39:49)



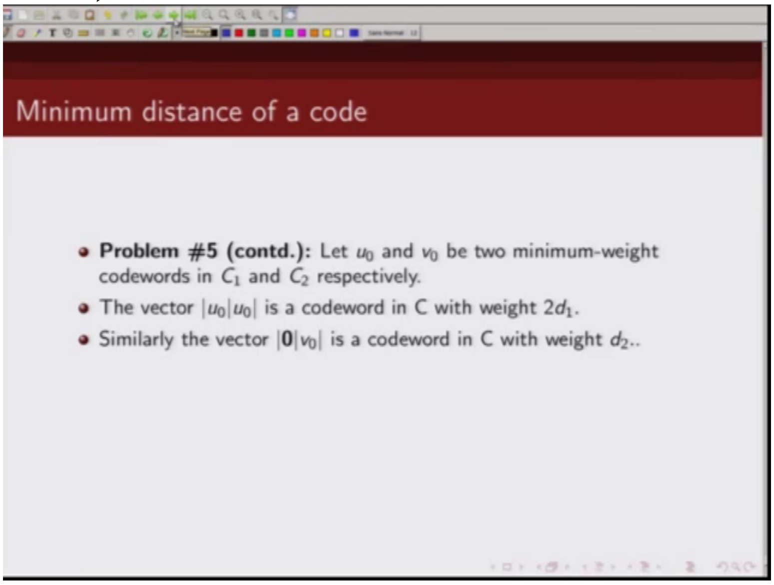
and v 0.

(Refer Slide Time 39:52)



Now this is a valid codeword in C and what's its minimum distance? It is 2 times d 1.

(Refer Slide Time 40:14)

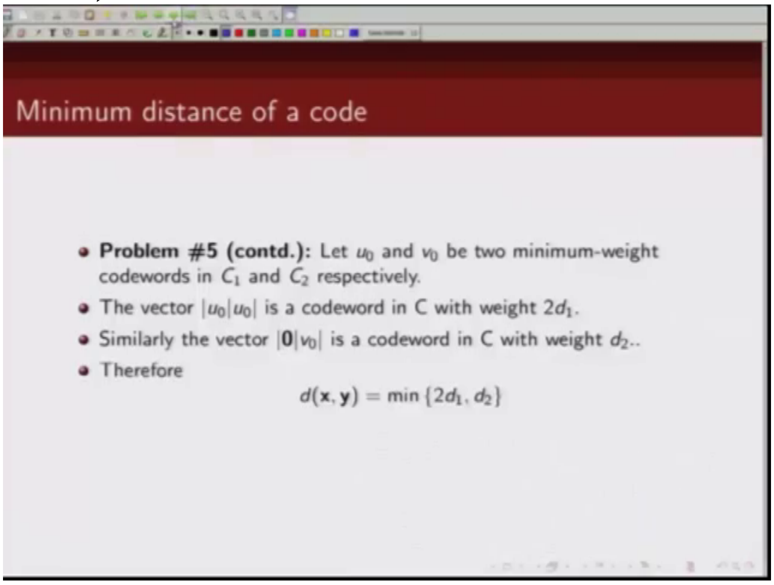


So if we take v 0 to v all zero codeword, what we get is u 0 and u 0, this is a valid codeword in C. And its minimum distance is 2 times d 1. Similarly if we take u 0 to be all zero codeword, then what we get is this codeword 0 and u 0, whose minimum distance is d 2. So hence we have shown that there,

(Refer Slide Time 40:28)

basically there minimum distance of code, of this code new code c is indeed minimum of 2 d 1 or d 2,

(Refer Slide Time 40:40)



Ok. Thank you