Digital Communication

Prof. Bikash Kumar Dey

Department of Electrical Engineering

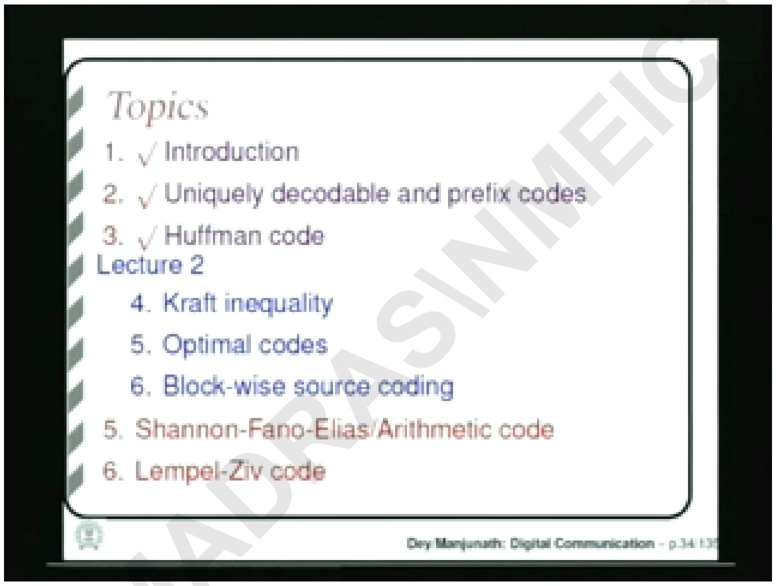
IIT Bombay

Lecture No - 27

Source Coding (Part-2)

so this is the second class on source coding

(Refer Slide Time: 00:00:59 min)



Topics

1. √ Introduction
2. √ Uniquely decodable and prefix codes
3. √ Huffman code

Lecture 2

4. Kraft inequality
5. Optimal codes
6. Block-wise source coding

5. Shannon-Fano-Elias/Arithmetic code
6. Lempel-Ziv code

Dey Manjunath: Digital Communication – p.34/135

IITMADRASINMEICT

in the last class we have seen why we need source coding and why it is possible to do source coding for most of the practical sources

and we have seen some of the desirable properties of source codes and what we should remember while designing source codes

so in particular we have seen that we should design uh uniquely decodable codes

so that at the designation we can design we can decode the code successfully and also we would like have prefix code so that we can decode instantaneously

then we have seen one particular source code technique called Huffman code and in this class we will do Kraft inequality which will give us some uh criteria for us being able to design a code of certain given lengths and then we will discuss about optimal codes there we will see what is possible how much compression is possible for any source

so obviously we cannot compress to uh any amount any size we want

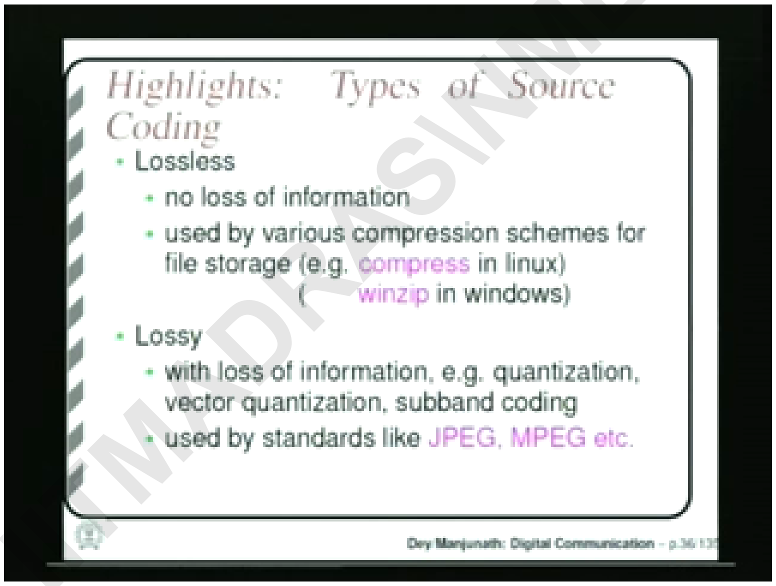
so what is the limit so we'll discuss about the limit uh of compression and then we'll talk about block wise source coding

so we have so far seen that uh we can uh depending on the statistics of the source we can code individual symbols to get compression but we can also combine several symbols and then uh encode a block of symbols together

so that way in fact we will see that we can achieve better compression

so first we will see what we have done in the last class

(Refer Slide Time: 00:02:47 min)



Highlights: Types of Source Coding

- Lossless
  - no loss of information
  - used by various compression schemes for file storage (e.g. compress in linux)
    (winzip in windows)
- Lossy
  - with loss of information, e.g. quantization, vector quantization, subband coding
  - used by standards like JPEG, MPEG etc.

Dey Manjunath: Digital Communication – p.36/135

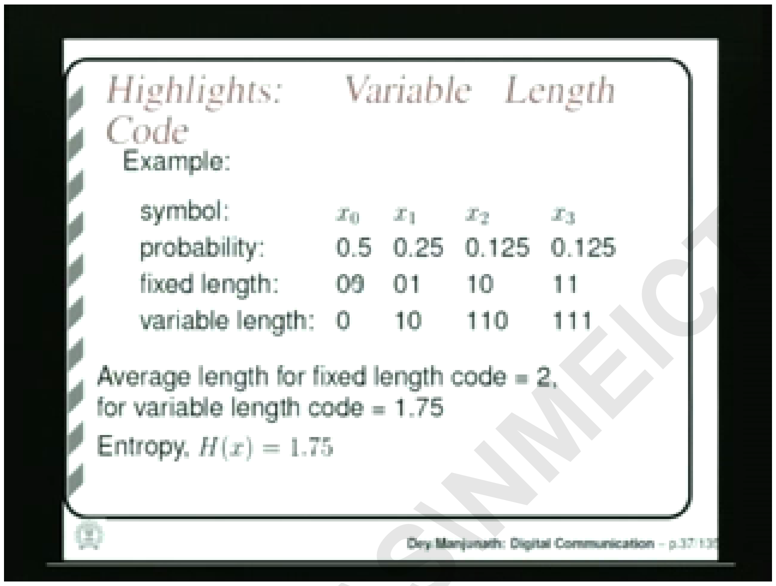
IITMADRAS|NPTEL

we have seen that there are two types of uh two types sources coding one lossless where there is no loss of information due to uh compression and the other is lossy compression where there is some of loss of information due to compression

so uh the a typical examples of lossless source coding is uh all the file compression techniques we using computer like in Linux we have ah command compress and in windows we use WinZip so these are all lossless compression technique obviously because we can uh get the file back from the compressed ah file in the lossy compression the typical examples are the standard like JPEG MPEG etcetera uh where speech or audio or video signals uh {im} (00:03:41 min) even image signals can be compressed very significantly

so first we have seen that ah

(Refer Slide Time: 00:03:52 min)



Highlights: Variable Length Code

Example:

symbol:	x_0	x_1	x_2	x_3
probability:	0.5	0.25	0.125	0.125
fixed length:	00	01	10	11
variable length:	0	10	110	111

Average length for fixed length code = 2,
for variable length code = 1.75

Entropy, $H(x) = 1.75$

Dey Manjunath: Digital Communication – p.37/135

we we should do variable length coding whenever needed it is not always possible to get good compression using fixed length codes

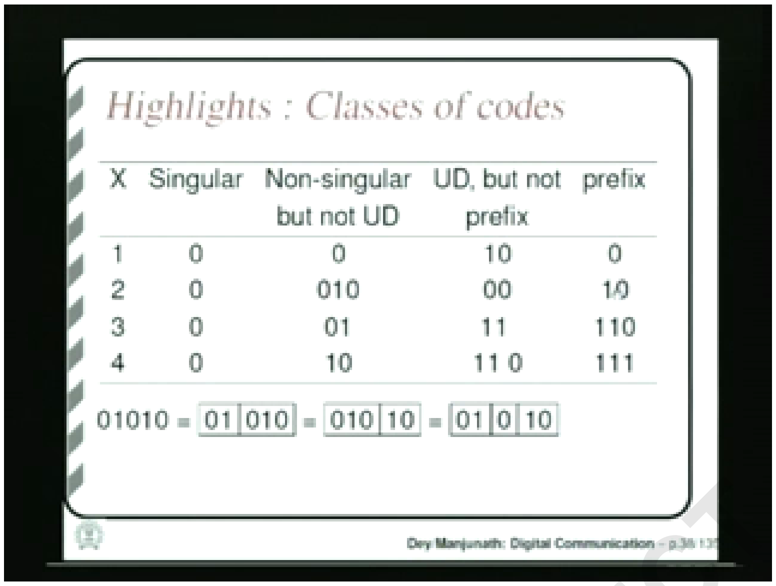
ah for example we have taken this particular random variable which ah takes four possible values with this probabilities and with fixed length code we can ah compress the signal

we can represent the random variable using two bits and ah so the average number of bits per symbols is two whereas using with this variable length code the average length part symbol comes to be one point seven five bits per symbol

and this is the ah maximum compression that can be done because the entropy of the random variables itself is one point seven five

(Refer Slide Time: 00:04:41 min)

IITMADRASINMEICT



Highlights : Classes of codes

X	Singular	Non-singular but not UD	UD, but not prefix	prefix
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

01010 = 01|010 = 010|10 = 01|0|10

Dey Manjunath: Digital Communication

and we have seen different kind of codes we have seen ah what is singular code where there are there are possibly two uh same codewords two different symbols represented by same codewords in this particular example this trivial example where all the codewords are same

obviously this code cannot be used ah uh to represent the sources because in that situation at the destination we will not be able to recover the values of the signal from the encoded bits

so the codeword should be different and when all the codewords are different the code is called non singular code but even this may not be uniquely decodable when we are transmitting one after another symbols because when you transmit a sequence of symbols that sequence of symbols should not be ah decodable uh in a two in two different ways

so that is a further ah restriction we have on the code

so we can see in fact that this code is not uniquely decodable in that sense because if you transmit for example zero one zero we don't no whether we have received first zero and then one zero or we have received this single codeword

so this code though it is singular it is not uniquely decodable

so this code for example is uniquely decodable but even then this code if this code is used we may not be able to decode the code instantaneously meaning by whenever we have received a symbol we may not be able to we may not be able to decode at that instant

we may need wait for longer time before we know that well this is what uh this particulars uh bits sequence was representing this particular symbol

IITMADRASNMEICT

so we may (00:06:28 min) {did}need to wait for several more codewords to reach before we can decide on a previous ah codeword

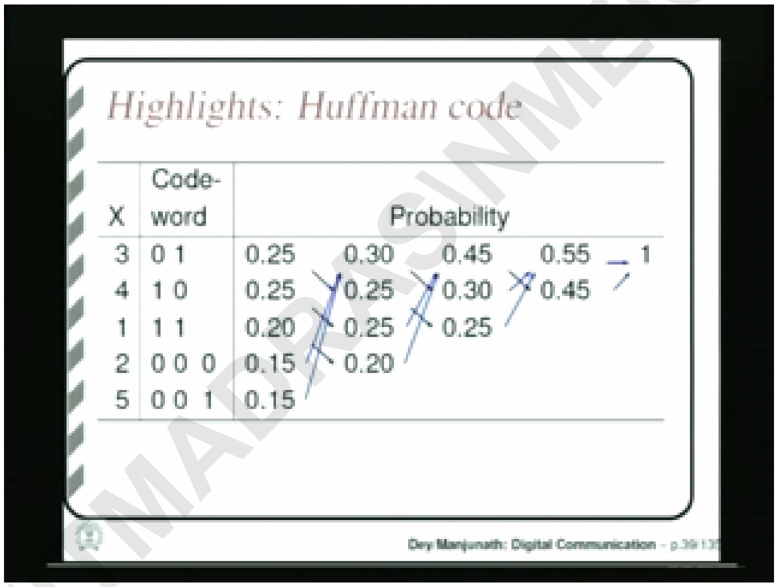
and that is not required when the code satisfies another more stringent property called prefix condition

such codes are called prefix codes or prefix free codes or instantaneous codes because they can be decoded instantaneously

here no codeword is a prefix of another codeword in that uh kind of codes the the codes can be decoded instantaneously okay

so this is to show that this code is not a uniquely decodable and that the shows this is not prefix code because this codeword appears as a prefix of this codeword okay

(Refer Slide Time: 00:07:18 min)



Highlights: Huffman code

X	Code-word	Probability				
3	0 1	0.25	0.30	0.45	0.55	1
4	1 0	0.25	0.25	0.30	0.45	
1	1 1	0.20	0.25	0.25		
2	0 0 0	0.15	0.20			
5	0 0 1	0.15				

Dey Manjunath: Digital Communication - p.39/135

IIT MADRAS NMEICT

then we have seen what is Huffman code we have constructed Huffman code so the first step was to arrange the probabilities of the random variable ah of the {sym} (00:07:27 min) values of the random variable in decreasing order like this

and then combine these two probabilities into one and at the probabilities and arranges rest of the probabilities with that new probability in decreasing order again here like this

so we have got point three probability because {thi} (00:07:47 min) because point three is greater than all this probability points three goes to the top

and we add zero to this symbol and one to this symbol as two bits of the codewords then we ah keep repeating this process so whenever we add this ah add we want to add zero here so add zero this symbol and want to add one here so one goes to here and

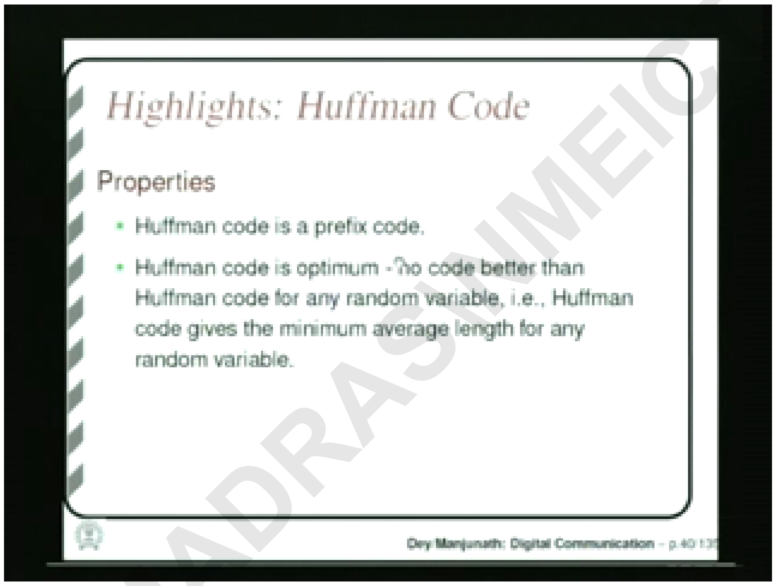
it goes on so whenever we want to add any bit to a derived probability we basically add zero on the left to all the symbols from which that probability came

for example this point three we want to add zero to this symbol this symbol came from these two symbols so we have added zero on the left to two both these symbols

so this is how we construct Huffman code

and we have seen some nice properties of Huffman codes

(Refer Slide Time: 00:08:470 min)



Highlights: Huffman Code

Properties

- Huffman code is a prefix code.
- Huffman code is optimum - no code better than Huffman code for any random variable, i.e., Huffman code gives the minimum average length for any random variable.

Dey Manjunath: Digital Communication – p.40/135

IITMADRASNMEICT

for example Huffman code is prefix code we have seen in in fact that in that example at least the Huffman code was a prefix code

and it can be proved that Huffman code will be always a prefix code

and Huffman code is optimum the sense that for any random variable we cannot design a better code then Huffman code

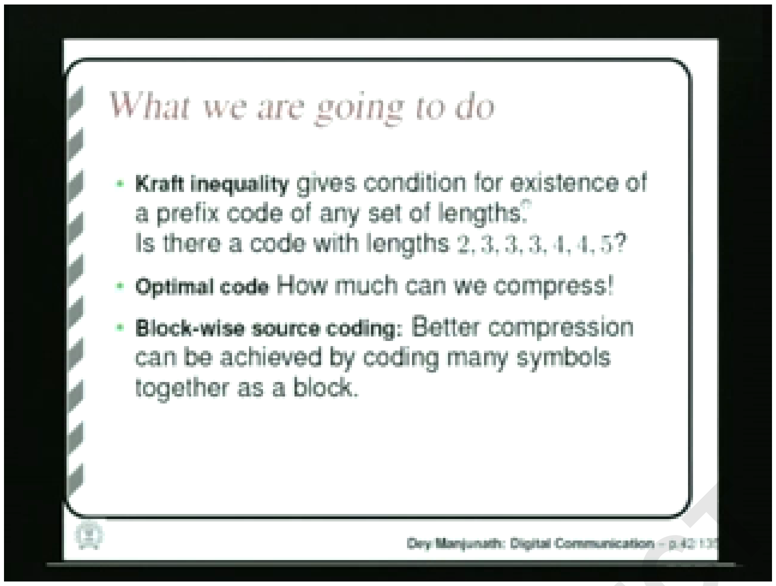
Huffman code will always give us the minimum average length per symbol

so Huffman code gives us the best compression for any random variable

so this is what we did in the last class

now in this class

(Refer Slide Time: 00:09:23 min)



What we are going to do

- **Kraft inequality** gives condition for existence of a prefix code of any set of lengths? Is there a code with lengths $2, 3, 3, 3, 4, 4, 5$?
- **Optimal code** How much can we compress!
- **Block-wise source coding:** Better compression can be achieved by coding many symbols together as a block.

Dey Manjunath: Digital Communication

we are going to do Kraft inequality which will give us a condition for existence of a prefix code for any set of lengths

so if we are given a set of lengths say i tell you that i want a prefix code with length three four four four four four four

so they should be six code words one should have length two five should have length four is there such a prefix code

so the Kraft inequality will answer such question

so it will give us a necessary and sufficient condition

so by checking if the length satisfy Kraft inequality you can say whether such a prefix code exists or not

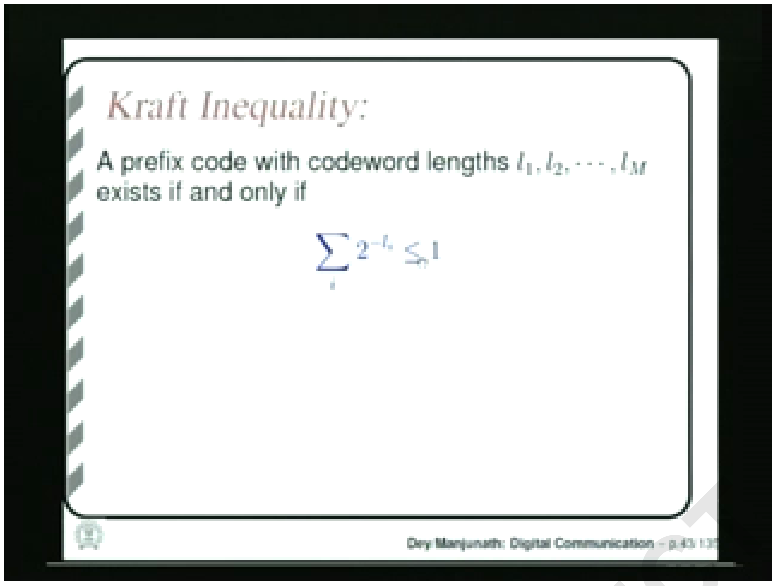
then we will talk about optimal codes which will answer this question that how much can we compress for any random variable

then we'll discuss block wise source coding there we will see that instead of coding every symbol individually we can code several symbols as a block together and that way we can achieve better compression

so here is Kraft inequality

(Refer Slide Time: 00:10:36 min)

IITMADRASINMEICT



Kraft Inequality:

A prefix code with codeword lengths $l_1, l_2, \cdots, l_M$ exists if and only if

$$\sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication – p.43/135

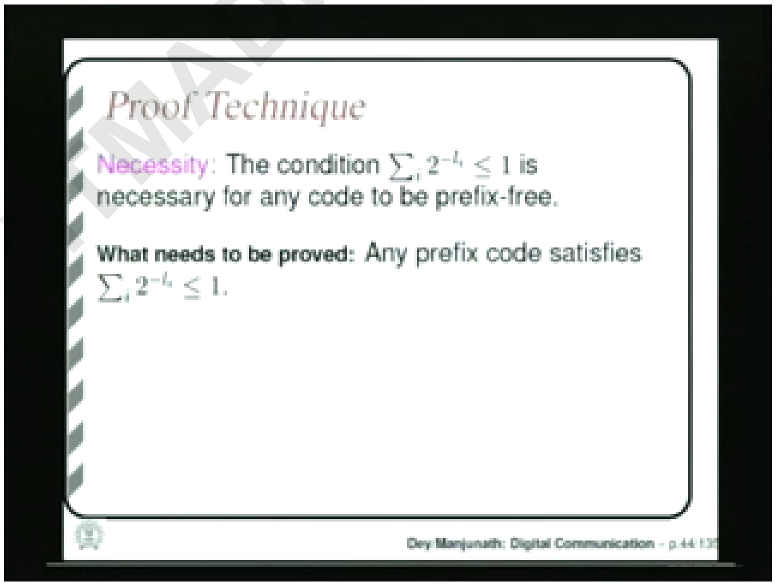
it is say that a prefix code with codeword lengths l one l two till lm this given lengths exists if and only if this condition is satisfied by these lengths

so if we are given a set lengths and if we want to ask the question whether a prefix code exists with these lengths then we need to check only this condition

if this condition is satisfied we know that this such a prefix code exists

if this is not satisfied we know that such a prefix code does not exist

(Refer Slide Time: 00:11:11 min)



Proof Technique

Necessity: The condition $\sum_i 2^{-l_i} \leq 1$ is necessary for any code to be prefix-free.

What needs to be proved: Any prefix code satisfies $\sum_i 2^{-l_i} \leq 1$.

Dey Manjunath: Digital Communication – p.44/135

so the proof of this inequality has two parts one is the so called necessity condition meaning by we need to prove that that condition is really necessary for existence of a prefix code

NPTEL IIT MADRAS

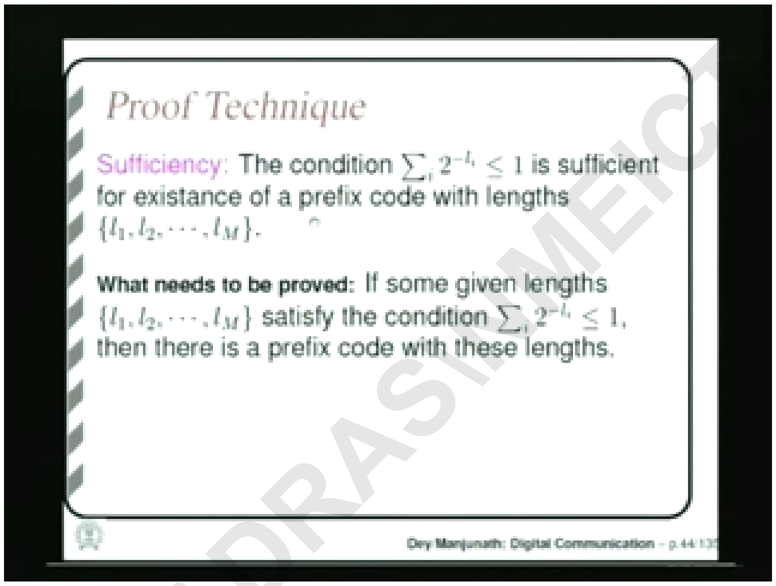
so for that what we need to prove we need to prove that any prefix codes will satisfy that condition

so if we prove that any prefix code satisfies this inequality then that will tell us that ah if a set of length does not satisfy the given condition the given inequality then there is no prefix code with those lengths because any prefix code should {condi} (00:11:57 min) satisfy that condition

so this is what we need to a prove in this part of the ah proof

so this is necessity of the condition for existence of a uh prefix code

(Refer Slide Time: 00:12:14 min)



Proof Technique

Sufficiency: The condition $\sum_i 2^{-l_i} \leq 1$ is sufficient for existance of a prefix code with lengths $\{l_1, l_2, \cdots, l_M\}$.

What needs to be proved: If some given lengths $\{l_1, l_2, \cdots, l_M\}$ satisfy the condition $\sum_i 2^{-l_i} \leq 1$, then there is a prefix code with these lengths.

Dey Manjunath: Digital Communication – p.44/135

IITMADRAS NMEICT

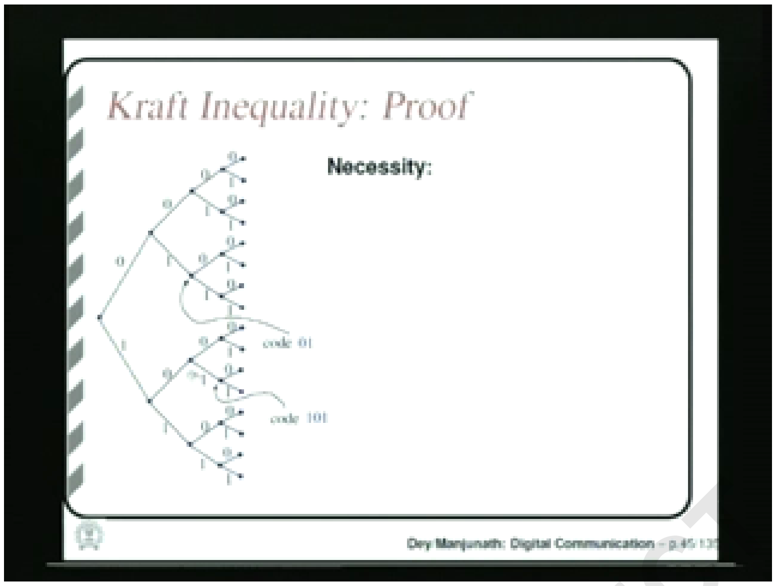
then we will prove sufficiency of the condition there we'll show that the condition in the inequality is really sufficient for a existence of prefix code

so for that what do we need to show we need to show that if a set of lengths satisfied the inequality then there is a prefix code with with those lengths

so that is what is meant by sufficiency which says the condition the inequality is really sufficient for existence of a prefix code

so we will prove in this part of the proof that in if some given lengths satisfy this condition then there is a prefix code with these lengths

(Refer Slide Time: 00:12:59 min)



Kraft Inequality: Proof

Necessity:

so we'll go on with the proof of the necessity first we first represent the whole possibilities for the codewords as a tree

so in this a binary tree each node from each node uh emanates two edges one leveled by zero the other level by one at every node there are two edges coming out of this one is zero the other is one

so this tree goes on even after this we have drawn on till the fourth level one two three four fourth level

and this tree actually goes on now in this tree every possible codeword is present for example this node represents a codeword zero one

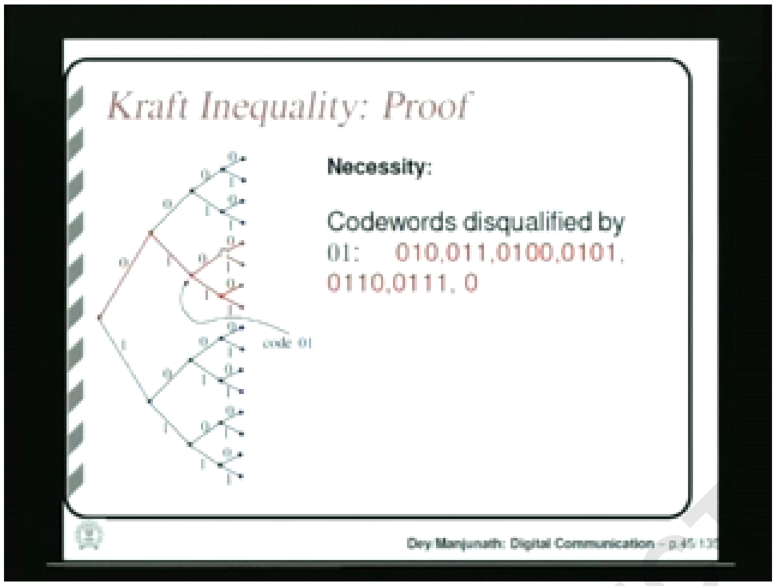
this node for example represents a codeword zero zero one so this is zero one codeword this is this is one zero one codeword

so you take the path from this origin node to this node you see all bits you come across and those bits give you the code corresponding that node

so all the possible codewords are represented by nodes in this particular tree okay

(Refer Slide Time: 00:14:30 min)

IITMADRASINMEICT


Kraft Inequality: Proof

Necessity:

Codewords disqualified by
01: 010,011,0100,0101,
0110,0111, 0

code 01
Dey Manjunath: Digital Communication

now what does prefix conditions signify in terms of this tree

so our codeword our code whatever code we are going to design or whatever code we have actually is a subset of all these nodes

so it is a our code is basically a set of nodes from this tree

so we are given a so this is uh recall that this this we are given a code and we are trying to prove that that code must satisfy the inequality given by the Kraft inequality

so if suppose this is a code codeword in our code this node is codeword that is zero one is a codeword in our code then we know that all these nodes cannot be codewords in our code because what is the codeword corresponding this node it is zero one zero of which zero one is a prefix

so if this is a code this node cannot be a codeword so any node which is coming after this node which is originating form here those nodes cannot be codewords if this node is because this node will be prefix of all the nodes coming out of this node

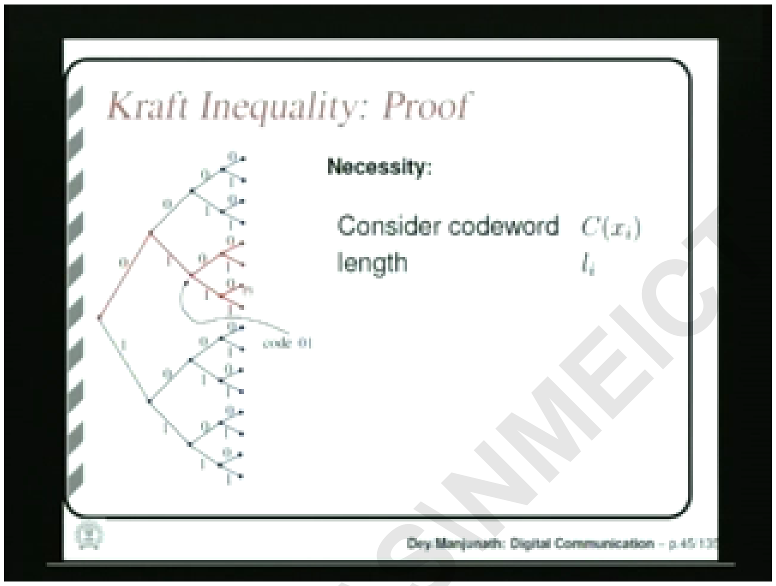
so all these nodes are disqualified as codewords if this node is a codeword

so zero one zero zero one zero this node zero one one zero one one this node then zero one zero zero zero one zero zero so this node this node this node and this node all these six nodes are disqualified as codewords if this is a codeword

and not only these are disqualified these nodes are also disqualified because this node for example this is the node corresponding to the codeword zero

IITMADRASNMEICT

this node cannot be also a codeword if this is a codeword because then this will be a prefix of this codeword and since our code or given code is prefix code this node cannot be a codeword
so if this a codeword in our code this node and all these nodes cannot be codewords
so these nodes are disqualified whenever you have this node has a codeword okay
(Refer Slide Time: 00:17:10 min)



Kraft Inequality: Proof

Necessity:

Consider codeword $C(x_i)$
length l_i

IITMADRASINMEICT

now considered a codeword C of xi xi is the value of the random variable of which C is the codeword
so C of xi is the codeword for xi and suppose it has length li okay
now we also assume that so so we are given the code now we have all these lengths l one l two till l capital L
now out of these lengths we have a maximum length so we need to expand the tree only till that level l max the maximum length
so till that level we need to expand the tree
now after we expand that the tree that much we considered each codeword
so each codeword has length less than equal to l max
so each codeword will be present in that tree before that level l max
so it will be one node from the tree now if we take that node that node is codeword
so that node disqualifies some other nodes as codewords
so it will disqualify in particular some nodes at the maximum level for example

if this is a codeword and we have expanded the tree till here fourth level suppose the our l max is the maximum length is four then we have expanded the tree till the fourth level and this is a codeword then it will disqualify four nodes at the maximum level one two three four

so we will count all these nodes at the maximum level that are disqualified by every codeword

so we take one codeword we count the number of nodes at the maximum level which are disqualified by that codeword

so how many codewords will be disqualified at that the maximum level by any codeword

if its length is li and the maximum level is a l max then the difference between this level and level is l max minus li

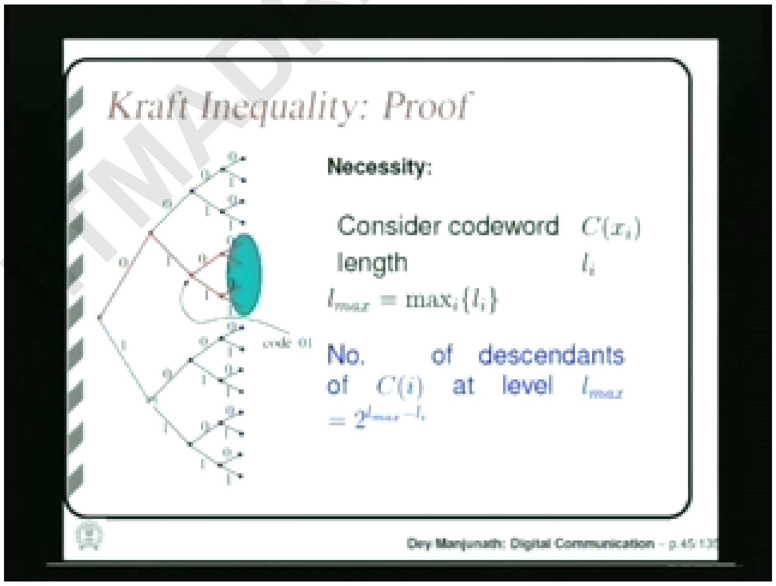
so the number of nodes that are disqualified that are coming out of this node here will be two to the power that because if the level difference is one

if you go one level from here there are two nodes so if you go two level from here there are four nodes two to the power two

if you go three three levels they will be eight nodes

so you it see the difference between this length and the maximum length two to the power that l max minus li is the number of nodes at the maximum level which are disqualified by this node

(Refer Slide Time: 00:20:02 min)



Kraft Inequality: Proof

Necessity:

Consider codeword $C(x_i)$

length l_i

$l_{max} = \max_i\{l_i\}$

No. of descendants of $C(i)$ at level l_{max} $= 2^{l_{max}-l_i}$

Dey Manjunath: Digital Communication – p.45/135

so if we now count all the nodes at the maximum level that are disqualified by different codewords

so let us see what happens

IITMADRASNMEICT

now the see that number of descendants here of C i of a of a given codeword at level l max is two to the power l max minus li as we have already observed

now you will count these numbers for all codewords and we know that if one codeword disqualified certain disqualifies certain nodes here

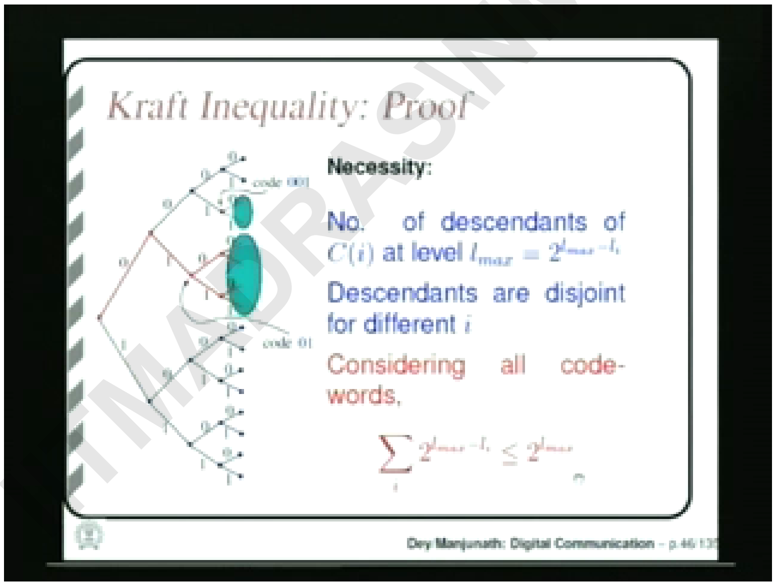
the another codewords suppose this one is a codeword and this will disqualify some other nodes and these nodes and these nodes cannot have any common node because this is the a prefix code this node if if only it was here or here they could have common descendants here

so they will be all different

so the descendants of this any other codeword will be different from this descendants of this codeword

so then the total number of nodes at the maximum level that are disqualified by all the codewords will be simply the sum of such numbers

(Refer Slide Time: 00:21:19 min)



Kraft Inequality: Proof

Necessity:

No. of descendants of $C(i)$ at level $l_{max} = 2^{l_{max}-l_i}$

Descendants are disjoint for different i

Considering all codewords,

$$\sum_i 2^{l_{max}-l_i} \leq 2^{l_{max}}$$

code 001
code 01

Dey Manjunath: Digital Communication – p.46/13?

IITMADRASNMEICT

so descendants are disjoint for different i so for different codeword the descendants are different for example if this a codeword this set is different this disjoint from this set they will intersect only when this node some where here or here and we know that this code prefix code so that cannot happen okay

so considering all the codewords what is the total number of nodes at the maximum level that are disqualified it is sum of two three power l max minus li for all i(s) and

we know that the total number of nodes at this level it self is two power l max so this number must be less than equal to this total number of nodes at this maximum level

so we have this inequality this inequality must be satisfied by any prefix code

so remember where we use the prefix condition we have used it to conclude that all the descendants of one codeword will be disjoint from all the descendants of another codeword

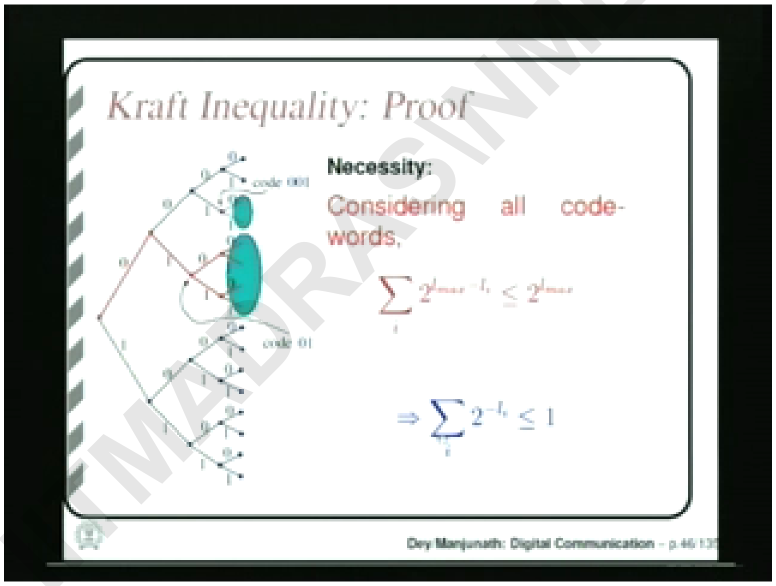
so these two are codewords their descendants will be disjoint because this is prefix no one is in the path of another

so this so this node cannot be here or here so their the descendants will be disjoint

so we have used the prefix condition there so the summation is less than equal to two to the power l max

now if we divided by two to the power l max both the sides left hand side and right hand side then we get the Kraft inequality

(Refer Slide Time: 00:23:05 min)



Kraft Inequality: Proof

Necessity:

Considering all codewords,

$$\sum_i 2^{l_{max} - l_i} \leq 2^{l_{max}}$$

$$\Rightarrow \sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication - p.46/13

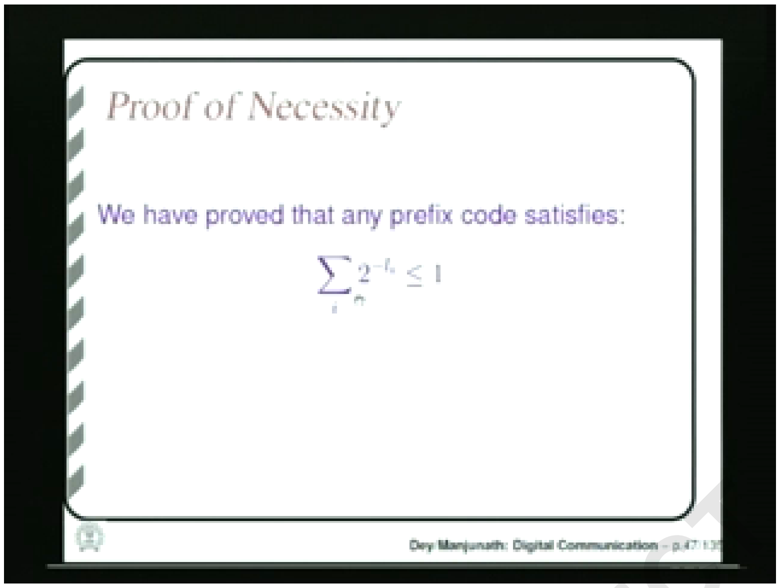
code 001
code 01

IITMADRAS\NMEICT

so we have this inequality now dividing by l max both sides we get summation two to the power minus li is less than equal to one

so we have proved the necessary necessity condition here we have shown that any prefix code must satisfy this condition

(Refer Slide Time: 00:23:23min)



Proof of Necessity

We have proved that any prefix code satisfies:

$$\sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication – p.47/135

so we have proved any prefix code must satisfy this condition

so remember again the necessity proof of necessity requires it needs to be shown that the condition is necessary to be satisfied by any prefix code

so we have proved that any prefix code must satisfy that so that to do something we have proved that the condition is really necessary for any prefix code okay

now we go on to the proof of sufficiency again we use this tree and ah we are now remember what we are going to prove now

we are given these lengths and these lengths are given in such a way that the satisfy the Kraft inequality

so we are going to now show that there is a prefix code with these lengths

so then what are you what will it prove if i show that it will prove that there this condition that Kraft inequality it is really ah a sufficient condition a for a for a prefix code to exist with those lengths

so in the previous part we have proved that a prefix code must satisfy Kraft inequality but now we are going to prove that if a set of given lengths satisfy Kraft inequality then there is a prefix code with those lengths

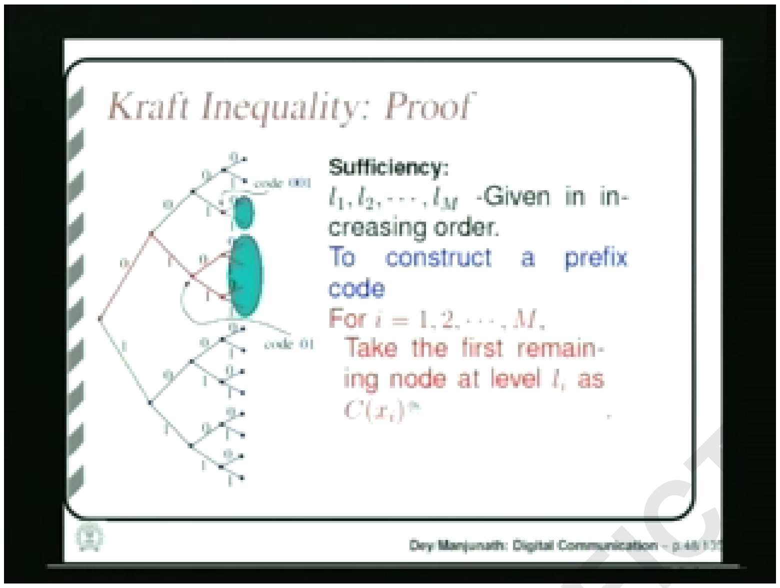
so this condition is really also sufficient for any prefix code to exist with those lengths

so we now try to prove that

so we first arrange the lengths in increasing order and we are in fact going to construct a code construct a code which will have these lengths

IITMADRASNMEICT

(Refer Slide Time: 00:25:14 min)



Kraft Inequality: Proof

Sufficiency:
$l_1, l_2, \cdots, l_M$ -Given in increasing order.
To construct a prefix code
For $i = 1, 2, \cdots, M$,
Take the first remaining node at level l_i as $C(x_i)$

code 001
code 01
0
1

Dey Manjunath: Digital Communication

so these lengths are given to us and it is told to us that these lengths satisfy Kraft inequality

then we are going to show that there is a prefix code

so how will we show that we will show that by actually constructing a prefix code

so first we arrange these lengths in increasing order then we take the first length

so how do you go about constructing a prefix code as below

so we take one by one we construct one by one codeword

so we take i equal to one to M so we need to construct M codewords

so we take the first length so remember that we are going to pick codewords from this this tree

so we will expand the tree till the maximum length that many levels and then l one is the minimum of this

so we take the that codeword will be at the lowest level on the left most level somewhere here

so we will take the first node from top at level l one

so suppose l one is one what we will do will go to the first level at first level there are two nodes we will take the first one from top this one

if l one was two we will go to level two here one two here

so at level two there are four nodes we will take the top most this one

so we will take the first node at level l one as the first codeword that will have the minimum length

IITMADRASNMEICT

then we'll so this will disqualify all the codewords here and this will disqualify all the codewords here also but we are not going pick any codeword from this anyway because any codeword here before this level will have smaller length and we don't have any smaller length than l one

so we don't need to construct any codeword of length less than l one

so these nodes are not of our interest these are disqualified by this node but ah we are not going to pick any codeword from here anyway okay

so this node disqualifies all these nodes at the later levels

then we will pick l two l two may be equal to l one or may be greater than l one

so if it is equal to a l one we'll take the next node available at this level so this one

so obliviously it will disqualifies some other nodes they will not intersect with this

so and this will not be a prefix of this and this also will not be a prefix of this note and if this l two is greater than this then we will take we will go to that level l two level

say suppose it is three l one is two l two is three then we will go to this level and then choose the first node available

so what do you mean by the first node available remember that some nodes at this level might be disqualified by already chosen codewords

for example if we have chosen this codeword then this has disqualified this two nodes at level three

so we cannot chose this level this node at the next codeword

so we chose the node at this level that available first node that is available

that is not disqualified so we take that and we keep doing that this will disqualify these two nodes and then again we take the next codeword similarly

so ((where)) are you using the condition given

so that condition is actually telling us that there is some node available at least

suppose at level ah level three you observe that there is no node available after taking these two codewords there is no node available at level three or or or for length l three at level l three there is no node available

so what do you do then well then we cannot choose another codeword but this condition will guarantee that there will be some node available

how

IITMADRASNMEICT

so that is as following remember that if there is no node available then there {low} (00:29:55 min) there will not be any node available at the maximum level as well

if all the nodes if if there is some node available at the maximum level then there will be something {aval} (00:30:08 min) available before also because if this is available then this will also be available

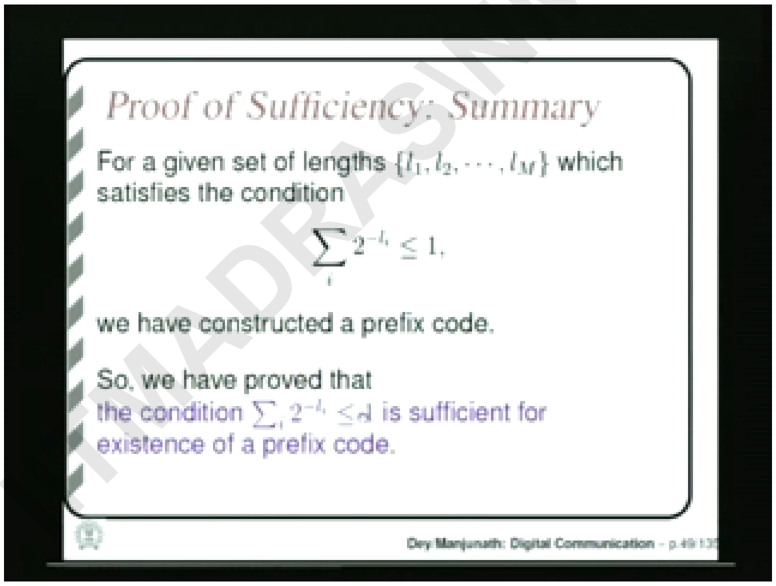
so so now keep counting the {maxim}(00:30:23 min) keep counting the nodes that are disqualified at the maximum level by already chosen codewords

and while we do that we can check that there will always be till we finish there will always be some nodes available at the maximum level

so this will guarantee that there will always be codewords available to be chosen till we finish okay

so this will give us some construction of a prefix code

(Refer Slide Time: 00:31:00 min)



Proof of Sufficiency: Summary

For a given set of lengths $\{l_1, l_2, \cdots, l_M\}$ which satisfies the condition

$$\sum_i 2^{-l_i} \leq 1,$$

we have constructed a prefix code.

So, we have proved that the condition $\sum_i 2^{-l_i} \leq 1$ is sufficient for existence of a prefix code.

Dey Manjunath: Digital Communication – p.49/135

IIT MADRAS NMEICT

so what we have done is to prove the sufficiency we have taken a set of lengths which satisfies this condition and if it satisfies this condition then we have shown that there is a prefix code with these lengths by construction

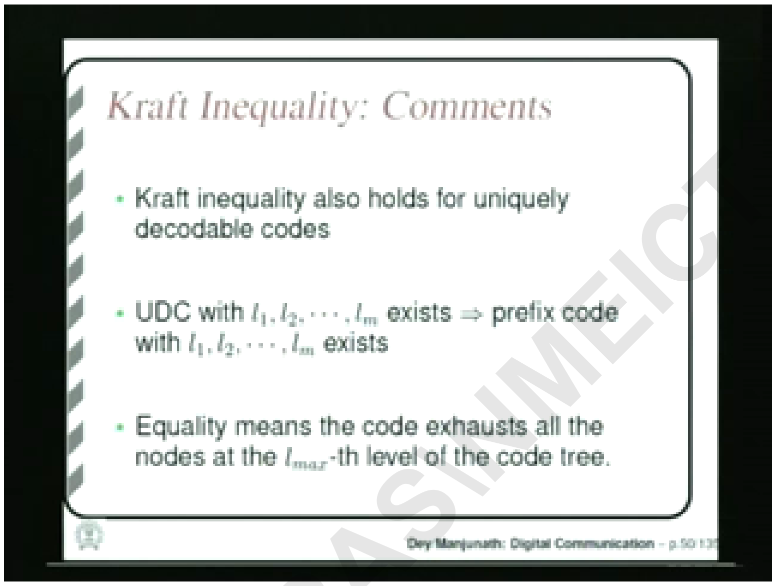
so we have proved that this condition is sufficient for existence a prefix code

so here are some ah comments about Kraft inequalities so so far we have proved Kraft inequality we have proved that Kraft inequality is really necessary as well as sufficient condition for existence of a prefix code if a set of given lengths is uh if a set of length satisfy Kraft inequality then we have shown that ah uh there is prefix code with those lengths

so the condition is sufficient and also we have shown that if there is prefix code with certain lengths that must satisfy the Kraft inequality so it is necessary also

so Kraft inequality satisfies uh

(Refer Slide Time: 00:32:06 min)



Kraft Inequality: Comments

- Kraft inequality also holds for uniquely decodable codes
- UDC with $l_1, l_2, \cdots, l_m$ exists $\Rightarrow$ prefix code with $l_1, l_2, \cdots, l_m$ exists
- Equality means the code exhausts all the nodes at the l_{max}-th level of the code tree.

Dey Manjunath: Digital Communication – p.50/135

necessary as well as sufficient condition now here are some more comments about Kraft inequality

Kraft {inequalities} (00:32:14 min) Kraft inequality also holds for uniquely uniquely decodable codes which means that so far we have {see} (00:32:23 min) we have seen that Kraft inequality ah is necessary and sufficient condition for existence of a prefix code

so it does not guarantee that it is also necessary for uh existence of a uniquely decodable code

it is sufficient certainly because if a set of lengths if a set lengths satisfy Kraft inequality we know that there is a prefix code with those lengths

so if there is a prefix code that prefix code itself is a uniquely decodable code as well

so Kraft inequality is certainly a sufficient condition for existence of a uniquely decodable code also that is trivial but that does not mean that it is also necessary

if may happen that uh a uniquely decodable code exists even if that Kraft inequality is not satisfied can that happen

so this comment says that it is really necessary even for existence of a uniquely decodable code

IITMADRAS NMEICT

so this in turn means that whenever there is a uniquely decodable code with certain lengths there is also a prefix code with those lengths

how because if there is a uniquely decodable code with these lengths then we know that [vocalized- noise] uh we know that Kraft inequality is satisfied by these lengths

and so there must be also a prefix code with these lengths

so whenever there is a uniquely decodable code with these lengths with certain lengths there is also a prefix code with the same lengths

so this tells us something interesting it tells us that we should never use a code which is not prefix code because whenever we are using a code which is not prefix code but it must be uniquely decodable code of course because we need to use it for practical purpose we must be able to decode successfully

so whenever we are a using code which is uniquely decodable code but not prefix code then we know that there is a code with same lengths which is also prefix

so why not use that prefix code it self instead of using the uniquely decodable code

so this tell us that we should always use prefix code instead of using simply uniquely decodable code okay

now okay so what does equality mean in the code so if the Kraft inequality is satisfied with equality by some lengths

so we are given a set of lengths and uh the Kraft inequalities satisfied with equality for these lengths meaning by the both left hand side and right left hand side becomes equal to one for those lengths

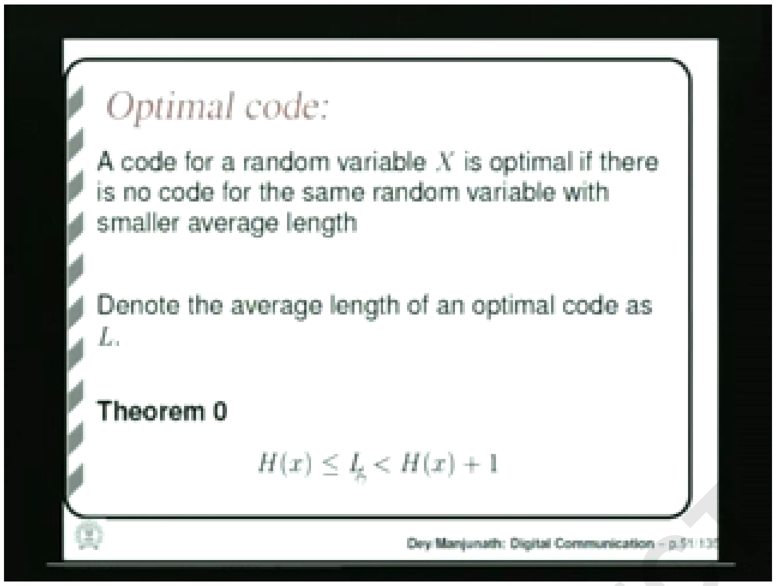
and what does it mean it means that at the l max th level of the code tree all the nodes will be exhausted

all the nodes will be disqualified after you choose all the codewords no node at the level l max will be left free after you choose all the code nodes codewords nodes

okay now we go into optimal codes

(Refer Slide Time: 00:35:47 min)

IITMADRASNMEICT



Optimal code:

A code for a random variable X is optimal if there is no code for the same random variable with smaller average length

Denote the average length of an optimal code as L.

Theorem 0

$$H(x) \leq L < H(x) + 1$$

Dey Manjunath: Digital Communication - p.51/135

so so far we have seen that ah Kraft inequality is necessary as well as sufficient condition for existence of a prefix code as well as for existence of a uniquely decodable code

now we have also seen that uh we have also seen in that previous page that equality in the Kraft inequality means that all the nodes at code tree at level l max the maximum length will be disqualified after you choose all the codewords from the code tree all the codewords in your code okay

so now let us go ahead to the next topic we'll now discuss on optimal codes

so we'll try to answer the question how much compression is possible

so we are given a random variable we know the statistics of random variable how much can we compress the random variable okay

so first of all we will define what is called of optimal code we have already seen we have already described Huffman code as an optimal code

so what you mean by optimal code it simply means that a code for random variable X is optimal if there is no code for the same random variable with smaller average length

so this code achieves the minimum average length that is possible to achieve for that particular random variable

so that is what is meant by optimal code that means that particular code will gives us the maximum compression on average okay for that particular random variable okay

so now ah we'll denote the average length of an optimal code as L from now on

IITMADRASNMEICT

so so remember that L is not the average length of any given code it is the it will denote the length of an optimal code

so this length will be the minimum achievable length by all possible codewords uh all possible codes

so here is a very interesting result and very useful result about optimality it says that the minimum achievable length for any random variable lies in between these two limits

so we have uh a previously information theory we have seen that ah this H x denotes the entropy of the random variable x ah is basically the average information content in the random variable

so and we have also seen that Shannon's source coding theorem says this is the minimum length to which a source can be compressed but that will require ah block wise encoding as we will see later but if you do encoding symbol wise

if you take a every symbol individually and encode it separately every symbol independently then what is the best you can do that best is given by this theorem that that best the minimum average length will be in between these two it is not always possible to achieve this

so this is the minimum this may not be possible to be achieved for any random variable in general it is possible to get a code which will give us average length less than H x plus one but of course its cannot be less than H x because of source coding theorem

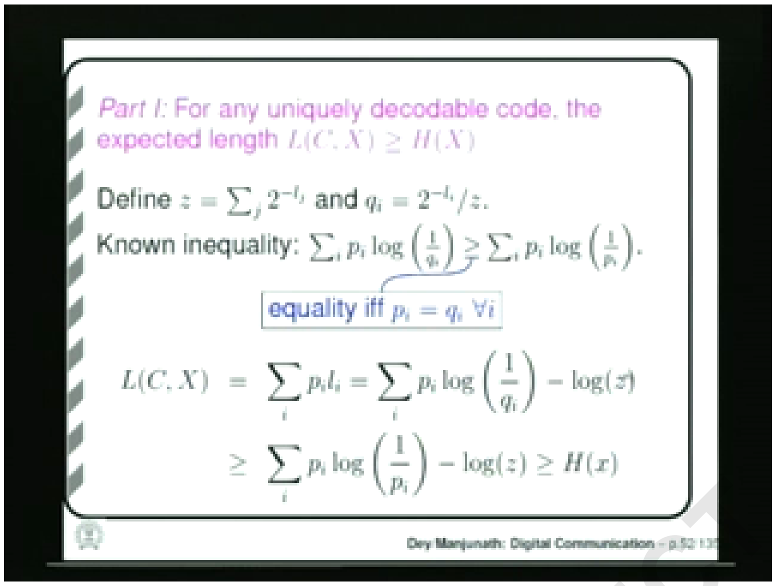
source coding theorem says it cannot be less than H x

so it is in this range

so from here we can say that Huffman codes length will be certainly in this length in these two limits in between because it is an optimal code

(Refer Slide Time: 00:39:51 min)

IITMADRASNMEICT



Part I: For any uniquely decodable code, the expected length $L(C,X) \geq H(X)$

Define $z = \sum_j 2^{-l_j}$ and $q_i = 2^{-l_i}/z$.

Known inequality: $\sum_i p_i \log\left(\frac{1}{q_i}\right) \geq \sum_i p_i \log\left(\frac{1}{p_i}\right)$.

$\boxed{\text{equality iff } p_i = q_i \ \forall i}$

$$\begin{aligned} L(C,X) &= \sum_i p_i l_i = \sum_i p_i \log\left(\frac{1}{q_i}\right) - \log(z) \\ &\geq \sum_i p_i \log\left(\frac{1}{p_i}\right) - \log(z) \geq H(x) \end{aligned}$$

Dey Manjunath: Digital Communication – p.52/130

so we will prove this theorem we will prove this theorem

so while proving this we will prove first that L is greater than equal to H x and also we will prove that L is less than Hx plus one

so there are two parts

so when you try to prove that L is greater than equal to H x we'll not actually prove that optimal length is greater than equal to H x we'll prove that any code's average length will be greater than equal to H x

so you give me any code for a random variable i can show that it will have average length greater than equal to H x

so it will be also true for optimal codes because it is true for any code

so we will prove that part for any code that is L is greater than equal to H x and then we will prove that L is less than equal to H x plus one

and here we'll prove for the optimal code because if you give me any code it may not have average length less than H x plus one it may be a very lousy code

so we'll prove that there is a code at least one code which will have average length less than H x plus one

so so the proof will be in two parts so first part for any uniquely decodable code the expected length is greater than equal to H x

so so we are given uniquely decodable code we'll prove that its length must be greater than equal to H x

IITMADRASINMEICT

so first we add all these quantities remember this quantity is basically the left hand side of the Kraft inequality this is represented by z because it is uniquely decodable we know that this must be less than equal to one because Kraft inequality also satisfied by uniquely decodable codes

so this is uh less than equal to one

now we will also denote by Qi this quantity two power minus li by z

so this z is a summation of all such two for minus li (s) and we take the i th one here there is qi this by ah summation of two power minus li

so what is the summation of qi (s) summation of qi (s) will be equal to one so it is like a probability distribution probability mass function qi

so we will already have probabilities of probabilities pi for the random variable that unknown

now we construct another mass function qi this way from the lengths of the given code okay

now this is a known equality we'll not prove but we'll assume this that summation pi log one by qi for any any qi in fact not only for this any qi satisfying summation qi equal to one this true that summation pi log one by qi is greater than equal to summation pi log one by pi

so if you replace pi by qi here so this this you know to be the entropy right but ah if you represent this pi by qi this quantity will increase that is the inequality okay

so and also this equality is satisfied if an only if pi is equal to qi so this side will be equal to this side only if pi is equal to qi okay

so this is a very important inequality and we'll use this to prove this theorem

so what is the length of the ah code average length of the code it is defined as summation pi li the average length so this is equal to summation pi and then li can be now replaced by this quantity

so how do we get this this is basically you can get from here

so what is li

li is one by qi will be two to the power minus so z times qi will be equal to two the power minus li okay

so one can show that li can be expressed in this form if we replace li from here we'll get this now here again we'll use this inequality first

we'll we'll know that this quantity is greater than equal to this quantity so you replace this by so you replace this by p

so this is less than this quantity from here and this log z is as it is here okay

IITMADRAS NMEICT

now this again greater than equal to H x because what is log z z is less than equal to one from Kraft inequality

so log z is less than zero because log z is less than zero minus log zero is greater than zero

so if we remove this something greater than zero it will decrease so this quantity is greater than equal to simply this quantity which is H x

so we have shown that this average length of any uniquely decodable code is greater than equal to H x

so we will just repeat again so here we have z is the left side left hand side of the Kraft inequality for the given uniquely decodable code and qi is constructed this way and we use this known inequality

then we replace this li from here we get this and from here we use the this inequality so replace qi by pi here and then log z is less than zero can be used to show that this is greater than equal to H x okay

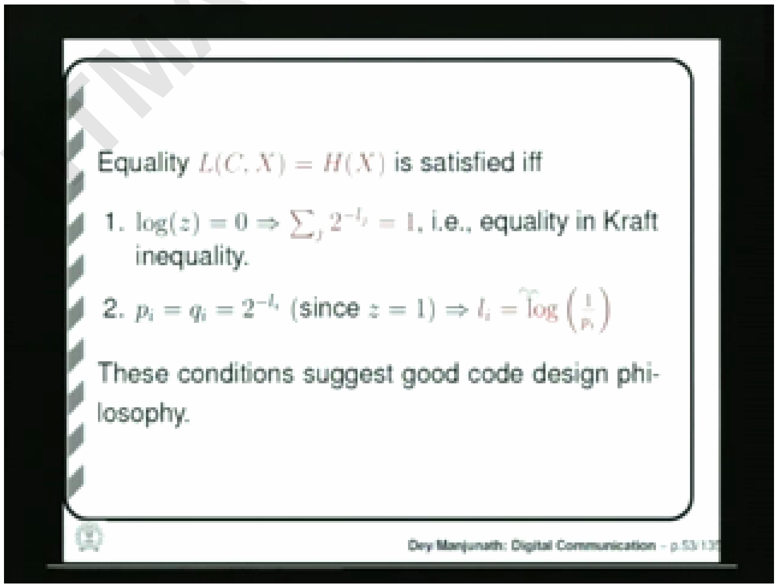
so this is proved but there are something uh some things to be observed in this proof that is when is this equal to H x when can we achieve the maximum compression

so the the that can be seen from the proof itself that ah what is what are the things that is to be satisfied to get equality in this inequality

so what are things that is to be observed

NPTEL MADRAS NMEICT

(Refer Slide Time: 00:46:16 min)



Equality $L(C,X) = H(X)$ is satisfied iff

1. $\log(z) = 0 \Rightarrow \sum_j 2^{-l_j} = 1$, i.e., equality in Kraft inequality.
2. $p_i = q_i = 2^{-l_i}$ (since $z = 1$) $\Rightarrow l_i = \log\left(\frac{1}{p_i}\right)$

These conditions suggest good code design philosophy.

Dey Manjunath: Digital Communication – p.53/13

first log z should be zero

so that means Kraft inequality should be satisfied this should be equal to one so Kraft inequality is should be satisfied

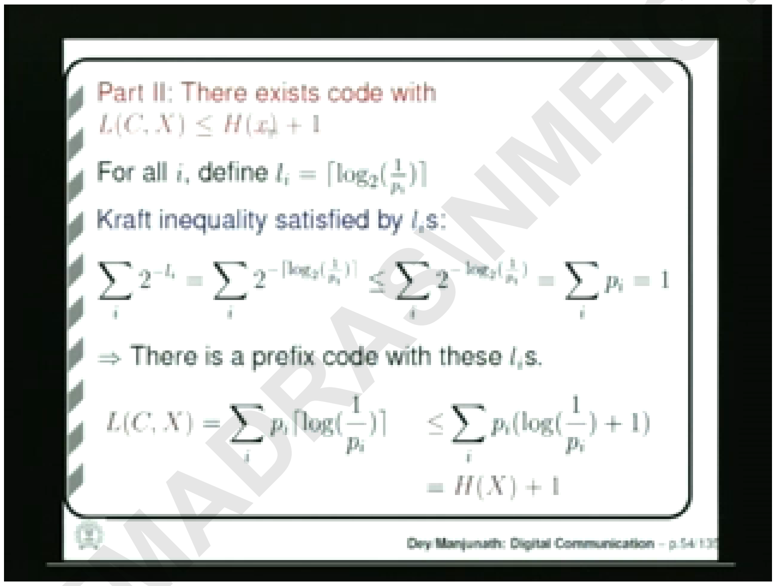
and then pi should be equal to qi

so li the length of the code should be equal to log of one by pi

so these conditions will suggest that how to design good code because to achieve good length small length this condition should be satisfied the Kraft inequality should be satisfied with equality

we should try to achieve as near one as possible this quantity and also length should be almost equal to this to get good code

(Refer Slide Time: 00:46:54 min)



Part II: There exists code with $L(C,X) \leq H(x) + 1$

For all i, define $l_i = \lceil \log_2(\frac{1}{p_i}) \rceil$

Kraft inequality satisfied by l_is:

$$\sum_i 2^{-l_i} = \sum_i 2^{-\lceil \log_2(\frac{1}{p_i}) \rceil} \leq \sum_i 2^{-\log_2(\frac{1}{p_i})} = \sum_i p_i = 1$$

$\Rightarrow$ There is a prefix code with these l_is.

$$L(C,X) = \sum_i p_i \lceil \log(\frac{1}{p_i}) \rceil \leq \sum_i p_i (\log(\frac{1}{p_i}) + 1)$$
$$= H(X) + 1$$

Dey Manjunath: Digital Communication – p.54/135

then we'll prove that there exist codes with L C X less than equal to H x plus one

so we will construct a code which will have this less than equal to this length first of all we will define the lengths let us define this way

that li is equal to the ceiling ceiling is the minimum number which is greater than minimum integer that is greater than this number

so if this number is two point five this ceiling will be three the next integer

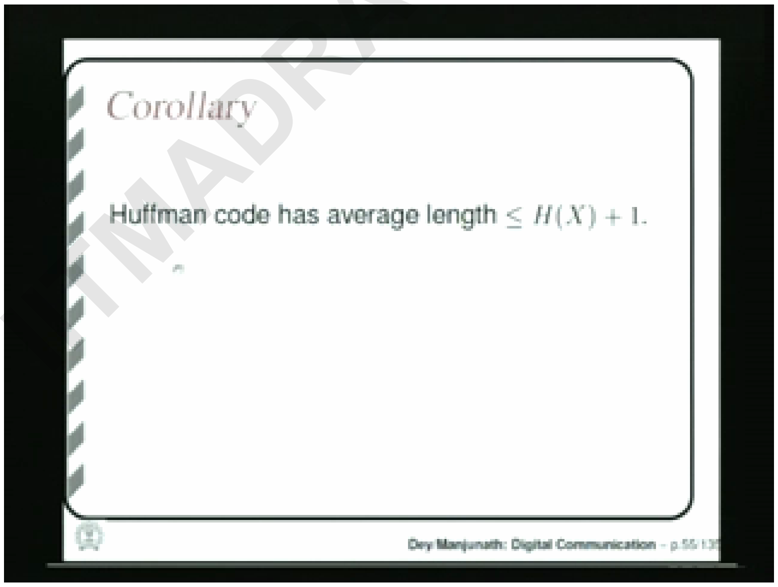
so define this length this way then we know that this we can check that these lengths will satisfy Kraft inequality

so this summation will be equal to this summation li is replaced by this and this will be less than equal to

IITMADRASINMEICT

if we remove the ceiling what will happen if we remove the ceiling this quantity will decrease
so minus of this quantity will increase so two to the power that will increase so this will increase
so this is less than equal to this and this is two to the power two to the power minus log two one by pi
so minus log two one by pi is log two pi so two to the power log two pi is simply pi
so summation pi is one so this is less than equal to one so it is satisfies the Kraft inequality these lengths as derived this way satisfy the Kraft inequality
so there is a code prefix code with these lengths and for this prefix code what is the average length of this prefix code it is summation pi li
and this is less than equal to this quantity plus one the two point five ceiling three must be less than equal to two point five plus one
so similarly this is ceiling is less than equal to this plus one
so now when you take summation you get this is less than equal to H x plus one
so there is a code which there is there is a prefix code which satisfies this condition which achieves this length less than equal to H x plus one
(Refer Slide Time: 00:49:02 min)

IITMADRASINMEICT



Corollary

Huffman code has average length $\leq H(X) + 1$.

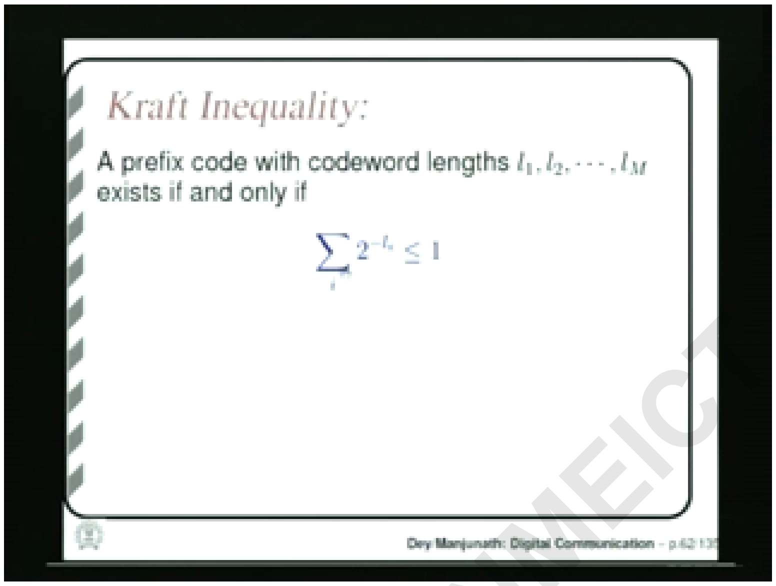
Dey Manjunath: Digital Communication – p.55/135

so the Corollary of this theorem is that Huffman code has average length less than equal to H x plus one because it is optimal code okay
so ah so we have discussed Kraft inequality and we have discussed ah we have discussed Kraft inequality and we have discussed what is optimal code and how much is possible to ah

how much is possible to compress for any random variable and ah

so we have we will just see

(Refer Slide Time: 00:49:43 min)



Kraft Inequality:

A prefix code with codeword lengths $l_1, l_2, \cdots, l_M$ exists if and only if

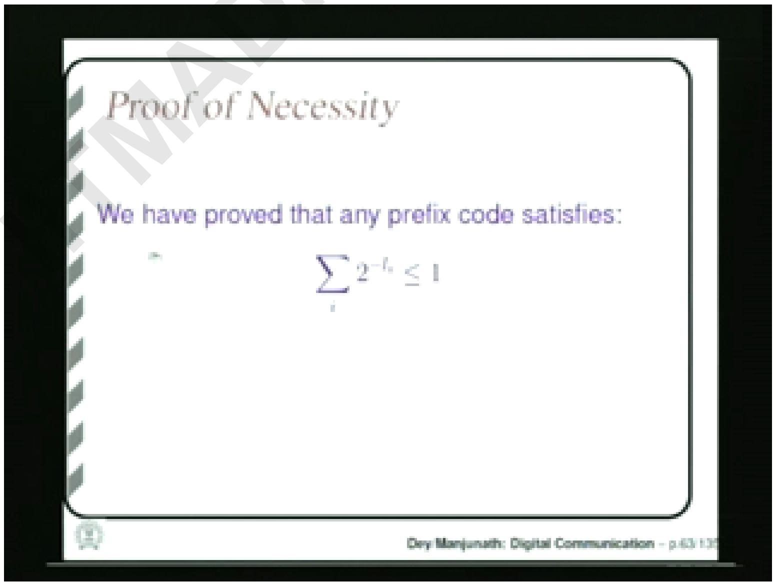
$$\sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication – p.62/135

what we have done in this class we have first seen what is Kraft inequality we have seen that this is the necessary and sufficient condition for a existence of a prefix code

we have proved

(Refer Slide Time: 00:49:55 min)



Proof of Necessity

We have proved that any prefix code satisfies:

$$\sum_i 2^{-l_i} \leq 1$$

Dey Manjunath: Digital Communication – p.63/135

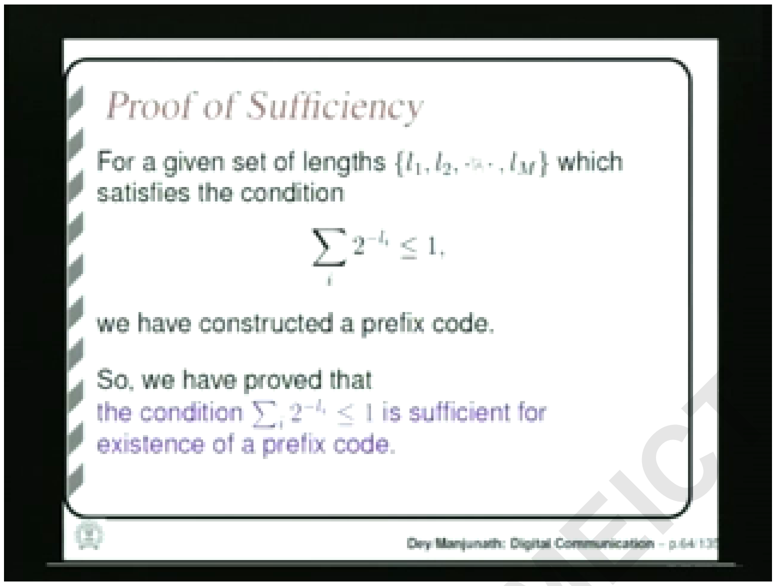
the ah inequality in two parts necessity and sufficiency

so this is necessity condition and so we have proved that any prefix code should satisfy this conditions

IITMADRASNMEICT

so this this condition is necessary

(Refer Slide Time: 00:50:10 min)



Proof of Sufficiency

For a given set of lengths $\{l_1, l_2, \cdots, l_M\}$ which satisfies the condition

$$\sum_i 2^{-l_i} \leq 1,$$

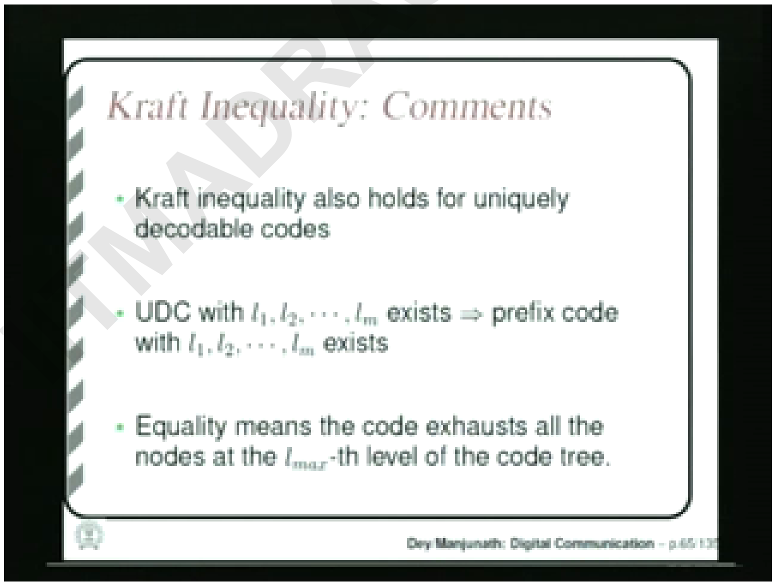
we have constructed a prefix code.

So, we have proved that the condition $\sum_i 2^{-l_i} \leq 1$ is sufficient for existence of a prefix code.

Dey Manjunath: Digital Communication – p.64/13

and then we have proved that this condition is also sufficient ah in a sense that if some lengths satisfy this condition then there is a prefix code with these lengths

(Refer Slide Time: 00:50:24 min)



Kraft Inequality: Comments

- Kraft inequality also holds for uniquely decodable codes
- UDC with $l_1, l_2, \cdots, l_m$ exists $\Rightarrow$ prefix code with $l_1, l_2, \cdots, l_m$ exists
- Equality means the code exhausts all the nodes at the l_{max}-th level of the code tree.

Dey Manjunath: Digital Communication – p.65/13

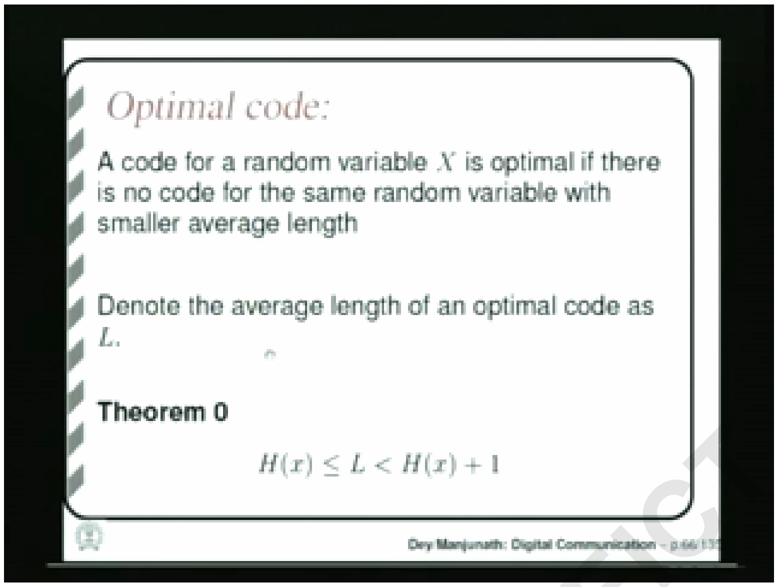
NPTEL IIT MADRAS

so we have seen that the Kraft inequality is also true for uniquely decodable codes it is necessary as well as sufficient for distance of uniquely decodable codes also

and that meant that if there is a uniquely decodable code with a set of lengths there is also prefix code with those lengths

then we have seen what equality means in Kraft inequality

(Refer Slide Time: 00:50:54 min)



Optimal code:

A code for a random variable X is optimal if there is no code for the same random variable with smaller average length

Denote the average length of an optimal code as L.

Theorem 0

$$H(x) \leq L < H(x) + 1$$

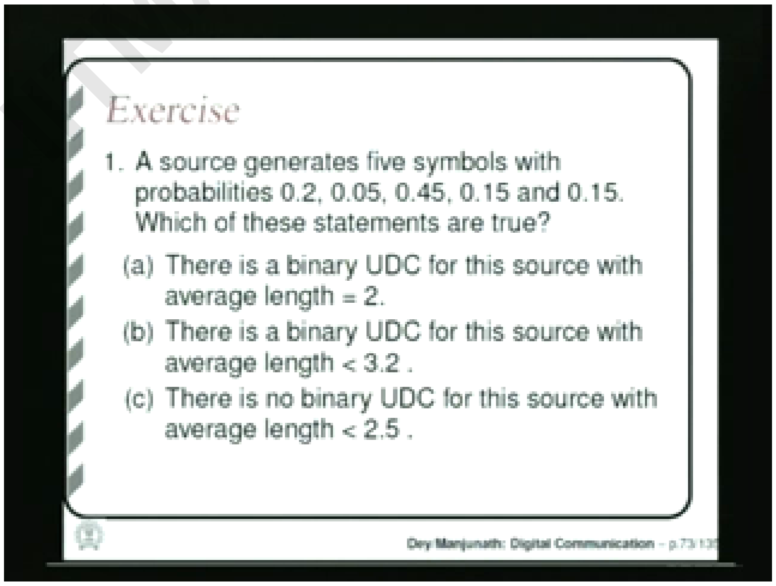
Dey Manjunath: Digital Communication – p.66/135

than we have seen a very important result about optimal code it says that the optimal length average length of a code ah ah for any random variable is in will be in between these two

so any codes average length will be greater than equal to H x and there is code with average length less than H x plus one okay

so so in this class we have we have uh done Kraft inequality and we have discussed about Huffman uh we have discussed about optimal codes we have seen that Huffman codes satisfy the uh uh optimality condition and we have some exercises

(Refer Slide Time: 00:51:41 min)



Exercise

1. A source generates five symbols with probabilities 0.2, 0.05, 0.45, 0.15 and 0.15. Which of these statements are true?
   (a) There is a binary UDC for this source with average length = 2.
   (b) There is a binary UDC for this source with average length < 3.2 .
   (c) There is no binary UDC for this source with average length < 2.5 .

Dey Manjunath: Digital Communication – p.73/135

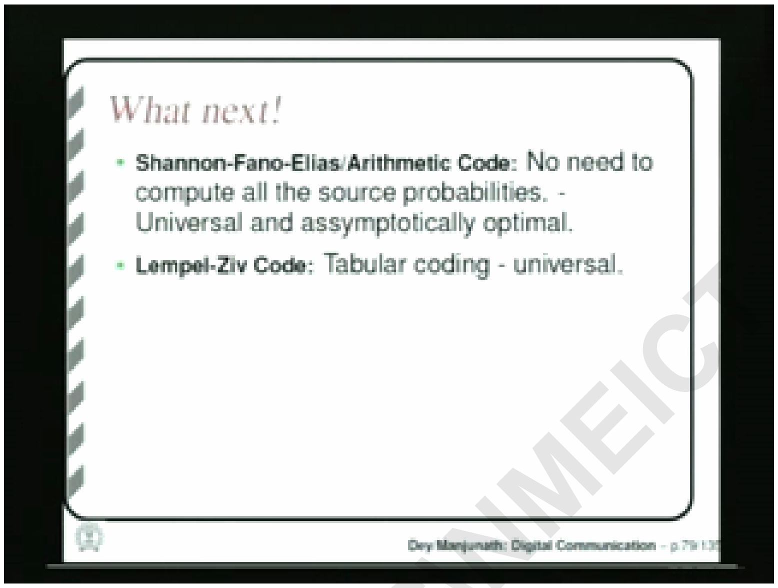
so these are some exercises we will solve in the next class

IIT MADRAS NPTEL

and the topics which we have covered in this class can be found in the same reading text we have given in the last time that is Thomas and Coover and McKay

and we will do in the next class Shannon-Fano-Elias code

(Refer Slide Time: 00:52:01min)



What next!

- **Shannon-Fano-Elias/Arithmetic Code:** No need to compute all the source probabilities. - Universal and assymptotically optimal.
- **Lempel-Ziv Code:** Tabular coding - universal.

Dey Manjunath: Digital Communication - p.79/135

that is also called arithmetic code and than Lempel Ziv code these codes are very different from Huffman codes

we will see in detail in the next class

so we have done in this class Kraft inequality optimal codes we have not done this block wise source coding we will do in the next class

and we will also do Shannon-Fano-Elias arithmetic code and Lempel Ziv code in the next class

IITMADRASNMEICT