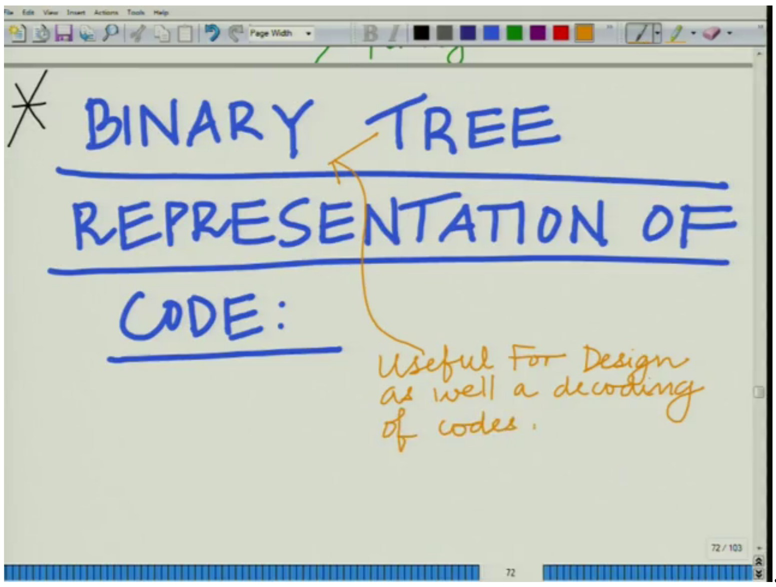
Principles of Communication Systems - Part II

Prof. Aditya K. Jagannatham
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 43
Binary Tree Representation of code, Example Kraft Inequality

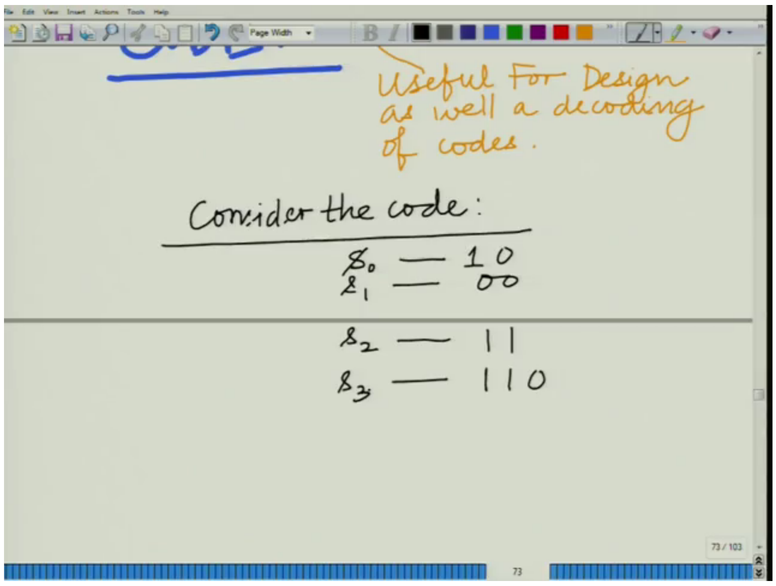
Hello, welcome to another module in this massive open online course. So, so far we are looking at source coding and we have looked at different kinds of codes. And we said that we are interested in the design of uniquely decodable codes and in particular prefix free or instantaneous codes all right. So, we are interested in design of prefix free or instantaneous which form a subset of uniquely decodable codes. Now let us look at a new concept coding in a of course, in a the design and as well as decoding of a such prefix free codes. And this is the binary tree representation of a code ok.

(Refer Slide Time: 00:50)



So, in this module we would like to start looking at a binary tree Representation of a code. And as I have told you this is useful for useful for in design as well, the design as well as decoding design as well as decoding of course, it is useful for the design as well as decoding of the codes.

(Refer Slide Time: 02:10)



Useful for Design as well a decoding of codes.

Consider the code:

s_0 — 1 0

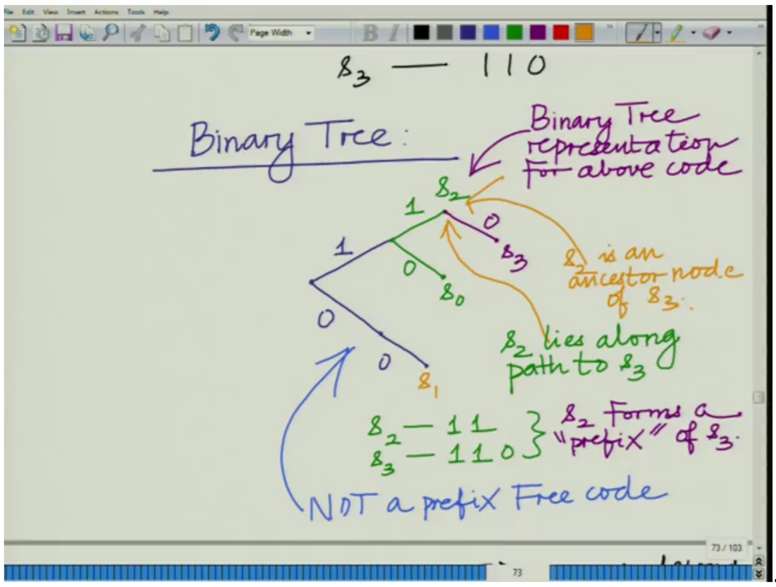
s_1 — 0 0

s_2 — 1 1

s_3 — 1 1 0

For instance consider the code for symbol code s naught represented by 1 0, s 1 represented by 0 0, s 2 represented by 1 1 and s 3 represented by 1 0. Now if you look at this code.

(Refer Slide Time: 02:49)



s_3 — 1 1 0

Binary Tree:

Binary Tree representation for above code

s_2 is an ancestor node of s_3.

s_2 lies along path to s_3

s_2 — 1 1

s_3 — 1 1 0 } s_2 forms a "prefix" of s_3.

NOT a prefix free code

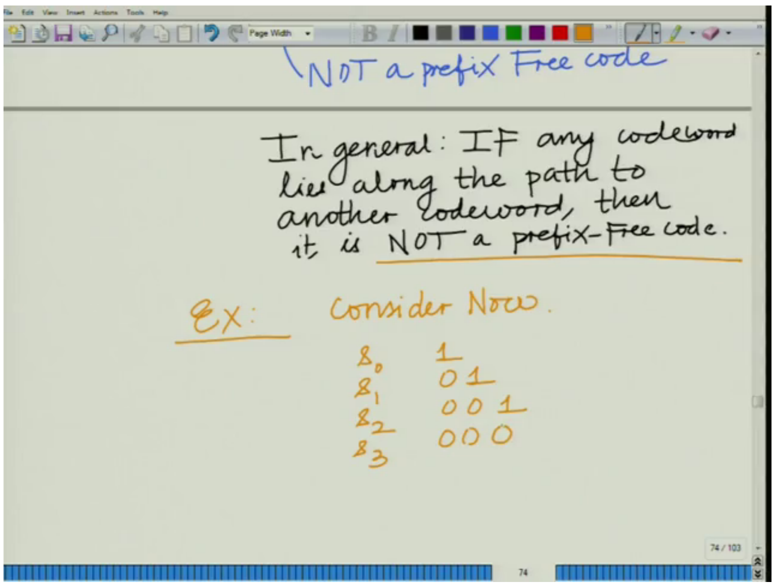
I can represented it as a binary tree and the binary tree representation would be as follows. So, we construct a binary tree for this code.

So, I start with the root, therefore 1 I go above for 0 I go below. So, for instance s 0 is 0 followed by another 0. So, this is s 0. Now s 1 would be well I am sorry, this is s 1 s 1 would be 0 followed by another 0. So, now, if I would like to do s 0 s 0 would be 1 followed by 0. So, for 1 you go to the top of the tree 1 you go the branch, take a branch going above and then followed by 0. So, this is s 0 and s 2 is 1 1. So, you go 1 and again take the branch above that is 1. So, this is s 1 and s 3 would be 1 1 0. So, 1 1 and you take a further 0 that would be your, that would be your s 3. So, s 3 would be 1 1 0. So, this s 1 So, s 1 s 0. So, we have represented the code using a binary tree that is what you done we simply represented the code using a binary tree ok.

So, this is a binary tree representation for the above code. Now if you can see in this code you can see that s 1 the code word s 1 lies allow or I am sorry this is the code word s 2 s 2 is 1 1 this is the code word s 2. So, you can see that s 2 lies along the path to s 3, s 2 lies along path 2.

S 2 lies along the path 2 that is symbol 2 lies along the path 2 symbol 3. And you can see s 2 you can see s 2 this is 1 1, s 3 is 1 1 0. So therefore, s 2 forms a prefix s 2 forms a prefix of s 3. Does we can see that s 2 in this binary tree representation of the code s 2 the node s 2 which represents which represents the code word, which is represented by the code word 1 1 that lies along the path 2 the code the code word for s 3 which is represented by 1 1 0 therefore, this is not a prefix free code ok.

(Refer Slide Time: 06:55)



NOT a prefix Free code

In general: IF any codeword lies along the path to another codeword, then it is NOT a prefix-Free code.

Ex: Consider Now.

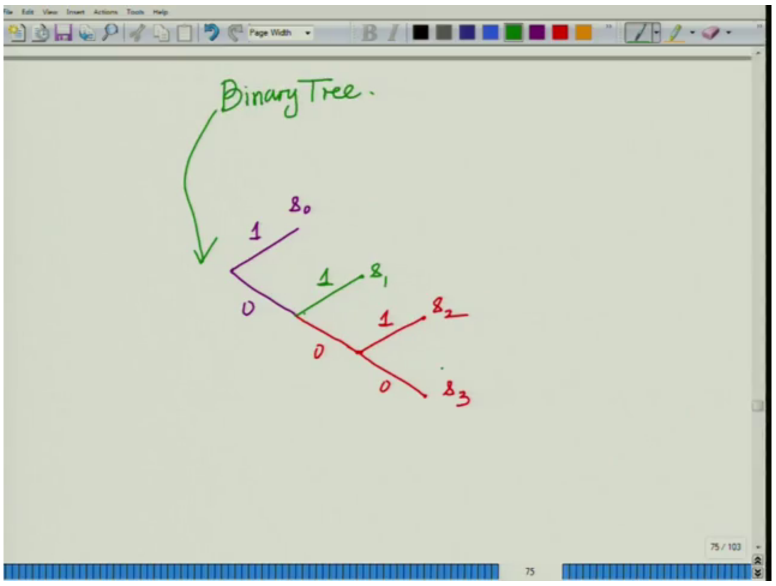
s_0	1
s_1	01
s_2	001
s_3	000

So therefore, this is not you can see that it is not it is not a prefix free code, because s 2 forms up s 2 lies alone.

So in general, if any code word right, if any code word lies along the path that is if any code word lies along the path, another code word in the binary tree representation then it is not a prefix, it is not a prefix free code; it is not a prefix free code. If any code word lies along. So, from the binary tree representation basically one can see it is a prefix free code or not. For instance consider another code ok.

Consider now s 0 represented by 1 s 1 represented by 0 1, s 2 represented by 0 0 1 and s 3 represented by 0 0 0.

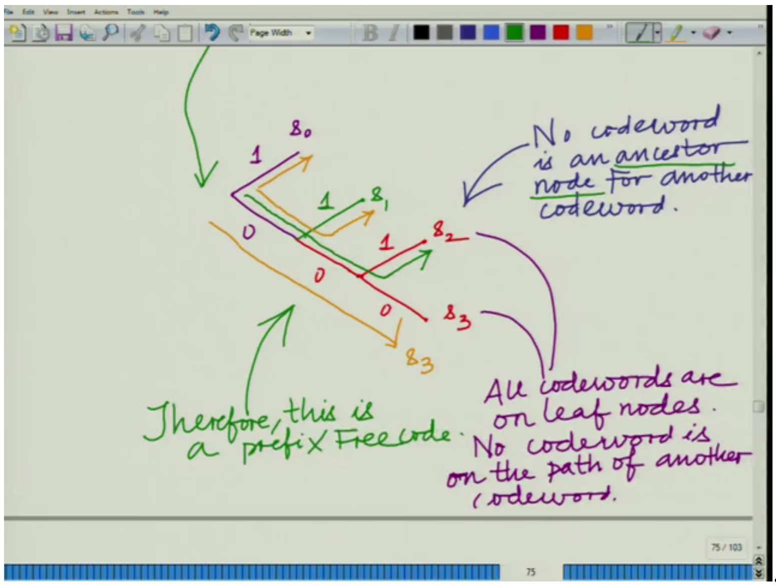
(Refer Slide Time: 08:44)



Now, if you look at this code you can see that well again let us look at the binary tree representation, go above for 1 below for 0 s 0 is 1. So, this is basically your s 0 s 1 is 0 comma 1.

S 1 is you go a below for 0 0 comma 1 this is s 1 and a for s 2 you have 0 0 1, for 0 0 goes below, s 2 is 0 0 1 and a s 3 is 0 0 0, s 3 0 0 0. And now if you can see basically you can say that no symbol lies or in the no code word for knows the code for any symbol. So, this is the binary tree code word for any symbol right.

(Refer Slide Time: 10:14)



No codeword is an ancestor node for another codeword.

All codewords are on leaf nodes. No codeword is on the path of another codeword.

Therefore, this is a prefix Free code.

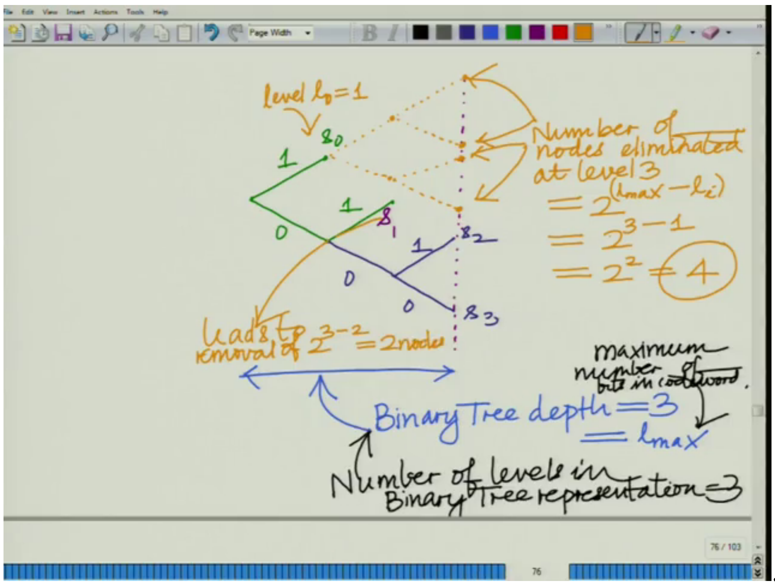
So, all codes end on leaf nodes, all code words end on leaf nodes. These are known as these are the various leaf nodes. All code words are on leaf nodes and basically, no code word is on the path of, an for instance if you would like to go to s 2 this is well we just draw it.

This is the path to s 2; this is the path to s 1, path to s 0 path to s 3. So, no code word is on the path of any other code word. At another way to says no code word is an ancestor of for code word right. So, our ancestor is basically where in node at that nodes split is into further nodes in the binary tree. So, this node is to is said to be the ancestor node of the of the corresponding branches and the leaves that emanate from those branches. So, no code word is the ancestor of another code for instance, here you can see in the previous 1 s 2 is the ancestor of s 3. Which is basically the same thing you are saying s 2 is a prefix of s 3, s 2 is an ancestor node of s 3.

However here you have no code is an ancestor no code word, all these are the same things, no code word is an ancestor node, no code word is an ancestor node for another code word. And therefore, no code word is an ancestor code for another code word therefore, this is a, Therefore this is a prefix free code.

So, in a binary tree for a prefix free code well, no code lies along no code word lies along path to another code word and no code word is basically ancestor node to another code word. All the code words are along with leaf nodes, which means the end nodes of any other branch they do not form intermediate nodes. So, that is a binary trees. So, from a binary tree representation one can see that if it is a one can check, quickly if it is a prefix free node or prefix free code or not. Now again if you look again at the same at the code let me draw the binary tree again.

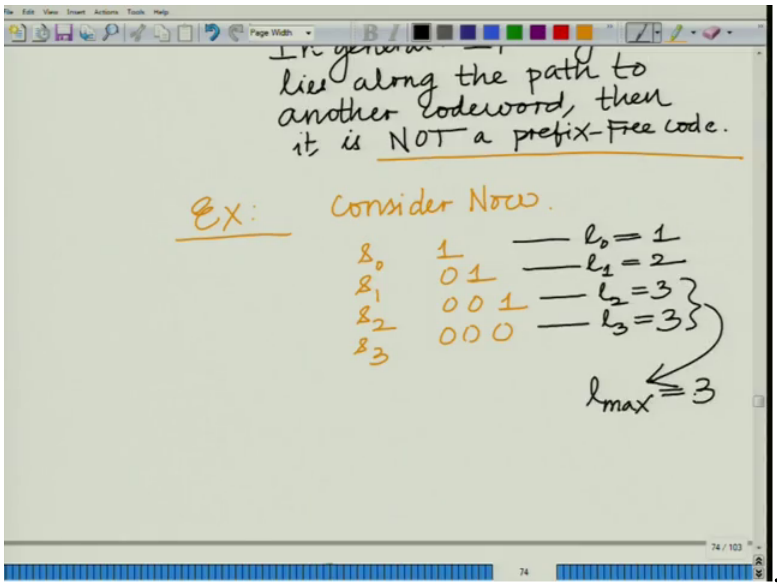
(Refer Slide Time: 13:55)


level $l_0 = 1$
s_0
1
0
1
s_1
s_2
1
0
0
s_3
Number of nodes eliminated at level 3
$= 2^{(l_{max} - l_i)}$
$= 2^{3-1}$
$= 2^2 = 4$
leads to removal of $2^{3-2} = 2$ nodes
Binary Tree depth = 3
$= l_{max}$
maximum number of bits in codeword.
Number of levels in Binary Tree representation = 3
76 / 103
76

This is 1 s 0, this is 0 this is 1, this is s 1 this is 0 1 0 s 2, s 3. Now if you look at the depth of this tree binary tree the depth equals l max equals 3. So, you have 0 0. So, if depth is the maximum number, that is the maximum number, you can think of that depth of the binary. So, the binary tree.

L max l max is the length. So, here you can see l max equal to 3 that is you have a code word with 3 bits that is maximum number of bits in a code word is 3. So, the maximum length of the code word is 3 and you can see that the binary depth of the binary tree is also 3 there is a numbers of levels in the binary tree used for representing the code number of levels. Number of levels and binary tree representation equal to 3, this is l max l max is equal to maximum number of bits in code word. So, you can see that if you look at the code words.

(Refer Slide Time: 16:03)


lies along the path to another codeword, then it is NOT a prefix-Free code.

Ex: Consider Now.

s_0	1	$l_0 = 1$
s_1	01	$l_1 = 2$
s_2	001	$l_2 = 3$
s_3	000	$l_3 = 3$

$l_{max} = 3$

Again If you go back you have l 0 equal to 1 this is the length one bit 1 1 equal to 2 and l 2 equal to 3, length of code word 2, length of code word 3 equal to 3.

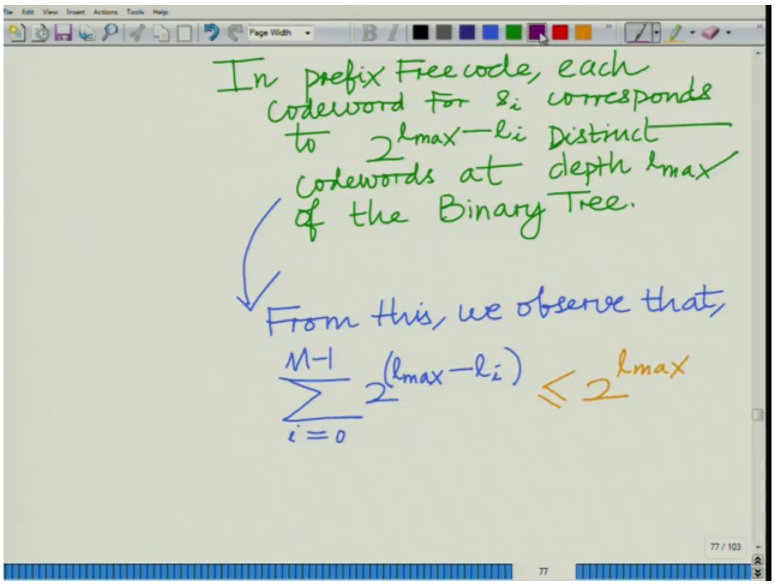
So therefore, from this you can see that l max is equal to 3. So, then so it is very simple that maximum 3 bits is are refuse to represent each code word. And that is the depth of the binary tree that is the number of levels in this binary tree. Now you will observe something interesting if you go to s naught, now let us look at, now it is not difficult to see that for a given binary tree. Now if you have the code word of s naught now because s naught terminates this leaf node, now the number of nodes that are eliminated because the leaf node terminates at n naught this is at leve1, 1 level l I am sorry, l 0 equal to 1 this at level l 0 equal to 1. Now therefore, the number of terminated nodes.

Now, what happens is because you are terminating this leve1 1 with s naught the corresponding number of nodes that are removed at level 3. So, the number of nodes you can see here in this case number of nodes eliminated at level 3 is equals 2 to the raise to l max minus l i that is basically, in this case that is 2 raise to 3 minus 1 l i that is i equal to 0. So, 2 raise to 3 I l i equal to i equal to 0. So, l naught equal to 1 2 raise to 3 minus 1 that is 2 raise to 2 equal to 4 and this is an interesting result. So, what you are observing is because we are terminating right, in this prefix free code because we are terminating the node the code had because no other code word can lie further down from s naught right. For that matter along any code word because then it would be a prefix that it would not be a prefix free code ok

So therefore, in order for it to be prefix free code you are terminating that particular branch at that code word s naught. Which means that further down at the depth of the binary tree that is the l max this leads to a removal from the binary tree of 2 to the power of l max minus l i. In this case 2 to the power of l max minus l naught for the ith code word it will lead to the removal of 2 to the power of l max minus l i in general. For instance, if you can look at s 1 s 1 is at the leve1 1, I cannot draw this binary tree completely, but if you look at level 2 if this is at level 2 correct s 1 is at level 2 because it as 2 bits 0 1. So, that will need to the removal of 2 to the power of l max that is 3 minus 1 1 that is 2 2 to the power of 3 minus 2 nodes at depth l max.

So, s 1 2 to the power of 3 minus 2 equal to 2 nodes. Now s 2 and s 3 are already at level 3 that is the depth. So, s 2 corresponds to one nodes s 3 corresponds to the one nodes that is 2 to the power of l max that is 3 minus 3 minus l i l I is also 3 bits 2 to the power of 3 minus 3 0 that is one node each node is corresponds to one node at level. So, what you see is each node in this binary tree corresponds to that if you extend it further corresponds to the depth of the binary tree that is l max corresponds to 2 to the power of l max minus l i distinct nodes. Now these nodes cannot overlap corresponding to the different code words because remember, it is a prefix free code right. So, no codes lies along the path to another code word therefore each code word.

(Refer Slide Time: 21:03)



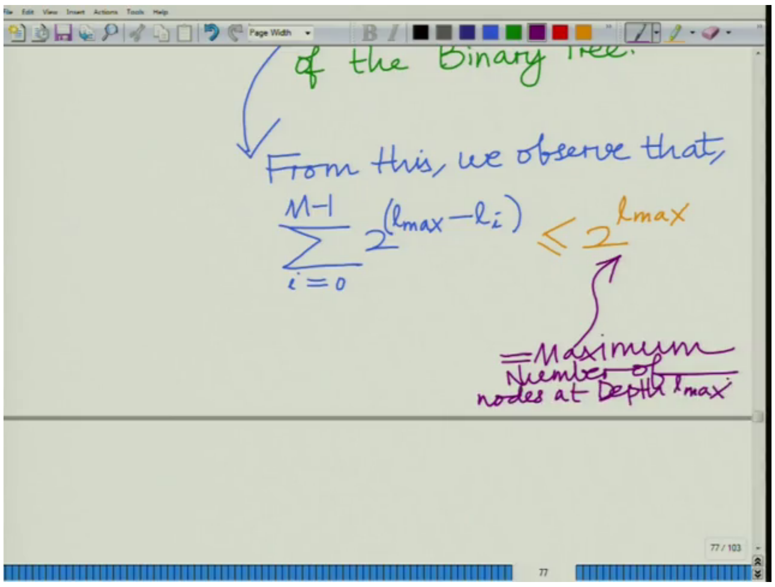
In prefix Free code, each codeword for s_i corresponds to $2^{l_{max}-l_i}$ Distinct codewords at depth l_{max} of the Binary Tree.

From this, we observe that,

$$\sum_{i=0}^{M-1} 2^{(l_{max}-l_i)} \leq 2^{l_{max}}$$

So, to summarize this in the prefix free code corresponds to 2 to the power of l max minus l i distinct, distinct code words at l max at depth l max or a depth of the binary tree at depth l max. Now from this we observe that summation, from this we observe that summation i equal to 0 m minus 1 2 to the power of l max minus l i must be less than or equal to that is 2 to the power of l max.

(Refer Slide Time: 23:05)



of the Binary Tree.

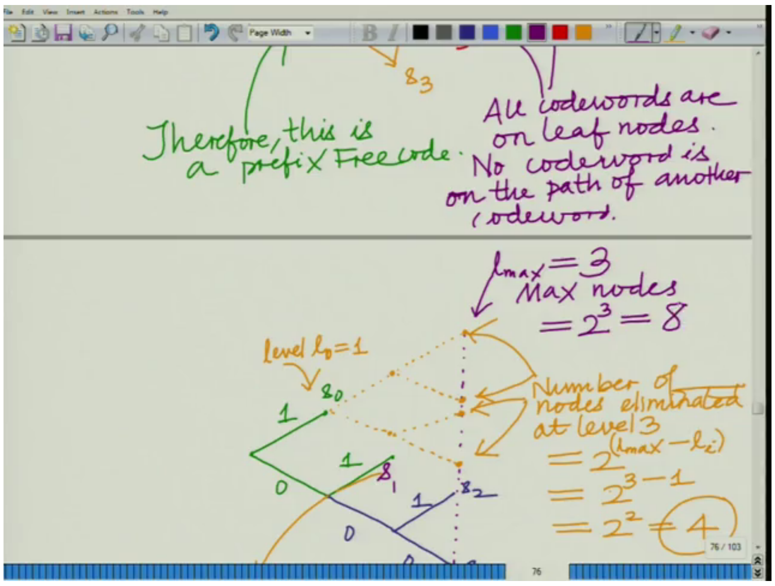
From this, we observe that,

$$\sum_{i=0}^{M-1} 2^{(l_{max}-l_i)} \leq 2^{l_{max}}$$

= Maximum Number of nodes at Depth l_{max}

Since 2 to the power of l max is the maximum number of nodes possible at depth l max. For instance if we look at l max equal to 3, correct?

(Refer Slide Time: 23:35)


s_3

Therefore, this is a prefix Free code.

All codewords are on leaf nodes. No codeword is on the path of another codeword.

$l_{max} = 3$
Max nodes $= 2^3 = 8$

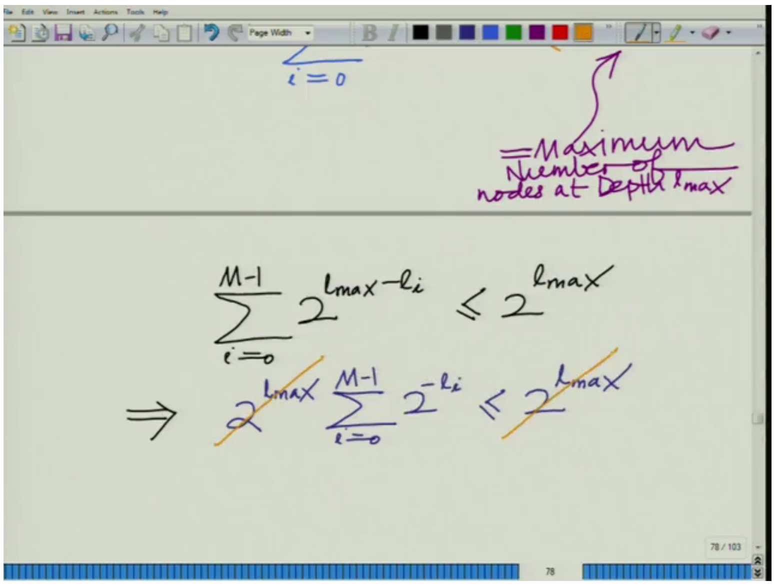
level $l_0 = 1$

Number of nodes ~~eliminated~~ at level 3
$= 2^{(l_{max} - l_i)}$
$= 2^{3-1}$
$= 2^2 = \boxed{4}$

L max equal to 3, the maximum number of nodes equal to 2 to the power of 3 equal to 8. Since what you observe is it is very simple add depth l max the maximum number of nodes that are possible in this binary tree is 2 to the power of l max. Now in each node it each code word You are removing or it corresponds to 2 to the power of l max minus l i distinct code words at the depth l max.

Therefore, the sum of all these nodes corresponding to e these different code words if you add the sum of all these nodes at the l depth max corresponding to all these code words, that must naturally be less than or equal to 2 to the power of l max, which is the maximum number of distinct nodes that is possible at the depth l max. And there from this we have a simple result.

(Refer Slide Time: 24:33)



$i = 0$

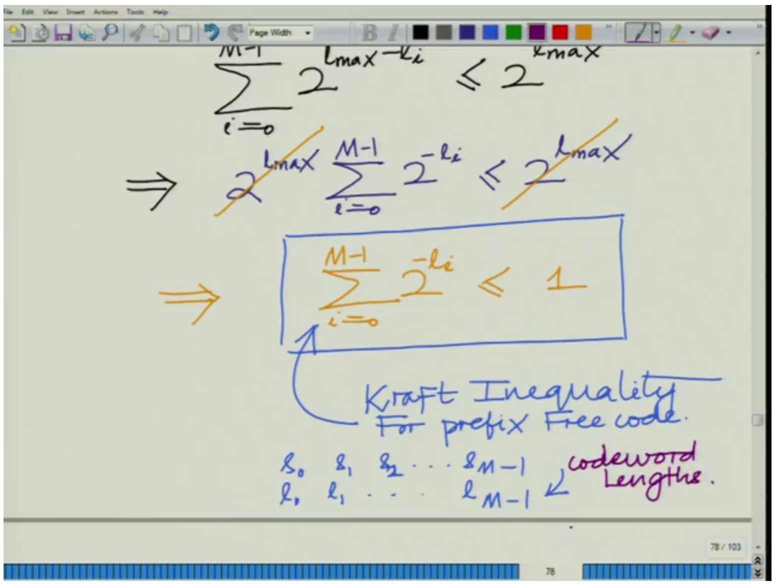
$=$ Maximum Number of nodes at Depth l_{max}

$$\sum_{i=0}^{M-1} 2^{l_{max} - l_i} \leq 2^{l_{max}}$$

$$\Rightarrow \cancel{2^{l_{max}}} \sum_{i=0}^{M-1} 2^{-l_i} \leq \cancel{2^{l_{max}}}$$

We have now rewriting this summation i equal to 0 to m minus 1 2 to the power of l max minus l i less than or equal to 2 to the power of l max. This implies that you take 2 to the power of l max common summation i equal to 0 to m minus 1 2 to the power of minus l i less than or equal to 2 to the power of l max.

(Refer Slide Time: 25:14)



$$\sum_{i=0}^{M-1} 2^{l_{max} - l_i} \leq 2^{l_{max}}$$

$$\Rightarrow \cancel{2^{l_{max}}} \sum_{i=0}^{M-1} 2^{-l_i} \leq \cancel{2^{l_{max}}}$$

$$\Rightarrow \boxed{\sum_{i=0}^{M-1} 2^{-l_i} \leq 1}$$

Kraft Inequality For prefix Free code.

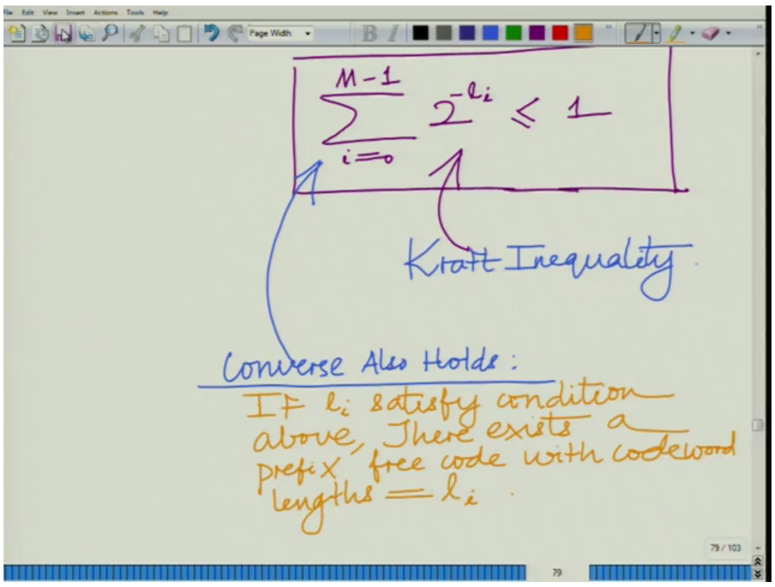
$s_0 \quad s_1 \quad s_2 \ldots s_{M-1}$

$l_0 \quad l_1 \quad \ldots \quad l_{M-1}$ codeword Lengths.

Now, cancel 2 to the power of l max now both sides and what you will get is a very elegant inequality which says summation i equal to 0 to m minus 1 2 to the power of minus l i is less than or equal to 1. And this is a fundamental inequality that as to be obeyed by the prefix free code which we have derived from the binary tree and the system as the Kraft inequality.

The system as the Kraft inequality for a, this is the Kraft inequality for prefix that is we are considering, s 0 s 1 s 2 s m minus 1 with lengths l 0 1 1 l 2 up to l m minus 1. These are the corresponding lengths; these are the corresponding code word lengths.

(Refer Slide Time: 26:42)



$$\boxed{\sum_{i=0}^{M-1} 2^{-l_i} \leq 1}$$

Kraft Inequality.

Converse Also Holds :

If l_i satisfy condition above, There exists a prefix free code with codeword lengths $= l_i$.

So, with these code word lengths the Kraft inequality tell me for a prefix free code we have to satisfy i equal to 0 to m minus 1 2 to the power of minus l i less than or equal to 1. This is your Kraft inequality, this is the Kraft inequality. This is a Kraft inequality and a now interestingly the converse can also be shown to shown to be true and it naturally follows, if you can look at the proof that if there are lengths l 0 1 1 l m minus 1 which satisfy this Kraft inequality, then there exists a prefix code for this alphabet with the given code word. Let us, let us node that also converse is also true which basically implies if lengths l i satisfy condition above.

There exist a prefix free code there exists a prefix free code with code word lengths equal to l i. So, that is what the Kraft inequality tell us. So, this module we have looked at a very important result that is Kraft inequality based on a binary tree representation for a prefix free code we said that the binary representation of the prefix free code is such that no code word is an ancestor code, word ancestor node for any other code word all right.

And using this property looking at them total number of distinct nodes that are possible at the depth maximum depth of the binary tree that is at the level l max corresponding to the maximum number of maximum length of the code word in this in any code. We have shown that this code word lengths l i must satisfy summation i equal to 0 to m minus 1 2 to the power of minus l i less than or equal to 1 the system as a Kraft inequality. And subsequently we will use the Kraft inequality to derive further insights about the nature of the code and principles regarding the code construction all right. So, we will stop here.

Thank you very much.