

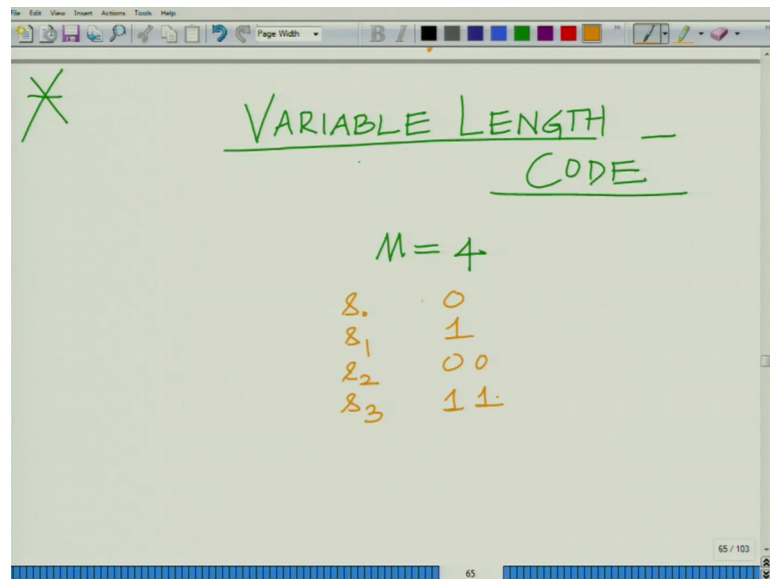
**Principles of Communication Systems - Part II**  
**Prof. Aditya K. Jagannatham**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Kanpur**

## Lecture - 42

## Uniquely Decodable Codes, Prefix-free Code, Instantaneous Code, Average Code length

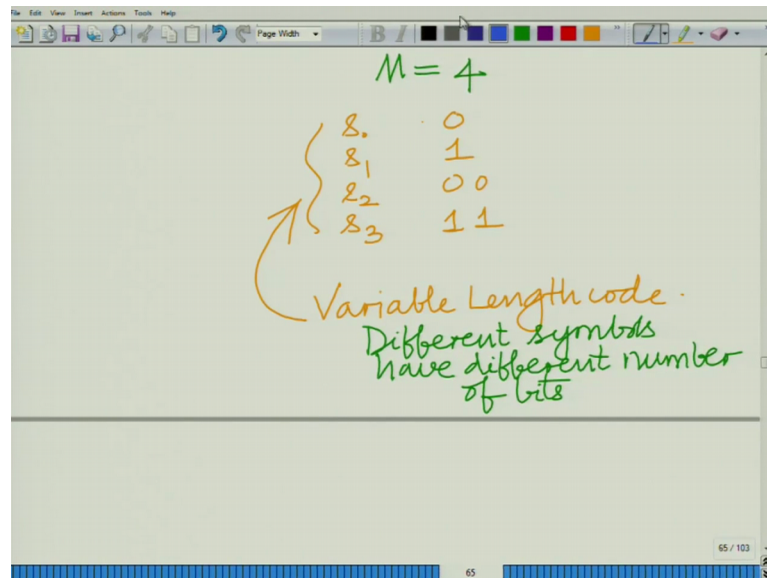
Hello. Welcome to another module in this massive open online course. So, we are looking at source coding that is representing the symbols emitted by the source as a sequence of binary symbols, right as a sequence of binary bits of information and our aim is to maximize the average length of code that is the average number of binary symbols or binary bits used to represent each symbol, all right. In this context, yesterday we have looked in the previous module at different kinds of course, we have looked at a fixed length code, we have also looked at a variable length code, ok.

(Refer Slide Time: 00:57)



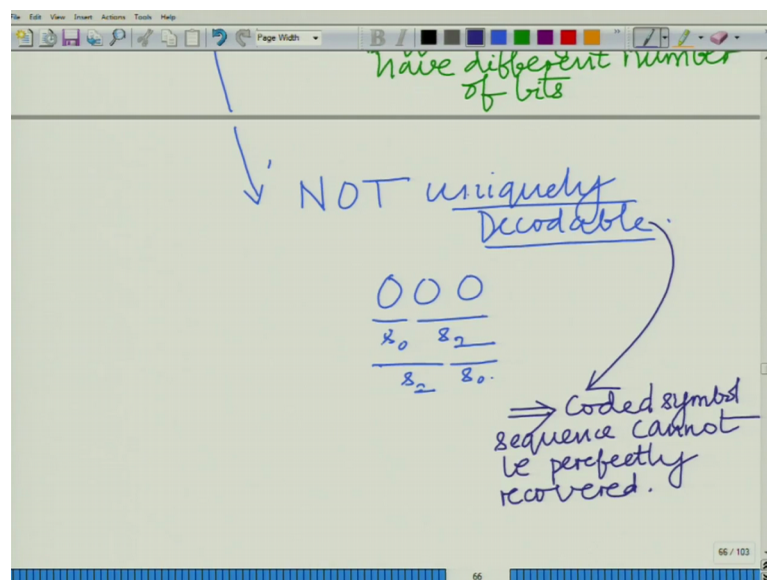
So, let us go about this. So, let us continue our discussion on this aspect of variable length code. For instance, with M equal to 4, we have seen an example of a variable length code s naught is 0, the alphabet s1 is represented by 1, s2 is represented by 0 0 and s3 is represented by 1 1.

(Refer Slide Time: 01:40)



We see this is a variable length code in that different symbols have different bits, different symbols have different number of bits. This is a variable length code.

(Refer Slide Time: 02:31)

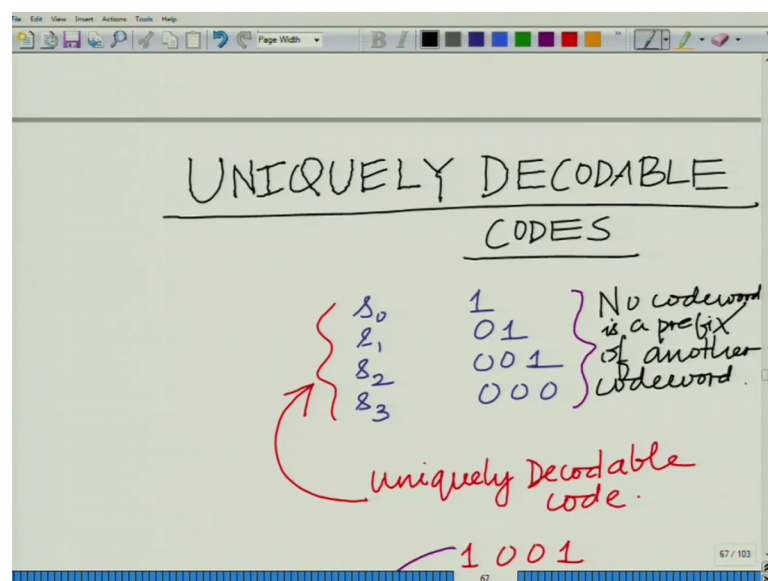


However, we have also seen that this is not uniquely decodable, it is not uniquely decodable, since you have the sequence 0 0 0. This can either correspond to  $s_0, s_2$  or it can correspond to  $s_2 s_0$  and so on. So, it is not uniquely decodable which means that the original that is the coded symbol sequence, this implies the coded symbol sequence cannot be perfectly reconstructed or cannot be perfectly recovered and obviously, that is

bad. That is once we code the symbols, obviously we have represented them using a sequence of binary bits. You would like to or you store them using sequence of binary information bits. One would like to obviously at a later time or at the receiver if that transferred across the channel, recover the original symbols by looking at the sequence of binary bits.

Now, if the sequence of symbols cannot be recovered from this binary bits, then there are going to be problems because the original message is lost, alright. So, it is a lossy coding and the original message cannot be recovered. So, obviously we are interested, naturally we are interested in designing uniquely decodable codes that is codes from which there is one to one mapping, that is one to one mapping from that is for any that is there is a mapping that is basically for any set off from a corresponding to a new set of sequence of binary information bits. The corresponding symbols can be uniquely recovered that is there are low two sequence of symbols which are mapped to the same sequence of information bits, ok.

(Refer Slide Time: 04:55)



So, we would like to design uniquely decodable codes and for instance, if we can take an example, again you can see that  $s_0 s_1 s_0$  is 1,  $s_1$  is 0 1,  $s_2$  is 0 0 1,  $s_3$  is 0 0 0. You can see this is a uniquely decodable code for example if we receive the symbol.

(Refer Slide Time: 06:00)

CODES

$s_0$	1
$s_1$	01
$s_2$	001
$s_3$	000

uniquely Decodable code.

No codeword is a prefix of another codeword.

1 0 0 1  
s<sub>0</sub> s<sub>2</sub>

No other symbol sequence corresponds to this bit sequence.

If we received the sequence 1 0 1 0 0 1, I can immediately decode this as  $s_0$  followed by  $s_2$ , ok. So, there is no other symbol sequence corresponds to this bit sequence. So, this is a uniquely decodable code.

Now, in part the reason for this you can see is because if you look at the set of codewords, all right you have four codewords corresponding to the four symbols. You can see that no codeword is the prefix of another codeword. You can see from this that no codeword is a prefix of no codeword is a prefix of another codeword, so no codeword.

(Refer Slide Time: 07:36)

VARIABLE LENGTH CODE

$M = 4$

$s_0$	0
$s_1$	1
$s_2$	00
$s_3$	11

Variable Length code.  
Different symbols have different number of bits

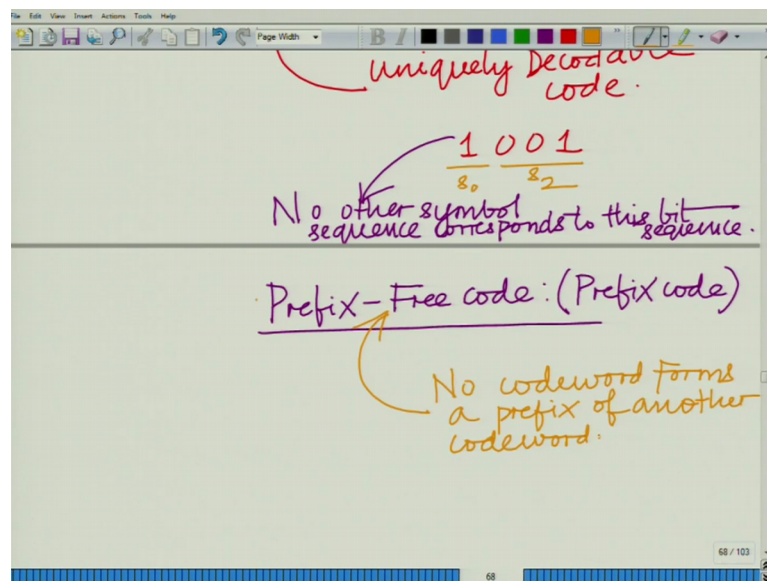
codeword for  $s_0$  is a prefix of codeword for  $s_2$



However, if you look at this, you can see that  $s_0$ , that is 0 is a prefix of 0 0. So,  $s_0$  or codeword for  $s_0$  is a prefix of the code for  $s_2$ , right. Here the codeword is a prefix, the codeword of  $s_0$  is the prefix of the codeword or  $s_2$ , ok.

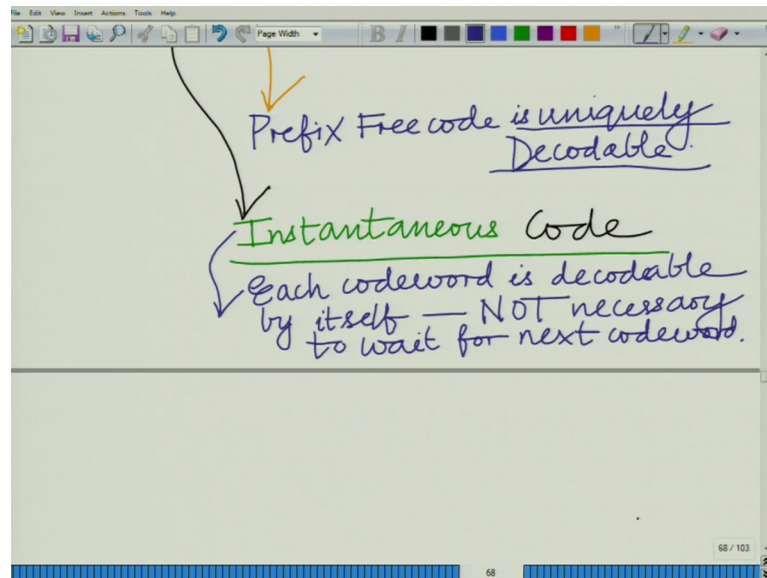
So, here you can see no codeword in this uniquely decodable code.  $s_0$  is 1,  $s_1$  is 0 1,  $s_2$  is 0 0 1,  $s_3$  is 0 0 0.

(Refer Slide Time: 08:43)



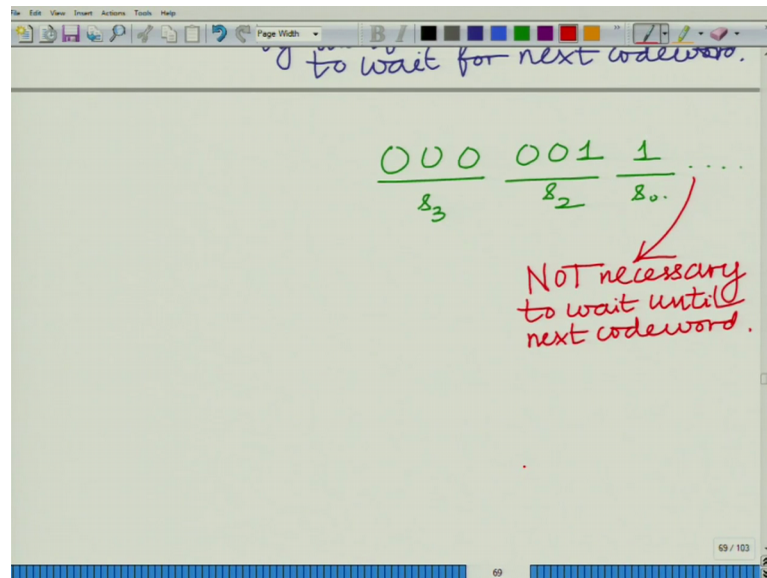
No codeword is the prefix of another codeword, ok and such a code, this is known simply as a prefix free code or also known as a prefix code. Sometimes it is explicitly called as a prefix free code or simply prefix code. Sometimes it is also known as a prefix code is basically that no codeword forms prefix of another codeword. Prefix means something that is placed before something that is in front of another word. Prefix of a word is something that is in front of another word, right, no codeword. If code is known as a prefix free code, if the mapping is such that no codeword forms a prefix of another codeword, ok.

(Refer Slide Time: 10:15)



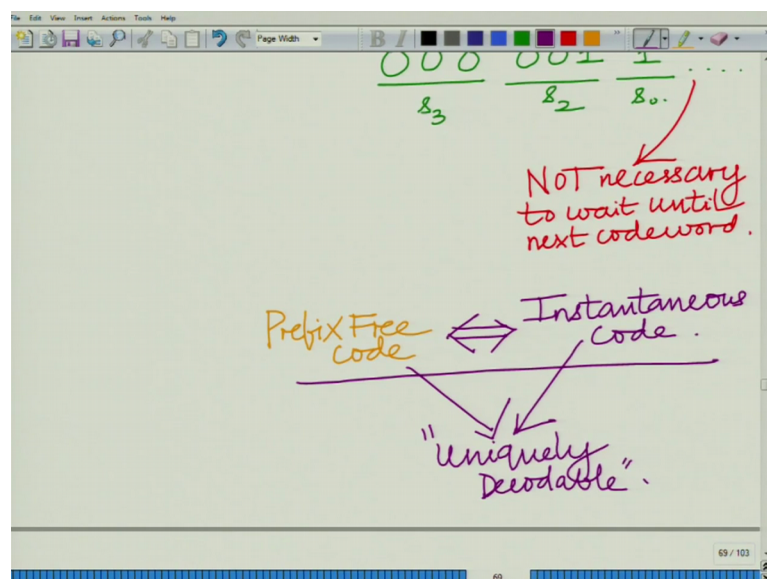
Now, such a prefix free code is uniquely decodable. So, prefix free code, the importance of a prefix free code is that prefix free code is uniquely decodable, more over prefix free code is also known as an instantaneous code. It implies that you do not need to wait for the next codeword to decode the current. So, that is each codeword is decodable by itself, that is it is not necessary to wait for the next codewords. So, what happens in instantaneous code is for instance, if you look at each codeword correct, the moment you look at a codeword, it can be a map to a symbol. You do not need to wait to the next codeword to begin. For instance, if you take a look at this, let us again take a look at this. Let say I have 0 0 0 followed by 0 0 1, ok.

(Refer Slide Time: 12:28)



So, I have this code  $000$  followed by  $001$  followed by let us say  $1$ . Now, the movement you will look at  $000$ , I can map it back to  $s_3$ , correct. The moment you see  $001$ , I can map it to  $s_2$ . The moment you see  $s_1$ , I can map it to, I do not need to find. It is not necessary to wait until the next codeword. This is known as, such a code is known as an instantaneous code and the prefix free code and an instantaneous code can be shown to be one and the same. If it is a prefix free code, a code is a prefix free code if it is an instantaneous, ok.

(Refer Slide Time: 13:48)

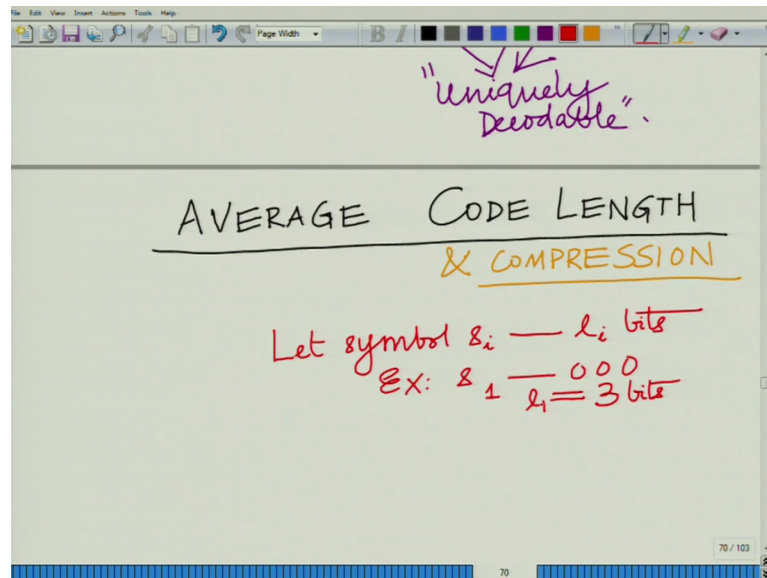


So, prefix free code and an instantaneous code and therefore, they are uniquely decodable prefix free code and instantaneous code and both of these are uniquely or both of these are uniquely decodable, that is given a set of coded bits, I can map it to a unique set of information. I can make it to a unique set of symbols, ok.

So, prefix free codes and instantaneous codes and you interested in designing such code that is basically instantaneous codes or prefix free codes. So, we will focus and of course, one important point to note here is that uniquely decodable does not imply, that is in other words, they can be other codes, other than prefix free codes which can also be uniquely decodable, but we not be looking at them, alright. Then, when the most important class of uniquely decodable codes or one of the most convenient lectures, frequently used convenient class of codes which are uniquely decodable, I am basically prefix free or instantaneous code and we will be interested in the design of such codes. That is how to design the prefix free or uniquely decodable code for a source with a given set of symbols with a set of probability, that is we are looking at a discrete.

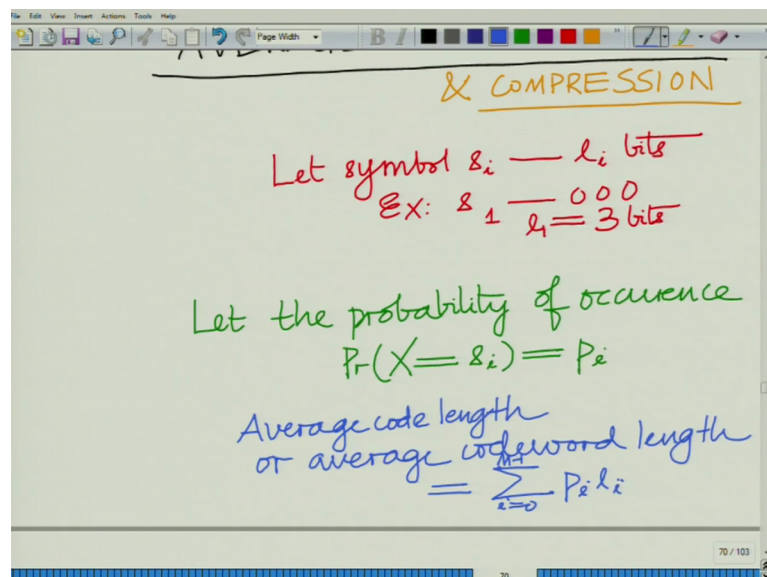
As we have clarified in the previous model, we are interested in looking at discrete memory less sources with a set of symbols with a fix set of probabilities and the probability of each symbol does not depend on the probabilities of the symbols generated in the past. There is probabilities of symbol. At time instant  $k$  do not depend on the probabilities on the symbols generated in the past.

(Refer Slide Time: 16:09)



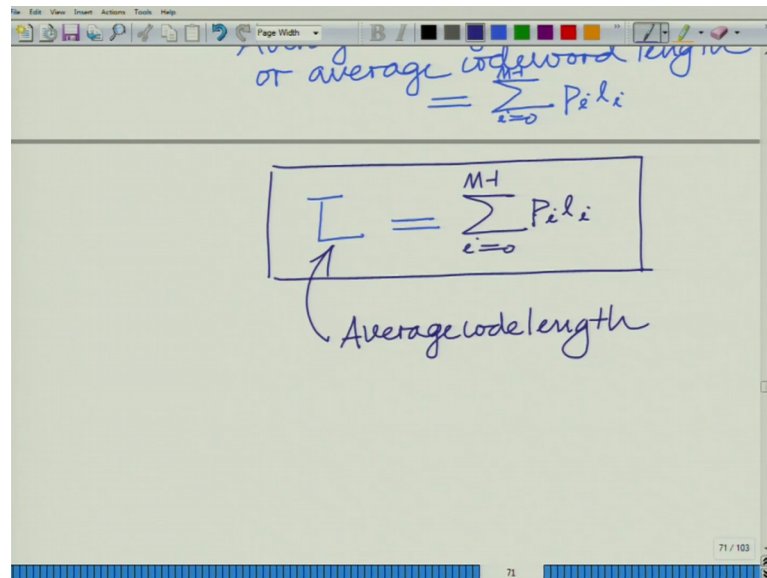
Let us now look at the average code length that is this is an important metric because this is basically going to design the efficiency of code design. So, what is the average code length? Now, the average code length and its relevance to compression for instance, let symbol  $s_i$  be represented using  $l_i$  bits.  $s_1$  is represented in 0 0 0 equal to 3 bits, that is length equals to 1, the length of  $s_1$  equals 3, ok.

(Refer Slide Time: 17:25)



Now, let the probability of occurrence that is probability  $X$  equal to  $s_i$ . This be equal to  $P_i$ . Now, the average code length or average codeword length is defined as summation over all codewords  $i$  equal to 0 to  $M$  minus 1  $P_i l_i$ .

(Refer Slide Time: 18:29)



The image shows a handwritten slide with a toolbar at the top. The text on the slide is as follows:

average code length or average codeword length  

$$= \sum_{i=0}^{M-1} P_i l_i$$

---


$$\bar{L} = \sum_{i=0}^{M-1} P_i l_i$$

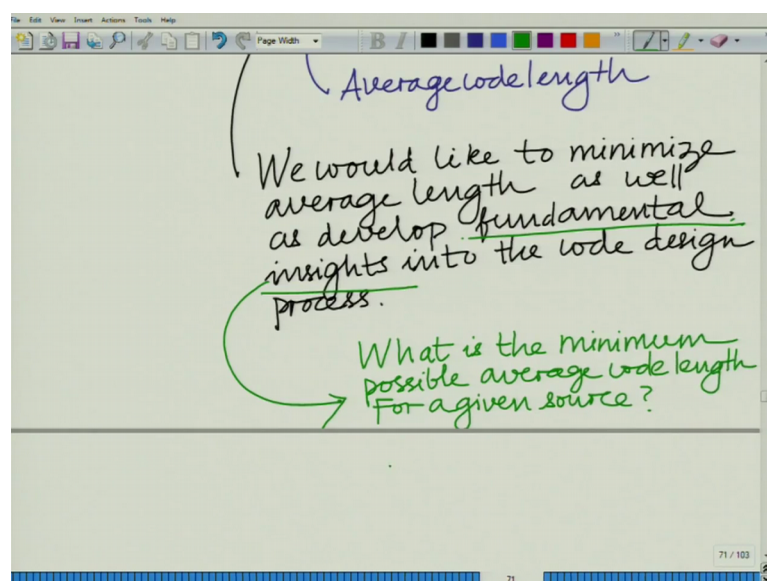
A bracket under the  $\bar{L}$  is labeled "Average code length".

So, average code length  $\bar{L}$ , let us denote that by  $\bar{L}$ . The average code length is summation  $i$  equal to 0 to  $M$  minus 1. That is the average that is you have the length  $l_i$  weighted by the probability  $P_i$ . This is the average length of the code and what we would like to do now is basically we would like to develop efficient codes which minimize this average code length that is the average number of bits that is required to represent any symbol generated by the source.

So, an efficient code is one which has obviously the least average code length because that requires the lowest number of bits on average to represent the symbol of the source and naturally, it is therefore makes it easier to communicate transmit information over the channel. It makes it easier to store the symbol generated by the source by coding them appropriately because the space that is going to be occupied by the bits, the coded bits is going to be minimum for this efficient code.



(Refer Slide Time: 19:54)



So, what we are interested in is, we would like to minimize average length as well as develop fundamental insights into the code design process, that is for instance, we would like to ask the question for a certain source, we generate certain set of symbols with certain probability, what is the lowest average code length that is possible. That is how low can I get the average code length to be? That is fundamental and therefore, one using the answer to this fundamental question, one can target an appropriate code length, alright. So, we will not only be looking a designing efficient codes, but we will also be interested characterizing how efficient can that process be, how low can the average length code be for a source with a given set of symbols with a set of probabilities. So, we would also like to ask the question, we would also like to generate some fundamental insights into this code design process, ok.

So, in the sense, how low or basically what is the minimum let us put it this way. What is the minimum possible average code length for a given source? So, this is basically what we would like to know, alright.

So, basically in this module, we have completed our discussion regarding the codes. What we have seen is, we have seen uniquely decodable codes and an important class of codes, prefix free codes which are basically no codeword is a prefix of another codeword or which are also known as the instantaneous codes because one need not wait till the next codeword to decode the current codeword, alright and what we have said is, we are

interested in designing such prefix free or instantaneous codes. And also, we are interested in answering some fundamental questions as how low, alright, what is the fundamental bound on the average code length for any code that can be designed for a particular discrete memory less source, alright. So, that will help us design it. So, we would like to come up with efficient schemes to design codes and we would also like to characterize their efficiency. There is how far are we from the lowest possible average code length that can achieve, that can be achieved for a particular source with certain set of symbols given a certain set of probabilities for the symbols, alright.

So, with this we will stop this module here, and look at other aspects in the sub segment modules.

Thank you very much.