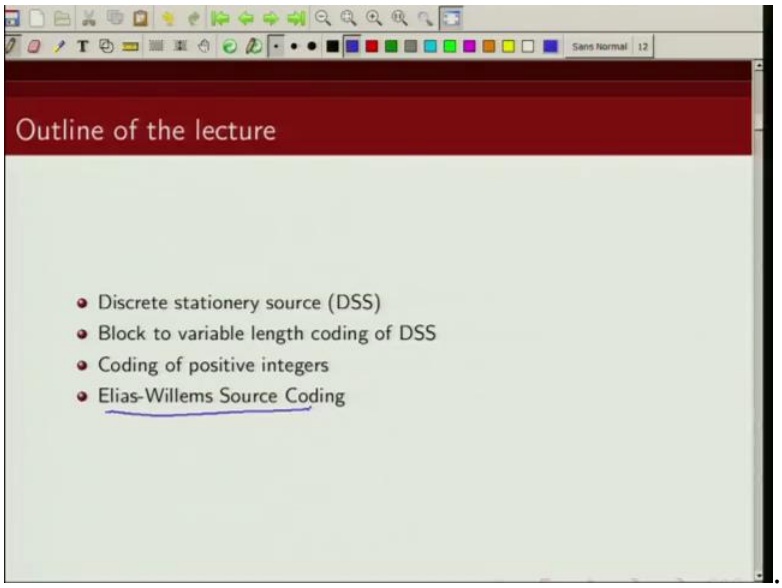
An Introduction to Information Theory
Prof. Adrish Banerjee
Department of Electronics and Communication Engineering
Indian Institute of Technology, Kanpur

Lecture – 08
Coding of Sources with Memory

Welcome to the course on an introduction to information theory. I am Adrish Banerjee. Today, we are going to talk about how to do source compression for sources with memory. So, so far we have discussed source compression algorithm of sources, which are discrete memory less sources.

So sources, but in general the sources that we see in nature have memory. So, can we exploit this memory, while doing source compression? That is what we are going to talk about today. So, before we discuss about that we will first define what we mean by a discrete stationary source with memory.

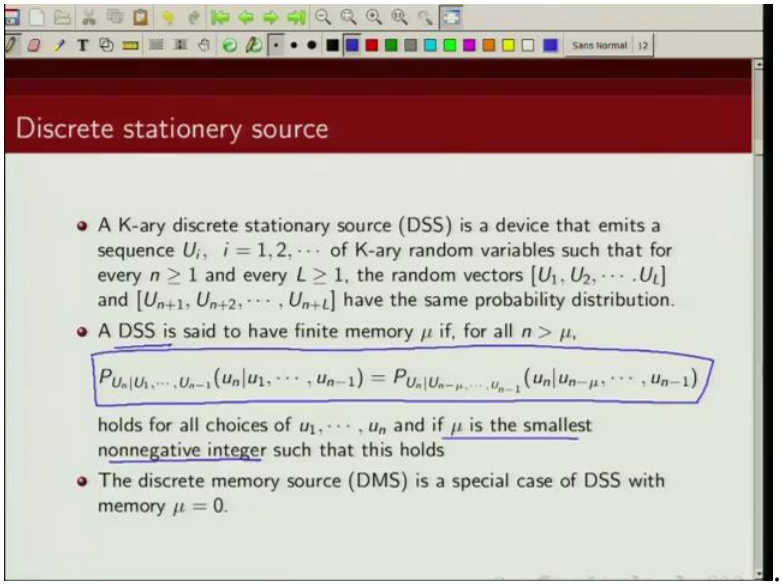
(Refer Slide Time: 01:02)



Once, we describe this discrete stationary source of memory, we are going to describe how we can do. We will take a simple example of block to variable length coding. So, how we can do block to variable length coding for a discrete stationary source? And, we will prove some properties of discrete stationary source. We will talk about a particular coding scheme, which is known as Elias-Willems source coding algorithm. And, it uses a

technique to code positive numbers. So, we will talk about how we can code positive numbers using a prefix free code. And, we are going to use that to design Elias-Willems source coding scheme, which is a block to variable length source coding scheme for a discrete stationary source.

(Refer Slide Time: 02:00)

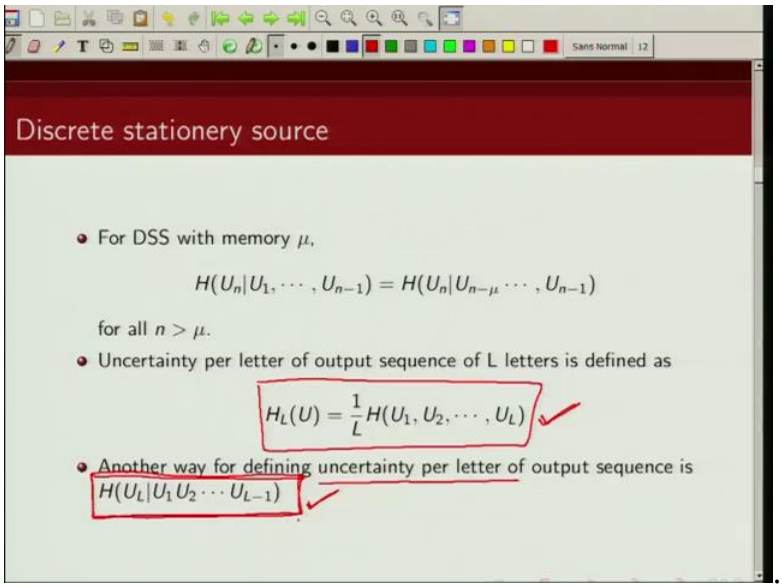


So, what is a discrete stationary source? So, as we can see it is discrete and stationary. So, a K-ary discrete stationary source emits a sequence U i, which consists of K-ary random variables, such that for every n greater than equal to one and every L greater than equal to one, the random vector denoted by U 1, U 2, U 3, upto U L and random vector U n plus one, U n plus two upto U n plus L; they have the same distribution. So, it does not matter what n is; n can be 2, 3, 5, and 100. If this random vector and this random vector have same probability distribution, then it has a stationary. Basically, it is a stationary source.

Now, we say a discrete stationary source to have a finite memory mu. If for all n greater than equal to mu, we can write probability of U n; given U 1, U 2, U n minus one, if it only depends on past mu values. So, probability of U n given, U n minus one, U n minus two, U 2, U n is, if it is same as probability of U n given, U n minus one, U n minus two upto U n minus mu, for all choices of U 1 to U n. Then, basically we say that this particular source has a finite memory mu. And, mu is the smallest non-negative integer for which this relation holds. Now, as you can see the discrete memory less source that

we have considered so far is a special case of discrete stationary source, which belongs to the case when mu is zero.

(Refer Slide Time: 04:27)



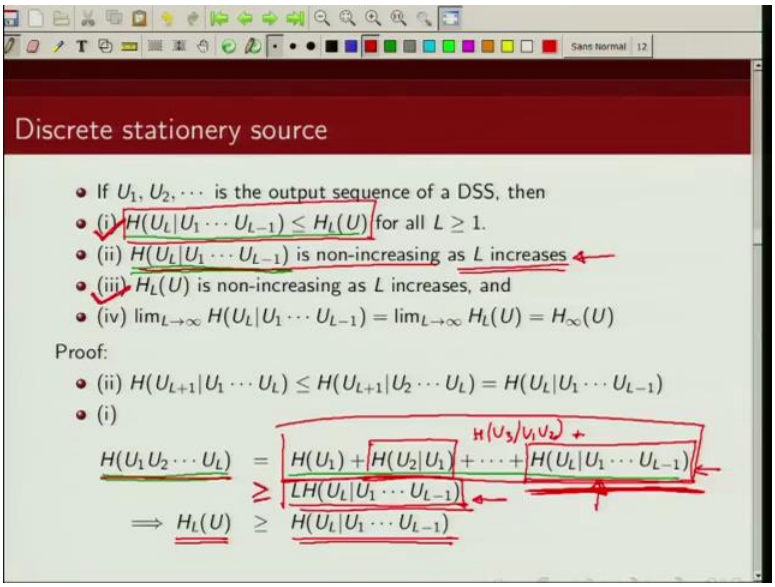
Now, it is not very difficult to prove that for a discrete stationary source with memory mu, the uncertainty in U n, given U 1, U 2, U n minus one is equal to uncertainty in U n, entropy U n, given U n minus one, U n minus two, U n minus mu. This follows directly from the property that is stated earlier that if you have a discrete stationary source of memory mu, it is going to satisfy this condition; that probability of U n, given U 1, U 2, U n minus one is same as probability of U n, given U n minus one, U n minus two, U n minus mu. So from this, and applying the definition of entropy, we can directly get this relation; that for our discrete stationary source of memory uncertainty in U n, given U 1, U 2, U n minus one is same as uncertainty in U n, given U n, U n minus one, U n minus two upto U n minus mu, where mu is the memory of this discrete stationary source. And, this is true for all n greater than mu.

Now, how do we define uncertainty for a discrete stationary source? So, there are two ways, we can describe it. One way, we can describe this uncertainty per letter. So, this is defined as follows. So, uncertainty per letter is defined as the uncertainty in U 1, U 2, U 3, U L. So, joint entropy of these U 1, U 2, U L divided by L. So, that is one way of defining uncertainty per letter for a discrete stationary source. Another way of defining this uncertainty per letter is using this conditional entropy. So uncertainty in U L, given

U 1, U 2, U 3, U L minus 1. So, there are two ways in which we could describe the uncertainty per letter for a discrete stationary source. One is this and the second method is, second way is this.

Now, we are going to prove some properties of these measures of uncertainty per letter; these two. And, we are going to show that when L is very large, when L goes to infinity, these two are same.

(Refer Slide Time: 07:24)



So, next we are going to prove some properties of the sequence, which comes out of a discrete stationary source. If U 1, U 2, U n is the output of a discrete stationary source, then following relation holds. Uncertainty in U L, given U 1, U 2, U L minus one is less than equal to this quantity; uncertainty per letter, which is defined in this particular way.

The next property that we are going to prove is this conditional entropy H of U L, given U 1, U 2, U L minus one. This is a non-increasing function of L. So, in other words uncertainty in let say U L plus one, given U 1, U 2, U L; this will be less than or equal to uncertainty in U L, given U 1, U 2, U L minus one. So, it is a non-increasing function as of L.

Next, we are going to show that this uncertainty per letter; this is also a non-increasing function of L. And finally, we are going to show that these two different ways of representing this uncertainty per letter. When L goes to infinity, both are same. And, this

we call as entropy rate of a discrete stationary source.

So, let us prove these properties one by one. So, we will first start with this second property. So, let us write this uncertainty in U L plus one, given U 1, U 2, U L. Now, to show that it is a non-increasing function of L as L increases, so what we need to show is this is less than equal to uncertainty in U L, given U 1, U 2, U L minus one. So, how do we prove it? So, first we write down this; uncertainty in U L plus one, given U 1, U 2, U 3, U L. Now, this is less than equal to uncertainty in U L plus one, given U 2, U 3, U 4, U L. So, we have removed the uncertainty on U 1.

Now, why is this less than equal to this quantity? We know the property of conditional entropy. The conditional entropy cannot increase the entropy. So, conditional entropy of this given U 2, U 3, U L, and this is conditional entropy of U L plus one, given U n and then U 2, U 3, U L. So, this is condition on an additional letter, which is U 1. So, this is condition on an additional letter U 1. So, this entropy cannot be more than the entropy of this. And that is why we wrote that uncertainty in U L plus 1, given U 1, U 2, U 3, U L is less than equal to uncertainty in U L plus 1, given U 2, U 3, U L.

Now, this quantity can be written because U i is the output of a discrete stationary source. So, uncertainty in U L plus 1, given U 2, U 3, U L will be same as uncertainty in U L; given U 1, U 2, U L minus 1.

So, I just, if you recall the definition of a discrete stationary source, what did we say? We said two sources. Basically, a source is a discrete stationary source is for every n is greater than equal to 1 and for every K greater than equal to one, the distribution of this random vector U 1, U 2, U 3, U L is same as the distribution of this vector U n plus one, U n plus 2, up to U n plus L.

So, invoking this property of discrete stationary source, we can then write; because U i's are coming out from a discrete stationary source, you can write this uncertainty of U L plus one, given U 2, U 3, U L as uncertainty in U L, given U 1, U 2, U L minus one. Specifically, we are replacing L by L minus 1. If we do that we get this. As since it is a stationary source, so this distribution and this distribution will be same, uncertainty of this will be equal to uncertainty in this. So, hence we have proved that the uncertainty in U L plus one, given U 1, U 2, U L is less than equal to uncertainty in U L; given U 1, U 2, U L minus one. So, we have proved that as L increases, this term is non-increasing

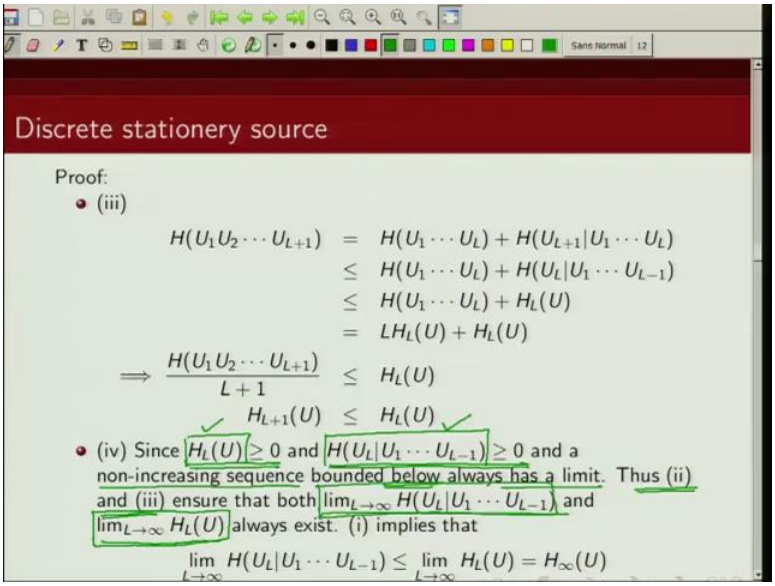
because this is less than equal to this. Now, we are going to make use of this property to prove other properties as well. And that is why I proved this property; number two, first.

Now, let us try to prove now property number one. So, we can write the joint entropy between U 1, U 2, U 3, U L. Using chain rule, we can write it like this. So, this is uncertainty in U 1 plus uncertainty in U 2 given U 1 plus uncertainty in U 3 given U 1, U 2 and up to uncertainty in U L given U 1, U 2, U 3, U L minus 1.

Now, from property number two we know that L; as L increases, this term is non-increasing. So, in other words among all these terms, this term H of U 3 given U 1, U 2 and like that this term, the smallest term is going to be this. Why because as L increases, this is a non-increasing. So among all these terms, this term will be smallest. And, there is L such terms. So, what we can do is this summation can be lower bounded by L times of this particular term. And that is what we are writing here; that this joint entropy is greater than equal to L times uncertainty in U L, given U 1, U 2, U L minus one. So, for this we obtained from chain rule; this we obtain by applying this property two.

Now, we divide both sides by L. So, what we get on the left hand side is uncertainty per letter. And, what we get on the right inside is uncertainty in U L, given U 1, U 2, U 3, U L minus one. So, then we have proved about property one, which says uncertainty in U L, given U 1, U 2, U L minus one is less than equal to this uncertainty per letter; for all L greater than equal to 1.

(Refer Slide Time: 16:51)



Next, we are going to prove this property three. So, to prove this property three, so let us first write the joint entropy of U 1, U 2, U 3, U L plus one. So, again using chain rule I can write this as uncertainty in U 1, U 2, U 3, U L plus uncertainty in U L plus one, given U 1, U 2, U 3, U L. Now because this particular term is non-increasing as L increases, so this term will be less than equal to this term. So, this follows from property two, which we have proved that uncertainty in U L, given U 1, U 2, U L minus one is non-increasing as L increases. So, since this term here is less than this particular term, what I get here is this joint entropy is then upper bounded by joint entropy of U 1, U 2, U L plus uncertainty in U L, given U 1, U 2, U 3, U L minus one. Now, I know from property one that this particular term is less than equal to uncertainty per letter, which is H L of U.

So, then I again can upper bound this by uncertainty in U 1, U 2, U L plus uncertainty per letter. Now, what is this term? This term can be written as L times uncertainty per letter. So, this is equal to this plus uncertainty per letter. So, this is equal to L plus one H L of U.

So, what we have shown so far is this joint entropy is less than equal to L plus one H of L U. Now, you divided both side by L plus one. What we get is H of L plus one U is less than equal to this quantity. So, as L increases we can see this uncertainty per letter. This is a non-increasing function. And that was our property three; H L of U is a non-increasing function as L increases.

Now, we are going to show the final property, which says when L is very large, and then L goes to infinity. Whether we describe this uncertainty per letter as H of U L, given U 1, U 2, U L or whether we define this as uncertainty per letter in this particular fashion. As L goes to infinity, they are both same. And, this we denote by H infinity of U, which is we call it entropy rate of a discrete stationary source.

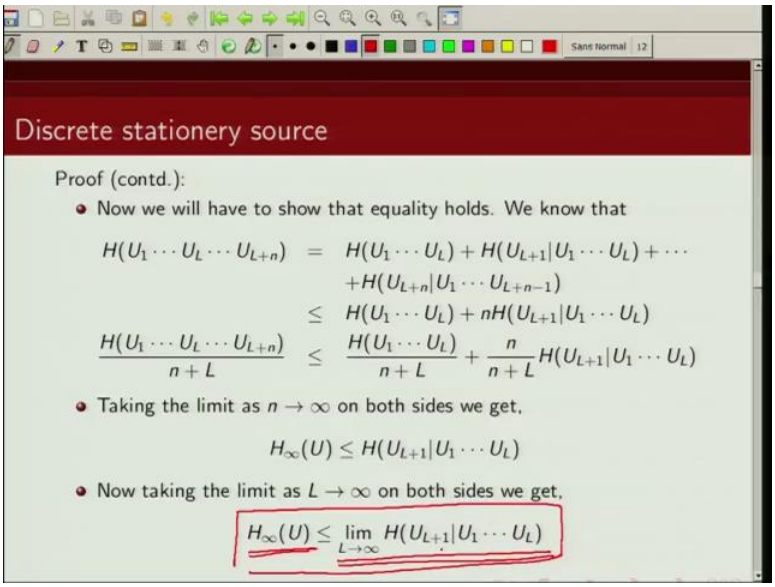
So, let us prove this. Now, we know that this is greater than equal to zero and this term is also greater than equal to zero. So, we have a non-increasing sequence. Remember, these sequences are non-increasing. So, we have non-increasing sequence, which is bounded below; because these non-increasing sequences are greater than equal to zero. So, they are bounded below. So, if you have non-increasing sequence which is bounded below, then as L goes to infinity, basically they always have a limit. So, thus from the property

two and three, which basically says that this particular term and this particular term is non-increasing as L increases. So, this ensures that when L goes to infinity, both these terms have limits which exist.

Now, as we have said we have to prove that as L goes to infinity, this particular term is same as this. So, how do we prove it? We will first show that one of these terms is less than equal to the other term. And then, we will show the other term is less than equal to the first term. So, for both of these conditions to exist, then they must be equal.

So, if we go back and look at property number one, what is property number one which says uncertainty is U L? Given, U 1, U 2, U L minus 1 is less than equal to H of L U uncertainty per letter. Now, if you let L go to infinity, then we can write this statement that uncertainty in U L, given U 1, U 2, U L minus one, when L goes to infinity, this is less than equal to limit L tends to infinity H of L U. And that we are denoting by H infinity U, which we are calling as entropy rate. So, this follows from property number one and we let L go to infinity. Now, to prove that they are equal, and now need to show that this particular term is less than equal to this particular term, so to prove this, I will do the following.

(Refer Slide Time: 23:40)



I will first find out the joint entropy of U 1, U 2, U 3, U L upto U L plus n. And using chain rule, I can write this as uncertainty in U 1, U 2, U 3, U L plus uncertainty in U L plus one, given U 1, U 2, U L upto uncertainty in U L plus n, given U 1, U 2, U 3, U L

plus n minus 1.

Now, what do we know about these terms? We know these are non-increasing, as L increases. So, among these n terms like you have U L plus one given U 1, U 2, U L, U L plus two. Uncertainty in U L plus two, given U 1, U 2, U L plus one upto uncertainty in U L plus n given U 1, U 2, U L plus n minus one. So, among all these terms this term is the greatest because we have proved from property two that this is a non-increasing function as L increases.

So, then we can upper bound this joint entropy by. So, this term I am writing exactly like this, but this whole summation, these n terms, this one, then up to this term, this can be upper bounded n times uncertainty in U L plus one given U 1, U 2, U L. And, this follows from the second property, which states that as L increases this uncertainty in U L given U 1, U 2, U L minus 1 is non-increasing.

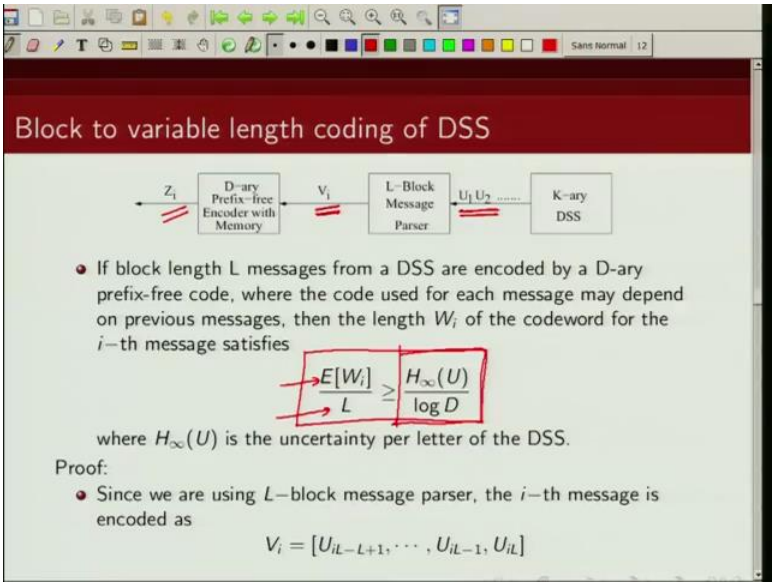
The next thing that we do is we divide both left hand side and right hand side by n plus L. So, if we divide both by n plus L, what we get is the left hand side we get this and right hand side we get this and this.

Now, we play a small trick. Instead of getting letting L go to infinity, we let n go to infinity. So, we let n go to infinity. So, if we let n go to infinity what do we get? What is this term? If n goes to infinity, this is nothing but the entropy rate. Correct. What about this term? Now look at the numerator, this is joint entropy of U 1, U 2, U 3, U L. So, this is a finite quantity. This divide by n plus L, where n goes to infinity, so as n goes to infinity, these particular term will go to zero. Now, what about this? This we can write as one plus one plus L by n. Now, as n goes to infinity this will become this. So, we have shown entropy rate is less than equal to uncertainty in U L plus 1 given U 1, U 2, U L.

Now, we let L go to infinity. If we let L go to infinity, then basically what we have shown is the entropy rate is less than equal to limit as L tends to infinity, H of U L plus one given U 1, U 2, U 3, U L. So note now, here we have proved that uncertainty per letter is less than equal to this uncertainty in U L plus one given U 1, U 2, U L, when L goes to infinity. Whereas, here we have proved that it is the other way round. Now, for both of these conditions to hold true, we would require the condition that both of them are equal. And that proves our fourth property that as L goes to infinity, uncertainty in U L given U 1, U 2, U L minus one is same as uncertainty per letter, when L goes to

infinity. And this as we said, we denote it by the entropy rate. So, this is the entropy rate of a discrete stationary source.

(Refer Slide Time: 29:14)



Now that we have proved some properties of discrete stationary sources, let us go ahead and see how we are going to design a block to variable length coding scheme for a discrete stationary source. We have already studied this problem for a discrete memory less source. And, if we recall we talked about optimal block to variable length coding, which was Huffman's coding for a discrete memory less source. So, this problem statement is as follows.

We have a discrete stationary source; this is a K-ary source. So, it emits K possible outcomes. So, this U i's are coming from this discrete stationary source. Now, since we are talking about block to variable length coding, so this source is parsed into blocks of fixed length, and that is blocks of length L. Now, these blocks are set to a prefix free coder. Now, since we are talking about sequence that is coming out of a discrete stationary source, which has memory, so we need to design a prefix free encoder which will exploit this memory. So, that is the difference from a discrete memory less case. In discrete memory less case, once we parse them into blocks of L bit, then we are doing prefix free coding. But, here we need to do prefix free coding, exploiting the memory of the source. And, output of this is my variable length code, which is denoted by Z i.

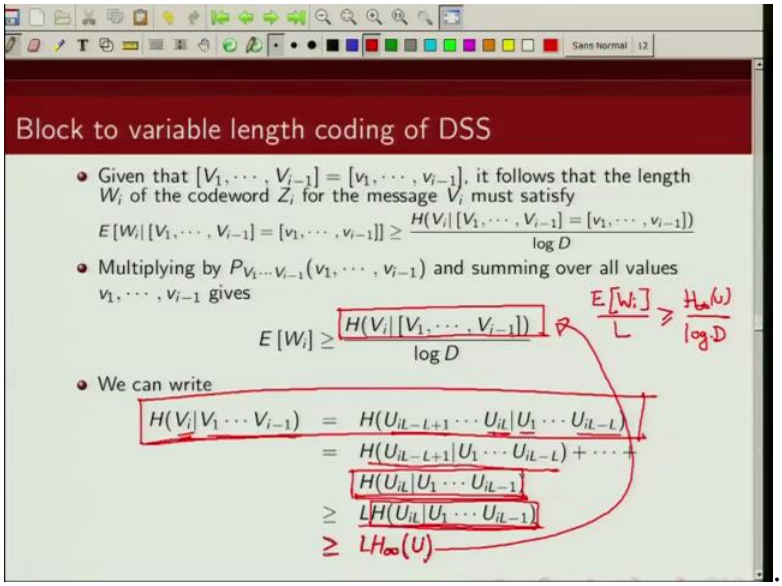
So, if block length L messages from a discrete stationary source are encoded using a D-

ary prefix free code, where the code used for each message may depend on previous messages, why we are saying this because our source has memory. So, it is probably likely that the sequences which happened in recent, past or more likely to happen in future also, as compared to sequences which happened long time back. So, we need to design our block to variable length encoder, which would exploit the memory of the source. And that is why we are saying that code word used for each message may depend on previous messages. Then, we can show that if the length of the i-th code word is W i, then expected code word length divided by input length. This is lower bounded by entropy rate.

So, we have already seen this in the context of discrete memory less source. And, we have seen that minimum number of bits per letter, in case of discrete memory less source is related to entropy of the source. We are going to show that for a discrete stationary source, this expected code word length by length of this code word block length parser. This is lower bounded by entropy rate. So, let us prove this.

So, let V 1, V 2, V i; this is denoted by this particular instance of this. Just a second, let us go back and see the notations. We are using U i's to denote the output of a discrete stationary source. Now this U i's, you are parsing them into blocks of L bits. And, we are using V i to denote this blocks. So, V 1 is U 1, U 2, U 3 up to U L. V 2 is U L plus U L plus two up to U 2 L. So, like that. So, V i is the output of the source parser; U i is the output of the discrete stationary source and Z i are the output of your block to variable length coder, which uses memory of the, exploits memory of the source.

(Refer Slide Time: 34:04)



So, the length of this codeword, output codeword Z I, so given you have this particular input to the BCC encoder, expected code word length given in this particular blocks of data V 1, V 2, V i minus one. This is lower bounded by uncertainty in V i, given this particular instance of V 1, V 2, V i minus one by log D. So, this follows from the fact that minimum number of bits that we are talking about block to variable encoding, minimum number of bits required to represent the source is related to entropy, in this particular fashion. And, this we have proved.

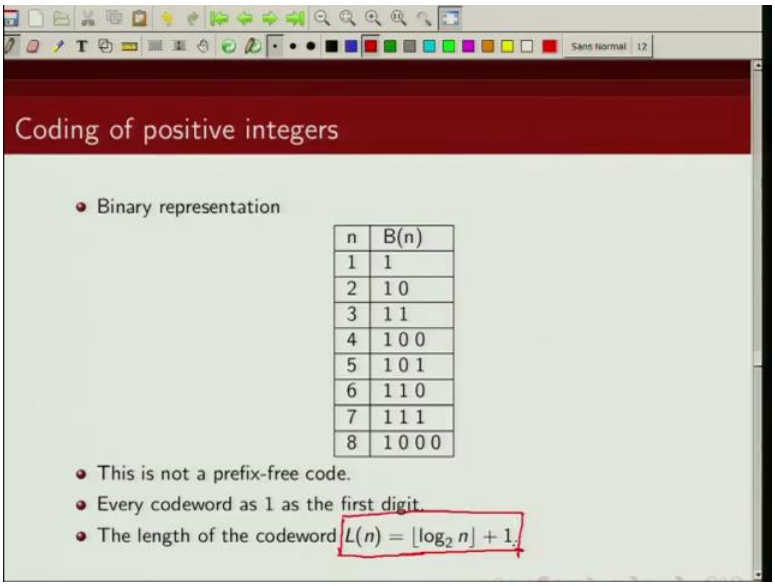
Then, we talked about block to variable length coding. Now, we want to get rid of dependencies on V 1, V 2, V i minus 1. So, what do we do? We multiply both sides probability of V 1, V 2, V 3, V i minus 1 and sum over all possible values of V i s. So, if we multiply the left hand side by probability of V 1, V 2, V i minus 1 and sum over all V 1, V 2, V 3, V i minus 1. What we get is unconditioned expectation. So, we get expected value of W i. And, if we do the same thing on the right hand side, what we are going to get is uncertainty in V i given V 1, V 2, V 3, V i minus 1.

Next, let us pay little bit attention on this term; uncertainty in V i given V 1, V 2, V 3, V i minus 1. Now, what is V i? So, V i is essentially U i L minus L plus one up to U i L. So, this is a block of L bit starting at this location to this location. And, what is V 1 to V i minus one? This is U 1, U 2, U 3 upto U i L minus n. So, from the definition of V i's we can write this in terms of uncertainty in U i's.

Next, we are going to apply chain rule. So, applying chain rule we can write this as uncertainty in U i L minus L plus one, given U 1, U 2, U 3, U i L minus L plus. You have terms like this. And finally, we have uncertainty in U i L, given U 1 to U i L minus one. Now, again from the property of the sequences that comes out of a discrete stationary source, we know that as L increases, this is a non-increasing function. So, then the smallest term here would be this particular term. And, there is L such terms. So, this right hand side can be lower bounded by L times H of U i L given U 1, U 2, U 3, U i L minus one. And, of course as I said this is non-increasing function, as L increases. So, this particular term will be less than equal to entropy rate. So, then we can further write that uncertainty in V i given V 1, V 2, V i minus one is greater than equal to L times entropy rate.

Now, we plug this value here and divide both sides by L. So, what do we get? We would get expected value of W i, given L is less than equal to this entropy rate by log of D. And that is what we wanted to prove. If you recall that is what we wanted to prove; that is what block to variable length coding. For a discrete stationary source, expected output code was length by input block size length L is lower bounded by this quantity, related to entropy rate of the discrete stationary source.

(Refer Slide Time: 39:33)



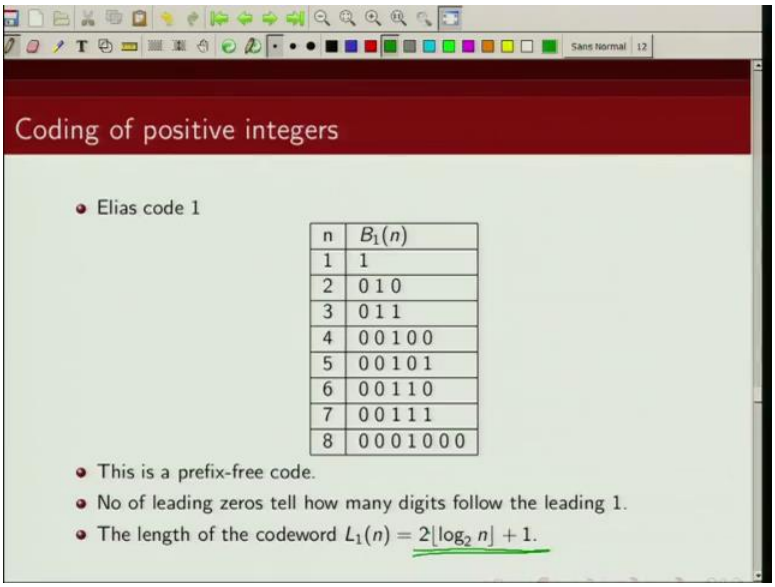
Now, we are going to show how we can do block to variable length coding for a discrete stationary source. And, to do that I need to first give a scheme, which will encode my

positive number. So, I will, I am going to talk about a scheme to design coding for positive numbers.

So, you are all familiar with binary representation of positive numbers. So, let us say I have this n, which is 1, 2, 3, 4, 5, 6, 7, 8. So, what is this binary representation? 1 is represented by 1, 2 are represented by 10, 3 is represented by 11, 4 is represented by 100, 5 is represented by 101, 6 is represented by 110. Similarly, 7 is 111 and 8 is 1000. So, like that you can write any positive number by its binary representation.

Now, is this a prefix free code? The answer is no. For example, you can see this is a prefix of this code. This is same thing of prefix of this code. Similarly, this is a prefix of this code, this code. So, this is not a prefix free code. So, if we are designing a block to variable length coding, remember we want for instantaneous decodability; we would like to design a prefix free code. So, one drawback of this binary representation of positive number is this is not a prefix free code. Now, what is the good part about this binary representation? The number of bits required to represent this number that small. So, code word length can be given as floor of log n to the base two plus one. So, number of bits required representing the number, positive number that is small; however, this is not a prefix free code. So, would not it be nice to design a prefix free code, which would have the number of bits required to encode close to this? So, that is the goal. So, Elias came up with a scheme to encode these positive numbers. So, I am calling this as Elias code one.

(Refer Slide Time: 42:10).



Now, what he did here was; So, let me show you. This is my one, binary representation of 1, binary representation of 2, binary representation of 3, binary representation of 4, 5, 6, 7 and 8. Now, what did we have before this? So, we had a runs of zero to denote how many bits followed. So, in case of 2, binary representations of 2, it is 10. There are two bits that are following. So, he denoted by the one run of zero. In case of 1, there is just one bit. So, that is nothing (Refer Time: 43:09). In case of 3, it is 11. So, we denoted that that by one run of zero. Similarly, 4 is 100. That is, three bits are required. So, he made this binary representation a prefix free code by adding a runs of zero as the prefix. So, by looking at the runs of zeros you can guess, you can find out how many bits follow this.

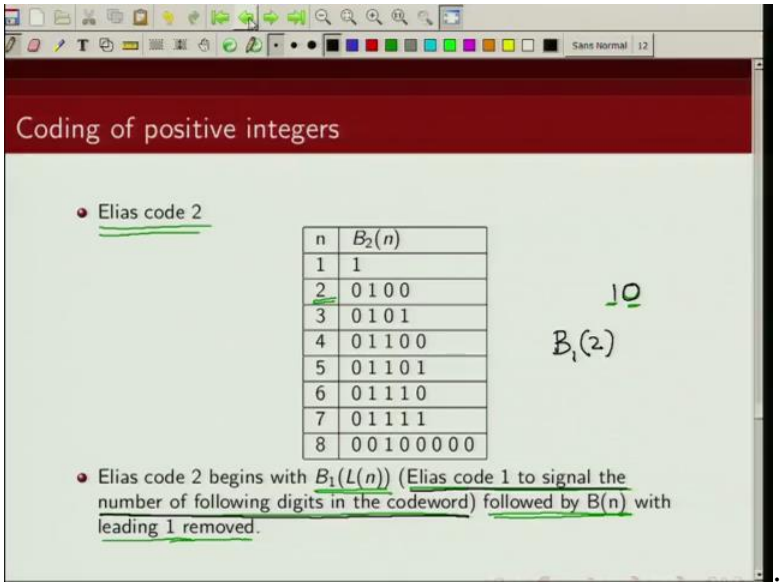
So, for example, if this is zero zero, then he knows that next three bits are the binary representation of the number. So, you can take, for example, 8. Now 8; the binary representation of 8 is 1000. So, there are four bits. So, if you prefix it by three runs of zeros, you get what is Elias code one.

So, the good thing about this code is it is a prefix free code. And, what is the bad thing about this code? That is the length. You are using almost same number of bits. I mean, actually the same minus one number of bits to denote the runs of zeros. So, as I said in this particular code, the numbers of leading zeros are telling me how many digits follow the leading one.

There is one more observation you should take note of. When we are writing binary representation of numbers, all of these numbers start with one. You see here 1 is 1, 2 is 10, 3 is 11, 4 is 100, 5 is 101. So, like that. So, in Elias code one number of leading zeros tells how many digits will follow the leading one. So if there are three zeros, then after this leading one, there will be a three more digits that we need to look at. And, this leading one followed by the other three digits, that is, the binary representation of the number that is we are talking about. Now, as I said the bad part about this code is you are now requiring two times floor of log to the base two of n plus one. That is why the length of the code word is almost double of what was required for binary representation.

However, the good part about this code is, this code is prefix free. So, then how can we design a code which is prefix free; however, the codeword length is smaller, especially for large (Refer Time: 46:29)?

(Refer Slide Time: 46:30)



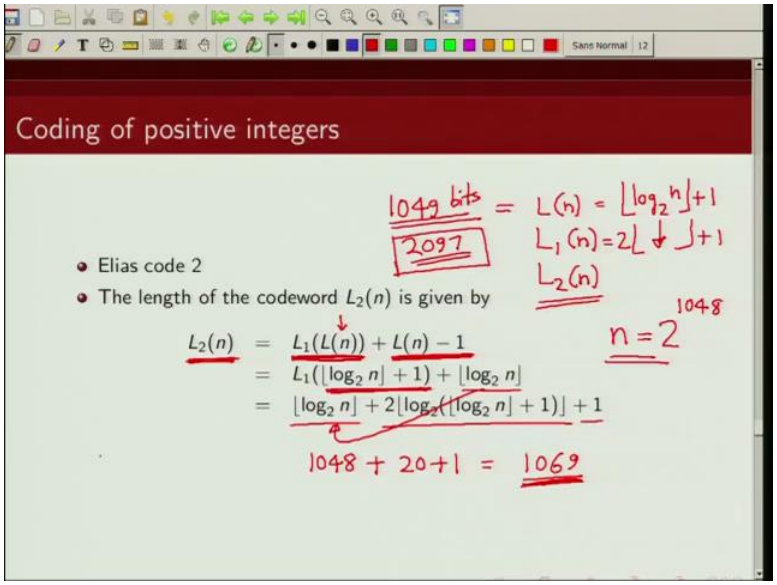
So, Elias came up with a trick and that is as follows. So, Elias two codes begin with Elias one code. So, Elias code one is used to signal the number of bits in the code word. So, earlier what we, what was used in Elias code one? We were using the runs of zeros. So, here that is replaced by Elias code one to signal the number of following digits in the code word, and followed by the binary representation of the number with leading one removed. As I said in binary representation of positive number, each of the number starts with one. So, that is the redundant information. So, what is done in Elias code two is you use Elias code one to signal the number of following digits followed by the binary representation of the number with leading one removed.

So, let us take an example. Let us take this; n equal to 2. So, what is the binary representation of 2? That is 10. Correct. Now, how many digits are there in 2? that is two; one, two. So, in the Elias code two we are going to use Elias code one for the number 2 because binary representation of 2 requires two bits. As I said in this particular scheme, Elias code one is used to signal the number of following digits in the code word. And in case of 2, there are two words, two bits; 1 and 0. So, we are going to use Elias code one corresponding to 2. And, what is that? Elias code one corresponding to 2 is 010. So, then 010 will illustrate or show that. So, this is the Elias code one for 2. And, this is followed by a binary representation of 2 with leading one removed. So, 2 is 10. We are removing this leading one. We are just writing zero. So, the Elias two codes for 2 will be 010 followed by 0.

Take another example. Let us say take this 6. Now, what is 6? 6 is, binary representation of 6 is 110. So, you require three bits to denote 6. That is, 110. So, in Elias code two which we are denoting by B 2 of 6, we first need Elias code one of for three because binary representation of 6 is 110. So, what is Elias code one of 3? That is 011. So, what we are going to do is we are going to first write 011. And then, this has to be followed by binary representation of 6 with leading one removed. So, if we remove this leading one, we are left with 10. So, that is your 10. So, then Elias code two for 6 is 01110.

So to repeat, we are using Elias code one to denote how many digits are there in the binary representation of that number and then we are following it by binary representation of number with leading one removed.

(Refer Slide Time: 50:59)



Now, that is the beauty of this particular code. Number one: this is a prefix free code. And, let us compute the length of the code word. So, remember the first part of the code is we are using Elias code one to denote the number of bits required in the binary representation of the number. So, the first part of the code length will begin with L 1 of L of n. The second part of the code is binary representation of the number n with leading one removed. So, that is why I am writing it as L n minus one.

Please note I am using this notation L n to denote the length of the code word for the binary representation of the positive number n. I am using this notation L1 n to denote the length of the code word, which has been encoded using Elias code one. And, I am

using this notation L 2n to denote the length of the code word, which is encoded using Elias code two. So, the length of the code word, Elias code two is the first part is Elias code one of length L n. And, the second part is binary representation of the number with leading one removed. So, this length will be given by this.

Now, what is L of n? We know L of n is given by floor of log to the base two of n plus one. So, if we plug that value of L n here, what we get is the length of this is L 1 of this quantity plus floor of log to the base two of n. Now, what is L 1 n? If you recall L 1 n is given by this quantity. It is two times floor of log to the base two of n plus one. So, we have to compute L 1 of this quantity. So, then this will be two times this plus one, and this term comes here. So, if you compare the length of this code word, Elias two codeword is given by floor of log to the base two of n plus two times log to the base two of this term plus one. Now to appreciate, what do we gain by going for Elias code two.

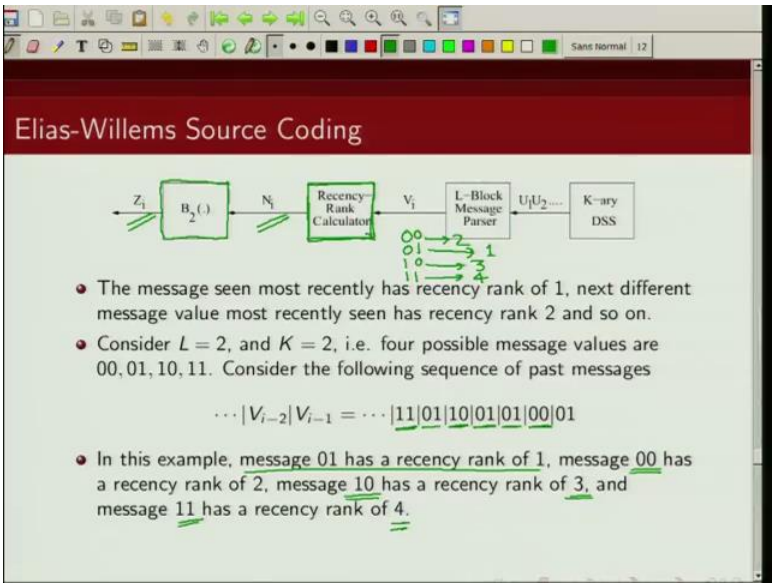
So, let us take an example. Let us take n. Let us take n to be a large quantity. Let us just take it n is 2 to the power 1048. Let us just take this. Now, in case of binary representation of one, so binary representation of this number, how many bits are required? We require log two n plus 1. And this, for this value of n this will come out to be 1048 plus one. So, this will be 1049 bits. But, the problem with binary representation is V is not a prefix free. Now, if we look at L, L 1 n, then the number of bits required is two times this term plus 1. So, this comes out to be then 2096 plus 1, 2097. So, it is almost double.

Now, look at Elias code two. So, here we get 1048. And, this is; this will be log to the base two of n, so 1048 plus 1. I mean, this I am ignoring one. So, log of 1049, roughly 1048. So, that would be ten; so, two times 10, that is 20, plus 1. So, this is coming out to be 1069.

So, note this is a prefix free code, but we are requiring lot less number of bits compared to Elias code one. Now, we can repeat this. We can design an Elias code three, which will use Elias code two to denote the length of the L of n, followed by binary representation of the number with leading one removed. So, for large values of n we can see that if we repeat this process, these are prefix free code. And, the number of bits required is coming closer to the binary representation of the number. So, the Elias code that we just described. We are going to make use of that in our block to variable length

coding problem for discrete stationary source.

(Refer Slide Time: 57:15)



So, this Elias Willems source coding technique is as follows. So, we have a discrete stationary source, a K-ary discrete stationary source, whose outputs are denoted by U i. Now, we have a block parser which is creating blocks of this U i's. So, V i is essentially a block of L U i's. Now, we will just mention what we mean by recency rank calculator. If you recall, since this is a discrete stationary source, the block to variable length encoding should exploit the inherent memory of the source. And that will be exploited by this recency rank calculator. And, since we are talking about block to variable length coding, for instantaneous decodability we would like to design a prefix free code. And, for that we are going to make use of Elias code two.

So, the new element that you see here which is different from what we have seen for block to variable length coding for discrete memory less source are these two blocks. So, we are using a prefix free code. However, we are exploiting the memory of the source. And, we will tell you in a little while what do we mean by recency rank.

So, the message that is seen most recently has recency rank of one, and then next different message most recently seen as this is the rank two and so on. So, recency rank is essentially an indication of how far in the back, in the past, you have encountered that particular sequence.

So, let us consider a very simple example. We are considering a block of length two. And, let us consider a binary discrete stationary source. So, when U i's are essentially zeros and ones, and since my block length is two, so the possible outcomes V; possible V i's could be 000110 or 11. Now, what this recency rank calculator will do is will look at the occurrence of these among the past sequence. And, whichever sequence occurs most recently will be given a recency rank of one. Then, the next one will be given recency rank of two, the third one will be given recency rank of three, and similarly the last one will be given recency rank of four.

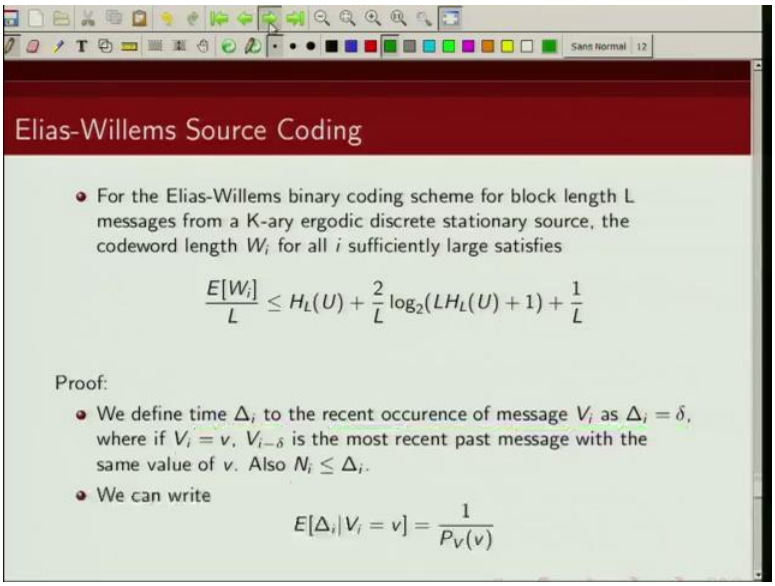
So, let us consider that this is a sequence that you have encountered in the past. So, this is, this is V i minus one, this is V i minus two, this is V i minus three, V i minus four and so on. Now, which among these four sequences have occurred more recently? That is 01. So, 01 will be assigned a recency rank of one. So, message 01 will be given a recency rank of one. Let me write, mark the sequences. So, this has recency rank of one.

Now, next 00; occurs in the past. So, this will have a recency rank of two. Then, 01 already has a recency rank of one. Next sequence that we see is 10. So, this will have a recency rank of three. And then, you have 11. So, this will have the recency rank of four. So, this is how we are assigning a recency rank. So, this block that I mentioned is essentially emitting a positive number. And that is an indication of how long in the past that particular sequence is occurred. So, as I said 01 has a recency rank of 1, 00 has a recency rank of two, 10 has a recency rank of three and 11 has a recency rank of four.

Now, what this Elias code will do is it will code these positive numbers and the resulting code is going to be a prefix free code. And, the memory of the source is exploited in this recency rank calculator.

So, what we are essentially assuming is the sequences that appeared in recent, past or more likely to appear in the future. And that is why they have been given a smaller recency rank. So, if we encode those positive numbers because the sequences which happened in recent, past will have smaller recency rank number. So, when we will encode them using B 2 code, the effective expected code word length would be smaller. So, in the Elias Willems source coding scheme, this recency rank calculator assigns a recency rank to the message V i, which is a block of L bits.

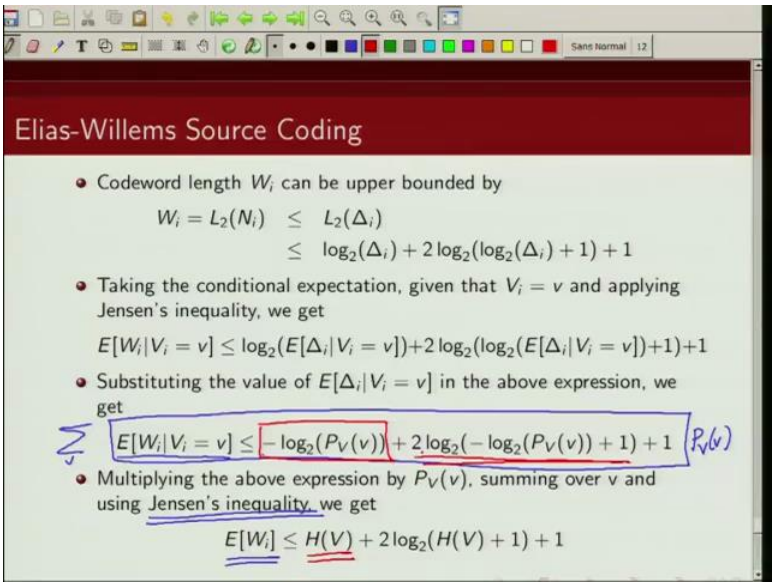
(Refer Slide Time: 63:38)



So, now we are going to calculate what is the expected code word length per input block length for this Elias Willems source coding technique. So, we are going to show you that for Elias Willems source coding technique, where input is a block length L and we have a K-ary discrete stationary source, the expected code word length divide by L is upper bounded by this quantity. And, if L goes to very large you can see these terms becoming smaller and smaller. So, this term is coming closer to entropy rate plus small term.

So, let us define delta i as the recent occurrence of the message V i. So, delta is this delta where if V i is V, then V i minus delta is the most recent past where this particular message has occurred. Now, let us go back to this example. So, let us say V i is 00. So, we have to go back in the past and see when did - 00 occur. So, we can see this is V i minus one, V i minus two. So, this is delta corresponding to two. So, delta i in this case will be two. Now, if you consider 11. So, when did 11 occur in the past? i minus 1, i minus 2, i minus 3, i minus 4, i minus 5, i minus 6, i minus 7. So, delta i is going to be seven. So, that is what I am saying that we define this time delta i as the recent occurrence of this message V i. So, if delta i is equal to delta, if V i is V, then V i minus delta is the most recent, past, when that particular value of V has occurred. Now, you can see clearly that delta i is going to be greater than the recency rank.

(Refer Slide Time: 66:40)



So, what is the expected code word length? The expected code word length is given by the length of Elias code two of N i, recency rank. Now, we have just now shown that this recency rank N i is less than equal to delta i. So, this code word length W i is upper bounded by the length of Elias code two of delta i.

Now, we know what is the length of the Elias code two. This is given by; if we just go back and refresh your memory, this is given by this expression. Now, if I remove these floors I can write length of Elias code is upper bounded by log to the base two of n plus two log to the base two into log to the base two plus one plus one. So, that is what I am doing here. So, this follows from the length of the Elias code. Correct.

Next, this is one thing which I skipped, which is the, what is the expected value of delta i given V i is V. So, that is inversely proportional to the probability of occurrence of V. So, that is basically given by this.

Now, we take conditional expectation of W i, given V i is V. And, we invoke Jensen's inequality. So, if we take conditional expectation of, this on the left hand side what we will get expected value of W i, given V i is V. So, that is this term.

Now, if you take expect, conditional expectation of this, so this will be expected value of log to the base two of delta i, given V i is V. Now, what do we know about log? Log is a concave function. And, if this is a concave function, what do we know? From Jensen's

inequality, we know if it is a concave function, then the expected value of the function is less than equal to function evaluated at expected value. So, if we take conditional expectation of delta i, a log of delta i to the base two, given V i is V. Since, log is the concave function, we can say expected value of this log function will be less than equal to log evaluated expected value of delta i. So, that is why we see here. So, there are two steps involved here. We are first taking conditional expectation of this term. So, we are taking conditional expectation of this given V i is V.

Next, what we are doing is since log is a concave function, we are saying expected value of the function should be less than equal to function evaluated at expected value of x. So, what is a function? Function is log; that is this. And, what is expected value of x? So, expected value of x in this case is expected value of delta i, given V i is V. So, what we did was we did two things here. We took conditional expectation, condition on V i is V. And, we invoked Jensen's inequality. The same thing we will do here for this particular term. We first take conditional expectation of this; given V i is V. And, since log is a concave function, then the expected value of the function is less than equal to function evaluated at expected value. So, that is why we move this conditional expectation inside. And, we can write this as two log log of expected value of delta i, given V i is V.

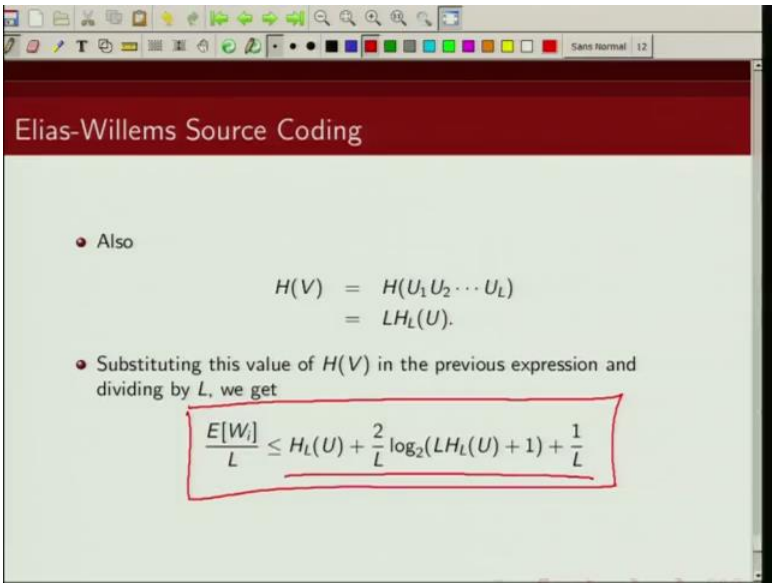
Next, we know what these terms are. From the ergodic property, we said that this is given by this. So, we plugin the expected value of delta i, given V i is V, into this expression. So, if we do that we will get this. Now, we need to get rid of this conditional expectation because we want to get the value of expected value of W i. So, to do that what we do is we multiply this particular expression. We multiply by probability of V and then we sum over all V.

And, another thing that we are going to use is we are going to make use of Jensen's inequality. So, when you multiply the left hand side with probability of V and sum over all V, we will get unconditional expected value of W i. Now, look at the first term on the right hand side. If I multiply this by probability of V and sum over all V, what I am going to get is uncertainty in V.

Now look at this particular term, if I multiply this by probability of V and sum over all V, this is like taking expectation of log log function. Now this, since log log is a concave function, and I can again say that expected value of the function will be less than equal to

function evaluated at expected value. So, what I am going to do is I am going to, and then by invoking Jensen's inequality, I can write this as two log of, I multiply this by P of V and sum over P of V. So, this particular term that you see here will become uncertainty in V. This comes as it is. And, this one comes here. So, again I use two things. I multiply it both sides by probability of V, sum it over all V and I invoked Jensen's inequality, which says that if you have a concave function, then expected value of that function is less than equal to function evaluated at expected value. And that is what I used here.

(Refer Slide Time: 75:28)



Now, let us pay attention to this particular term H of V. Now, H of V is what? What is V? V is this block of L bits, which are coming from a discrete stationary source. So, H of V is nothing but uncertainty in U 1, U 2, U 3, U L. And, this I can write in terms of uncertainty per letter that L times H of L U. Now, in this particular expression that I derived, I am going to replace H of V by L times H of L U. If I do this, I get this expression that expected. And, I divide by L throughout; I get this expression that expected code word length per input length per input bit is upper bounded by this quantity. So, this gives us upper bound on expected number of bits required to encode the sequence, which comes out of a discrete stationary source. So, this is just one simple example to illustrate how we can do, exploit memory to encode a source which has memory.

So with this, we will conclude our lecture on coding for sources with memory.

Thank you.