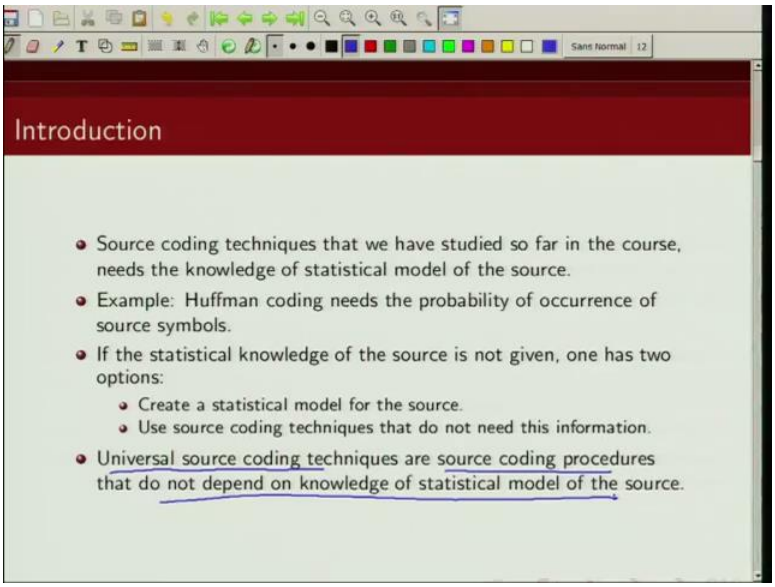
An Introduction to Information Theory
Prof. Adrish Banerjee
Department of Electronics and Communication Engineering
Indian Institute of Technology, Kanpur

Lecture – 7A
Universal Source Compression Algorithms

Welcome to the course on An Introduction to Information Theory. So, today we are going to talk about universal source compression algorithms. The source compression algorithm that we have studied so far requires you to have knowledge about the source distribution a prior. Now, many times we do not have that information with us, right. So, how do you do source compression? So, that is our topic of discussion today. So, we will under this, will talk about Lempel Ziv algorithm. In this class, we will talk about Lempel Ziv 77 algorithm and in the next lecture we will talk about Lempel Ziv Welch algorithm. So, we will talk about encoding and decoding of Lempel Ziv 77 algorithm in this particular lecture.

(Refer Slide Time: 01:08)

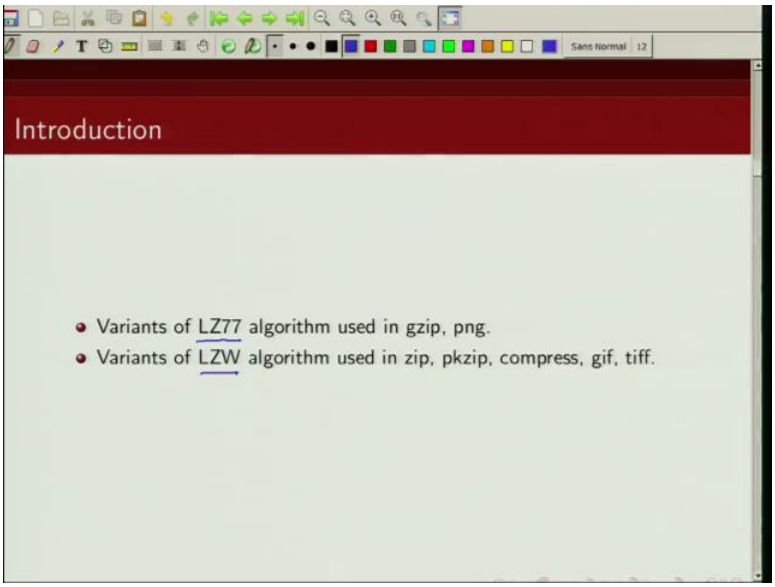


So, as I mentioned the source compression algorithm that we have talked about so far take Huffman coding, Tunstall coding, Shannon Fano codes. They all require a priori the knowledge about the source distribution. If we do not have that knowledge, the source distribution you cannot do optimally the source compression. So, in many times we do not have that information available with us. So, what if you do not have that information

available to us? What should you do? One possible thing is we could create a statistical model. So, you have some initial training phase, you try to guess the probability of these various symbols which have been emitted by this source and you create a statistical model. Other technique could be using source compression algorithm that do not require this a priori knowledge about the source distribution and today we are going to talk about such source compression algorithm that do not require a priori information about the source distribution.

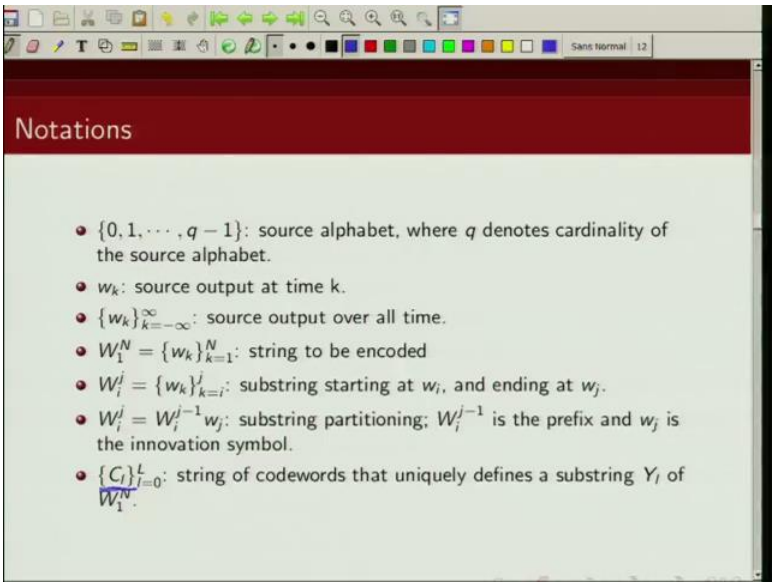
So, universal source compression algorithms are the source coding algorithms that do not require any knowledge about the statistical model, about the source. They are very efficient. We are not going to talk about in this class about the optimality of these algorithms, but asymptotically they are also optimal.

(Refer Slide Time: 02:52)



Now, these algorithms are widely used in practice. For example, variant of this LZ77 algorithm is used in this compression software gzip and png, whereas z compress tiff uses a variant of Lempel Ziv Welch algorithm. So, we will talk about these two Lempel Ziv algorithms in this this lecture and thus next lecture.

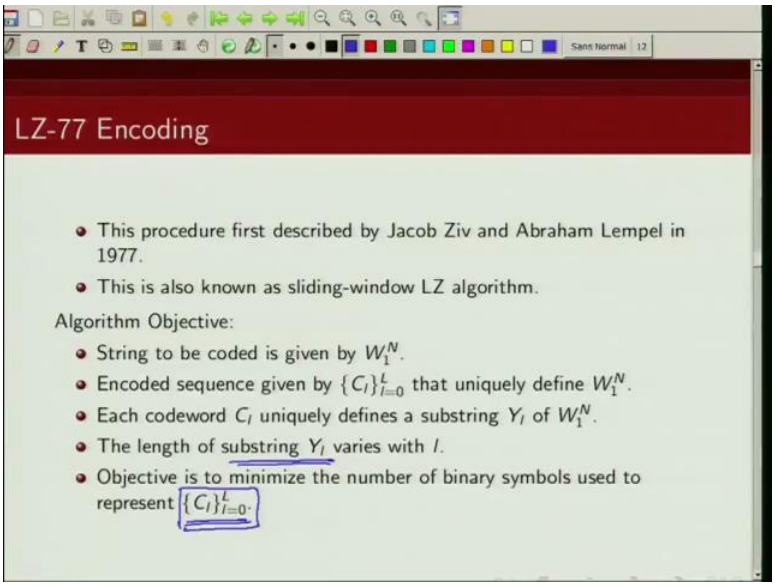
(Refer Slide Time: 03:27)



So, let us first describe the notations that we are going to use for this particular lecture. So, I am going to denote my source alphabet by 0 1 2 q minus 1, where q is cardinality of my source alphabet. The source output is denoted by small w and k denotes the output at time k. So, if I write source over possible times, then I will use this notation w k, where k goes from minus infinity to infinity. Now, the string that I want to compress is a block of length n which I denote by this capital W and I use these indices 1 to n to denote, it is a block from 1 to n. So, this can be as string which goes from k equal to 1 to n. I use this notation w i j to denote a substring starting at k equal to i and going up to k equal to j and I can write this string from i from k going from i to j as a string going from i to j minus 1 concatenated with another output at time j.

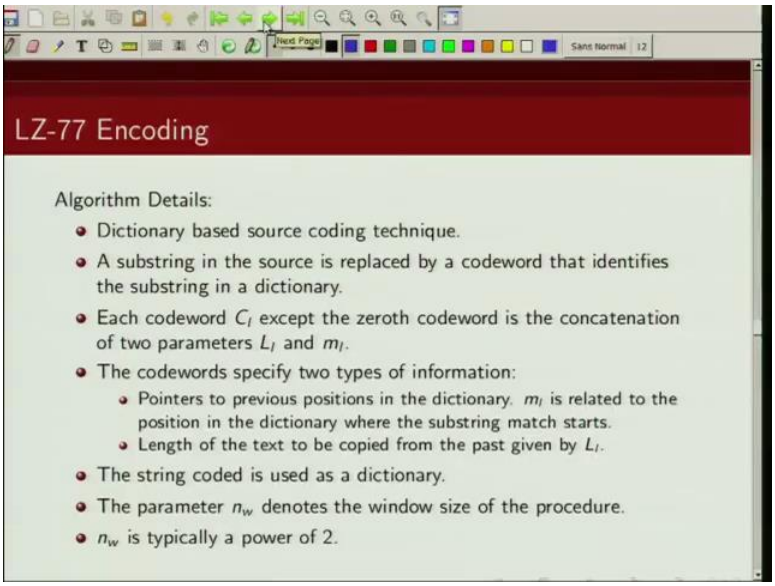
So, w i j minus 1 can be considered as a prefix and w j is what we call an innovation symbol that makes w i j different from w i j minus 1. The string of code words we will denote by c. So, c of l where l goes from 0 to l basically l set of l code words that those where denoted by c of l.

(Refer Slide Time: 05:42)



So, this Lempel Ziv 77 algorithm was given by Jacob Ziv and Abraham Lempel in 1977. This is also known as sliding window version of Lempel Ziv algorithm and it is a dictionary based source compression algorithm. We will talk about that. So, we are given a string of n bits that we want to compress and set of code words is given by c of l and these code words uniquely determine the substrings or this set of encode code words complete describe the string that we are encoding now each code word. So, they said this block of n bits is described by code words c l codes from 0 to l. Now, each of this code word c l uniquely determines a substring y l of this block of n bits. The length of the substring is variable and our objective is to minimise the number of bits require to represent this string.

(Refer Slide Time: 07:15)



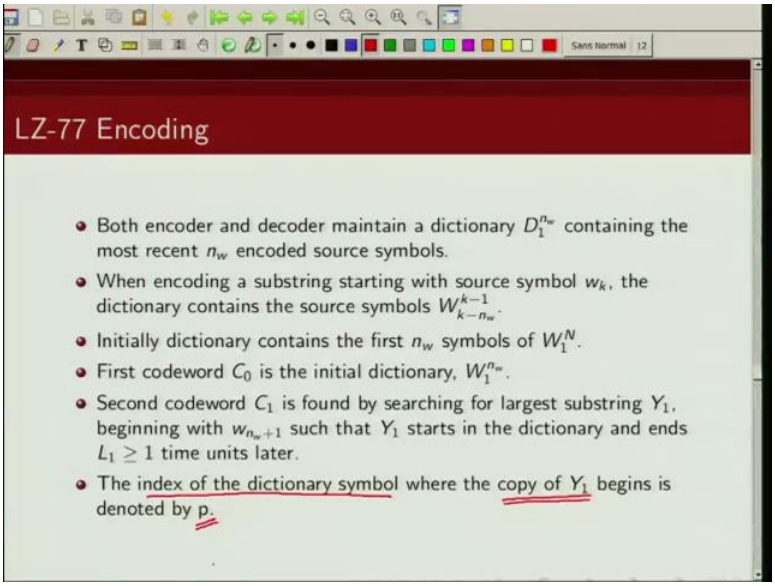
So, we would like to minimise a number of bits required to represent our set of co-ordinals. So, as a state this is a dictionary based source compression algorithm and we will talk about how we create a dictionary. So, a substring in the source is replaced by a code word that uniquely identifies that substring in the dictionary. So, we are going to represent this string. So, you can think of it as this string that we want to encode can be partition into smaller substring and corresponds each of these substrings. There is a code word now do we partition it into the substrings depends on what is there in our dictionary and we will talk about that each code word except the initial code word which I called as 0th code word, each code word except the 0th code word is concatenation of two parameters. One is the length of the match of substring which I denote by capital L1 and Ml which denotes the location where the matches occurred.

So, as I said it is a dictionary based encodings. So, the strings of bits are there in the dictionary. So, whenever the new string comes in or we are searching for the largest newer string which is already there in the dictionary, so we are interested in finding out the length of the string that is match or string which is already there in the dictionary and we are also interested in finding where in the location of the dictionary, this match is occurring. So, we are interested in two things. When we specify code words except the 0th code word and these two things as I said are these parameter L and M.

What is this parameter L? L is the length of the text from the given past which is there in the dictionary which got mapped, matched and this M is related to location in the dictionary where the match has happened. So, our set of code word is as follows. Initially the set of code word is essentially set of bits that we are encoding like. So, we have a dictionary of window length and w that is much smaller than the block of data that we are trying to encode.

So, initially we will send the first end, but that is our initial dictionary and then, subsequently what we are trying to do is, we are trying to find the largest match of the string. So, starting from location n w plus 1, we are looking at largest string which gets map to a string already present in the dictionary and we are interested in knowing how long is that string matched and we are also interested in knowing the location of the match. What we are going to do is, we are going to code these two parameters, L and M and this will be our code word. As I mentioned, this is our sliding window kind of version of Lempel Ziv algorithm. So, I have window of size n w and typically this is taken as power of 2.

(Refer Slide Time: 10:59)



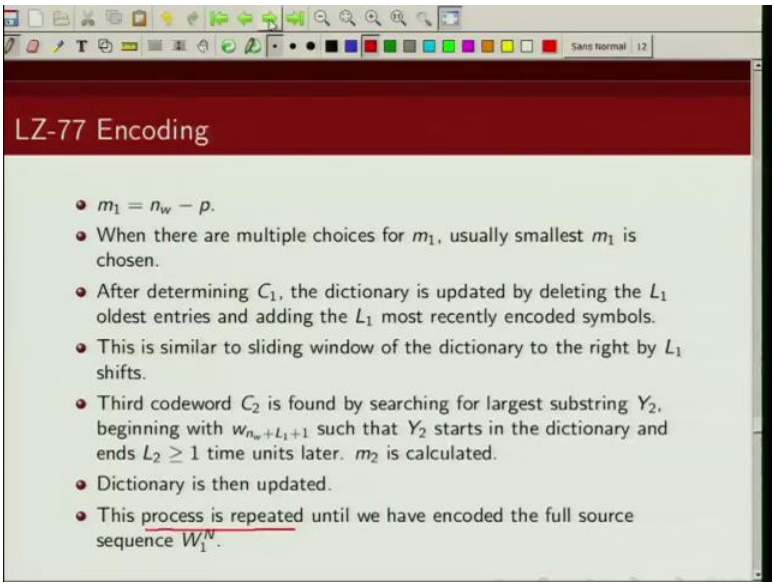
So, in this scheme both encoder and decoder, they maintain a dictionary containing most recent n w bits. So, both encoder and decoder are maintaining a dictionary of most recent received n w encoded sources bit symbols. When encoding a substring starting from a source symbol w k, the dictionary contains source symbols from w k minus 1 k minus n

w to k minus 1. So, if you have a string of bits, this is string of bits and I am looking at kth location. So, what you are going to see is from location k minus 1. So, this location k minus n w to k minus 1, these bits are there in our dictionary. So, this dictionary is of size n w and this is in the dictionary and you are looking at the kth bit which is w k. Now, how do you encode this string coming out of w k? What you do is, initially the first n w symbols of your sequence that you want to encode are initially sent to the decoder. So, initially the decoder is initialised with the first n w symbols of this string that you are trying to encode.

So, the first code word c 0 is nothing, but initial n w contents of the dictionary. Next second code word c 1 is found by searching for the largest of substring y 1 beginning at location n w plus 1, such that y 1 starts in the dictionary and end one times later. So, what we are doing is, initially as I said 1, 2 and w, this is my initial n w symbols. They are in the dictionary. Now, I look at the next symbol which is n w plus 1. Now, what I am going to do is, I am going to look at the apex stream, largest stream starting at location n w 1 which is already there in the dictionary. So, my dictionary contains as I said my dictionary contains symbols from location from time k minus n w to k minus 1. So, some symbols of length n w in the dictionary now at location n w plus 1, I am trying to look at what is the largest string which is already there in the dictionary. So, this match let us say this is 0 1 0 1. So, I look at in the dictionary, where my 0 1 0 1 is there.

So, I look at first in the dictionary where my 0 is. So, there may be multiple spaces where 0 is, then I look at 0 1 which are the location, where 0 1 is there. I will may be there are few of them. When I look at 0 is 1 0, where there a match with 0 1 0, then I look at 0 1 0 1. So, I am trying to find the largest match which is already there in the dictionary and please note this match has to start inside the dictionary. So, that is what I meant. I am searching for the largest substring beginning at time n w plus 1, such that the substring starts in the dictionary and end one time later.

(Refer Slide Time: 15:45)



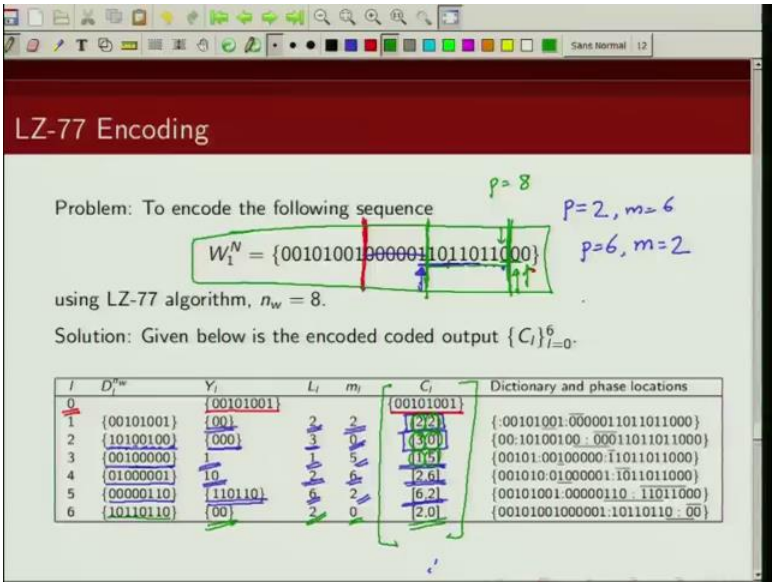
Then, I also need to find out where in the dictionary this match is starting and that is given by this p. So, the index of the dictionary symbol, where copy of y 1 begins is denoted by p and m is given by n w minus p. So, let us say my n w is 16 and this match happened at third location. So, then m 1 could be 16 minus 3 which is 13. Now, there might be a multiple places, where I have match of the largest string, right. So, which value of m should I choose? I should choose the smallest value of m. The reason being if I am using where ever length coding to describe these m, the smaller the m I will require lesser number of bits. However, if I am using prefix number of bits to represent m, then it does not matter what choice of m I use. So, once I have found out the largest map of the string y 1 inside the dictionary, what I am going to do is my code word c 1 is nothing, but this pair of length of match and this m 1. So, this will give me the code word c 1.

Next I am going to update my dictionary. Now, how I am going to update my dictionary? I am going to update my dictionary by deleting the one oldest entries from the dictionary and adding this new one bits. So, look at this. So, I had these n w bits in my dictionary, right. Now, I looked for the maximum match, so this string y 1, this match of length 1. So, what I am going to do is, I am going to remove this one oldest bit from the dictionary and my new dictionary is now this. So, note my dictionary size remains same. It is just sliding by dictionary style is sizes still n w. I deleted this old one bits symbols from the dictionary, instead I added this new one symbol corresponding to the substring y 1 and I repeat this.

Next I will look at the largest strings starting from this location which is already there is my this dictionary and let say that substrings comes out to be y 2, then I will again find out what is the length of the match and I will find out the location of the match which is related to m 2 and then, I will add this l2 bits to my dictionary and remove old one. I remove this old one 2 bits. So, my new dictionary now will be this. So, this process of sliding the dictionary and finding the largest substring which is already there in dictionary, this process will continue until I come to the end of the block which I want to encode.

So, that is what I said after I determine this code word. The third code word is found while finding the largest substring y 2 beginning at this location n w plus l, 1 plus 1 such that y 2 starts in the dictionary and ends L2 times units later and I also need to calculate the location, where the matches are as happened that index is given by p and p and m are related by this relationship. Again I will update my dictionary. I will add these substrings y 2 which is of length l2 and I remove the oldest one 2 entries. So, this process is going to be repeated until the full sequence has been encoded.

(Refer Slide Time: 20:19)



So, let us take an example to illustrate what we have said. So far we have a sequence of bits which I denote. So, binary bits which I want to encode using Lempel Ziv 77 algorithm and it is given that the dictionary size is 8. So, then what is the first step of the initial code word is nothing, but first n w symbol. So, then you can see at l equal to 0, I

look at the first eight symbols 1 2 3 4 5 6 7 8. So, this is my initial contents of the dictionary which I am writing it here. So, initial code word the 0th code word is nothing, but these n w bits. So, the receiver has now this dictionary.

Now, what is a next step? The next step is starting from this location I need to find the largest map match in the dictionary. So, let us look at is 0. There is 0 there in dictionary. Dictionary has this. Yes 0 is here, 0 is here, 0 is here and 0 is here and 0 is here. What about 0? 0 is 0. There in the dictionary, the answer to this is yes. Look here, this is 0, 0 starting from this location is 0. 0 of course, at this 0, 0 match is not there, but at this location we have 0 0. What about 0 0 1? Do we have a match of 0 0 1? So, where are the two places? We had match at this location, but this is 0 0 1. So, we do not have a match of 0 0 0. What about this? So, we have map of match of 0 0 0. The answer is no because this is 0 0 1. So, the larger substring that is matched with the symbols in the dictionary is 0 0. So, the substring y 1 is 0 0. So, that is what I am writing.

Now, what is the length of the match that is 2? Now which are the locations where the match has happened? So, if I write this as 1 2 3 4 5 6 7 8, the match has happened at p equal to 1 and at p equal to 6. So, correspondingly m is n w minus p which is 7 and in this case it is 2. So, since this is smaller, I am taking this m. So, that is my m. So, then my code word consist of this length of 2 and the location which is denoted by this 2. How to I encode this l and m? I will come to that later. I am now just writing passing my sequence into the match length of the match and the location of the match now. So, I have found string y 1 which is 0 0 which is matching with whatever is there in my dictionary at location which is given by m equal to 2.

So, next step is I am going to add these two symbols into my dictionary and I am going to remove the oldest two times. So, my new dictionary is given by this. So, that is what I have here. You look here. We do initially is 1 0 0 1 1 0 0 0 0 1 0 0 1 0 0. So, I added this string y 1 which was of length 2 and remove the two oldest entries from the dictionary.

So, my new dictionary is now this. So, my new dictionary is this. This is my new dictionary, content of my new dictionary. Now, again follow the same procedure, start from this location 0 is there a match of 0. Yes, there is a match here, there is a match here, there is a match here, there is a match here, there is a match here, and there is a match, right. What about 0 0? Now, here there is no match of 0 0, but here there is a match of 0. Here there is no

match of 0 0. So, I just remove this here. There is a match of 0 0 and here also there is a match of 0 0 because this 0 is followed by this 0. So, the match has to start within the dictionary, but it can go beyond. So, so if you look at, there is a match of 0 0. What about 0 0 0? Now, at this location this is 0 0 1. So, this is not a match, but what about this at this location? It is 0 0 0. Similarly at this location, this is 0 0 0. So, there is a match of 0 0 0. What about 0 0 1. Look here this is 0 0 0 0. So, that is not a match. What about here at this location? This is 0 0 0 0. So, clearly there is no match of 0 0 0 0 1. So, the largest match is 0 0 0.

So, this substring y 2 is 0 0 0. So, then the largest map is this 0 0 0. What is a length of this match? That is 3 because the string of length 3 is getting mapped and what is the location. So, this match happens at this location which corresponds to p equal to 7 or the match happens at this location which happens to be p equal to 8. So, m is either 1 or m is equal to 0. So, that is how we take the smallest m. So, we take m to be 0. So, corresponding this to the string y 2, we have a length, we have a string of length 3 and location m given by 0.

Now, the next step is we are going to add these string y 2 to the dictionary and we are going to remove the three oldest strings. So, our new dictionary is this and that is what I have written here. You can check 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0. So, the new dictionary is given by this. Again we follow the same procedure. So, we look at the first bit one. If there a match of one here, do you see a match of one here in this dictionary? Yes there is that is this. What about 1? One is there a match of 1. The answer is no.

So, there is only a match of 1 here. So, then what I am going to do is the substring which gets matched is 1. It is of length 1 and this corresponds to p equal to 3. So, it corresponds to m equal to 5. So, this is m equal to 5. So, the code word corresponds to match string 1 is 1 and 5. So, we further continue this process and what we are going to do next is, we are going to now include this one into our dictionary and delete this one. So, what we are left is this. So, we deleted this string and added this one. So, the new dictionary is now given by this which is this 0 1 0 0 0 0 0 1. Now, we again start with this symbol 1. There is a match here, there is a match here. Now, 1 0 1 0, there is a match here, but here there is no match. So, there is the match only here that about 1 0 1. There is no match of 1 0 1. We can see this is 1 0 0.

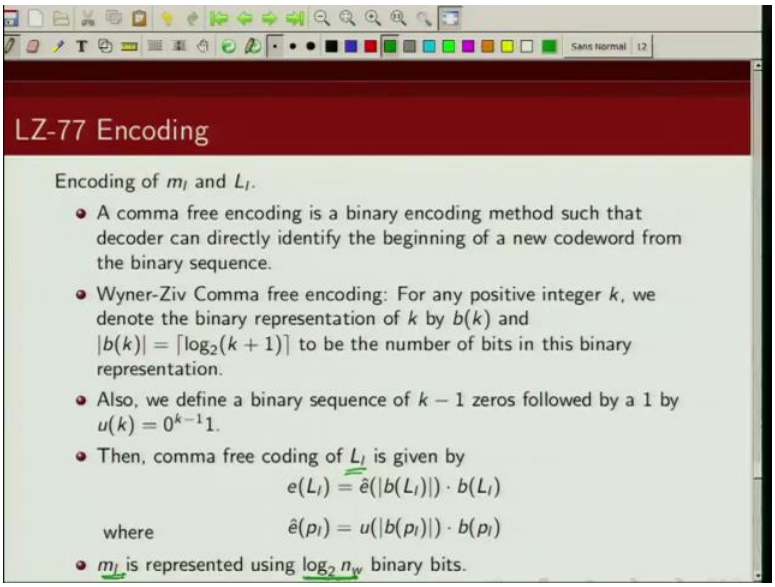
So, the largest match that we are getting is corresponding to one 0. So, this new string y 4 is then 1 0. So, that is 1 0 length 2 and the match is occurring at this location which corresponds to p equal to 2. So, then m is equal to 6, m is 6 that is this will add these two symbols to my dictionary and remove these two symbols. So, my new dictionary now is this one. This is my new dictionary. So, note that the dictionary is sliding across this sequence that you are trying to encode. So, this is my new dictionary 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0. That is my new dictionary. Now, we again follow the same procedure. Let us see the match one we just clean up little bit. So, this is my new dictionary, right is one getting map matched. There is a match of 1 here; there is a match of 1 here.

What about 1 1 1 1 is getting matched here, but there is no match of 1 at this location What about 1 1 0? You can see there is a match of 1 1 0 here. What about 1 1 0 1? Again there is a match of 1 1 0 1. What about 1 1 0 1 1? Again there is a match and what about this 1 1 0 1 1 0 1 1 0 1 1 0. So, please note the match as to start inside the dictionary, but you can go beyond. What about this 1 1 0 1 1 0 0 0? 1 1 0 1 1 0 0 0 is not there. So, the maximum match that we are getting here is corresponding to this 1 1 0 1 1 0. So, the new string y 5 is basically 1 1 0 1 1 0 and that is this and what is the length of this string. The length of this string is 6 that is this and what is the p corresponding to the match. The match happens at this location which is 1 2 3 4 5 6. So, p equal to 6 and then, m corresponds to n w which is 8 minus p. So, 8 minus 6 is 2. So, this is my m. So, corresponding to this string my code word is 6 and 2. Now, I need to add these six symbols to my dictionary and I will remove the oldest 6. I remove this, this, this, this, this. So, my new dictionary is now this. This is my new dictionary and you can check dictionary 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 0. Again look at for the maximum match. Let us just clean up little bit, ok.

So, I look for the maximum match 0 is there, a match of 0. Yes that is this; that is this; that is this. What about 0 0? There is no match of 0 0 here. I just removed because this 0 1 there is no match of 0 0 here also, but there is a match of 0 0, that is set this. So, the maximum string that gets match is 0 0. It is of length two and the match happens at p equal to 8. So, n w minus p m comes out to be 0. So, note that now we have encoded this whole string. So, the list of code word is given by this. So, this is a code word corresponding to this string. So, what we need to send to the receiver is now we need to send some representation of this, these numbers because the decoder as to understand

that length is 2 and m is 2, length is 3 and m is 0. So, we need to communicate to receiver these l i and m is length and the location where the match is happening. So, how do we do that? So, we will show that in the next slide.

(Refer Slide Time: 35:31)



So, we are going to do what we call comma free and encoding of m and l. Now, of course we would like to know if you are using variable length codes. I like to know where the boundary of l and m is. So, decode if you use the comma free code encoding decoder can be directly identify the beginning of the new code what from the sequence of numbers that you receiving. So, one such encoding scheme is given by Wyner and Ziv. So, this is as follows. So, if you have a positive number k, we denote its binary representation by b of k and number of bits require to represent the binary numbers which is basically given by this quantity is be denoted by absolute value of b k. So, let us say i have k equal to 4. So, the binary representation b of k is 1 0 0 and this term will be equal to 3 because i need three bits to represents 4. Similarly if k is 8, then b of k will be 1 0 0 0 and this term will be equal to 4.

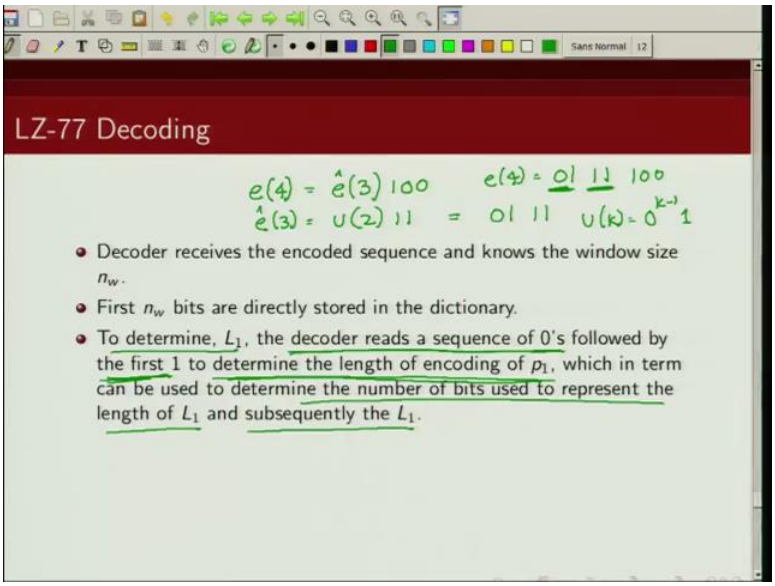
Now, we define a binary sequence of k minus 1 0 is followed by 1 which we denote by u of k. So, u of 3 is basically 0 0 1, u of 2 is 0 1 like that. So, the comma and free encoding of this length of the match of this substring, this was given by this. So, comma free encoding of l is given by e hat of number of bits required to represent the binary representation of l concatenated with this binary representation of l and this quantity is

given by u of number of bits required to represent this into binary representation of this. So, let us take an example. Let us see one length is 4. So, according to this, then e of 4 will be e hat. Now, binary representation of 4 is what? It is 1 0 0. So, how many bits we require to represent this 4 that is 3. So, this will be e hat of 3 and what is a binary representation of 4 is 1 0 0. How do we find out e hat of 3? That is nothing, but u of. Now, how do we write 3? 3 is written as 1 1 1. So, number of bits required to represent 3 is 2.

So, this will be u of 2 and binary representation of 3 which is 1 1 1, right and e of 2 is nothing, but 0 1. So, then e of 4 is given by 0 1 1 1 0 0. Now, note how we can identify 4 from here. So, from the runs of 0s we find out how many next bits we should look at. So, here we have 1 run of 0. So, we should look at next two bits. So, we look at next two bits; that is this. So, next two bit is 1 1 which is 3. Now, I look at next three bits that is a number which has been encoded. So, this number is 4. You can try out other examples as well. Let us say you want e of 3. So, this will be e hat. Now, 3 is represented using 1 1. So, this will be e hat of 2 and what is a binary representation of 3 is 1 1. Now, e hat of 2 can be written as now what is 2 2 is 1 0 and representation of 2 is one 0.

So, we require two bits to represent 1 0. So, this would be then u of 2 and 2 is 1 0. So, this is nothing, but 0 1 1 0. So, e of 3 would be 0 1 1 0 1 1 and note how do we find out 2 from here. So, run of 0 is 1 run of 0. So, we look at next two bits. What is next two bits is that is 2.

(Refer Slide Time: 41:43)



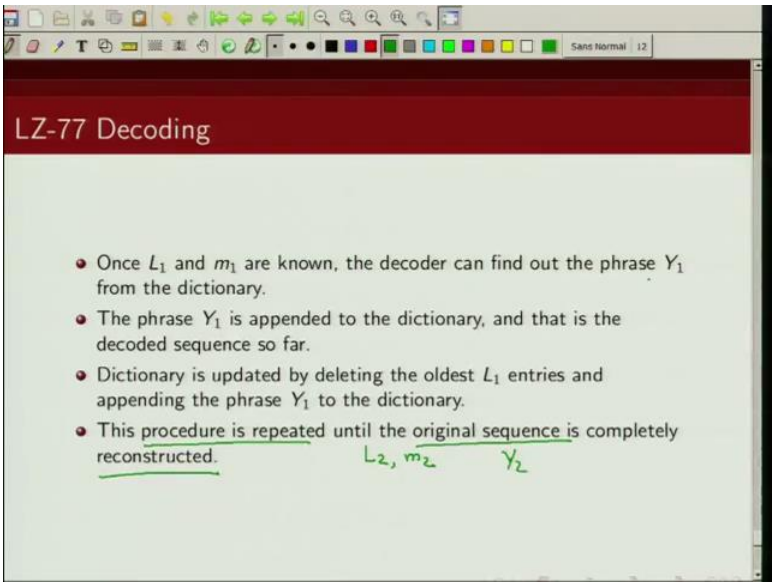
So, we look at next two bit and that is a number which has been encoded which is 3. So, this is how we are going to encode our information about l i and we could use the same thing for m i as well, but we are just right now using prefix number of bits which we are denoting by log of n w bits. Now, how does the decoding happens? So, what the decoder will do is, decoder will receive this encoded sequence which is nothing, but the initial n w bits and the encoding of l and m. So, the first thing that the decoder has to do is from the received sequence, it has to find out the values of l i and m, right. So, first n w bits are nothing, but stored for the initial bits that were encoded and then, to determine l 1 as i said the decoder will read a sequence of 0s followed by the first one to determine the length of encoding which is p 1.

Now, once it is known, what is a p 1 is those number of bits are going to be used to represent the length of l 1 and once it note the length of l 1, it will look at subsequent l 1 bits to find out what was the bit that was encoded and we just give an example to illustrate that remember we did let's say comma free coding of e 4 was comma free coding of number of bits required to represent 4 followed by binary representation of 4 and this was given by run of 1 1 binary representation of 3 is 1 1. So, this is u u of 2 followed by 1 1 and u of 2 was basically 0 1 1 1. So, e 4 was 0 1 1 1 1 0. So, that is what I am saying the decoder first reads the sequence of 1. So, it starts the sequence of one followed by the first one to determine the length of encoding of p 1. So, there is only a single run of 0s. So, the length of encoding of p 1 is going to be 2 because u of k was 0 k

minus 1 1. So, length of encoding of p 1 is in these examples is 2 that is this which is then used to determine the number of bits used represent l 1.

So, this we look at next two bits. This will tell us number of bits used to represent l and what is this? This is 3. So, we have knowledge about that next three bits are used to represent l. So, we look at next three bits and what is that? That is 1 0 0. That is the binary representation of 4 and that is how we decode that this l is 4, right. So, that is how we do it now have been determine l 1 and then, since we are using prefix number of bits to represent m. So, we know what are the next bits that we should look at in our examples that we have considered our n w is 8.

(Refer Slide Time: 45:39)



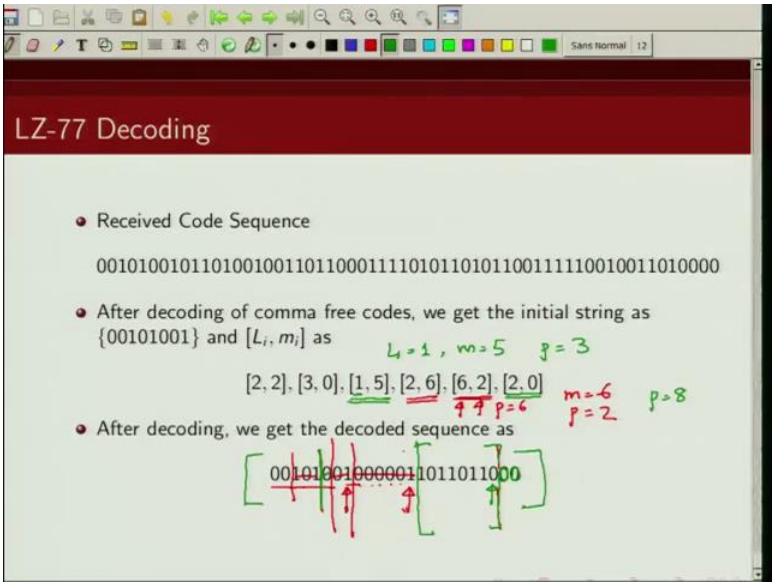
So, that means we are using next three bits to represent m, right. So, depending on the value of m from 0 to 7, we are using 0 0 0 to 1 1 1 to represent m. So, we can find out then what L 1 is and what m 1 is. Now, once L 1 and m 1 are known, then we can find out the phrase Y1 from the dictionary. Why? Because Y1 was the string that was matched to what was already there in the dictionary. The length of the match was l 1 and the location of the match was function of m.

So, if I give you Y1 and m1, you can find out what is this string that is matched with the dictionary. So, I can find out Y1. Now, once I find out Y1, the next step is to append this y on to my dictionary and remove the old L1 entries from the dictionary. So, my new dictionary will now have these Y1 string and will have the old L1 bits removed. So, this

Y1 will be appended to a dictionary. Next the dictionary is updated by deleting the oldest L1 entries and appending the phrase Y1 to the dictionary. So, that is my new dictionary. Now, I am going to have, I have my dictionary and I will again repeat this process to find out what my L2 is and what is my m2 is. Once I find out what my L2 and m2 are, I am going to find out the string Y2 which is of length L2 from my new dictionary and I am going to add this string Y2 to my dictionary and remove the oldest L2 entry.

So, now, that will be my new dictionary. Then, again I will de code what my L3 and M3 were and then I will from the dictionary I will find out the new string Y3. I am going to add this string to my dictionary and remove the oldest L3 entry. So, this procedure is repeated until the original sequence is completely reconstructed. Now, since these match happened from the dictionary, I know what exactly those strings were. See let us take an example to illustrate that.

(Refer Slide Time: 48:08)



Now, this is to receive sequence that I have received. Now, I am considering the same example. So, my n w is 8 which is known to the dictionary which is known to the encoder and decoder that the size of this hiding window decode dictionary is 8 and I am using comma free encoding for these L is and I am using prefix number of bits to denote these m is. So, log of 8 was 3. I am using three bits to denote mi and for Li, I am using comma free encoding. So, how do I decode this string? So, let us first find out. So, what does this is set of code word I have received, right. So, what does this code word consist

of? This code word consists of first eight bits are nothing, but the initial sequence that I wanted to encode and subsequently I will have the value of L1 m1, L2 m2 like that, right. So, let us first from this particular receive sequence, try to find out these code word corresponding to L1 m1, L2 m2. So, as I said the first n w is 8. So, the first eight bits are my initial sequence 1 2 3 4 5 6 7 8. So, this is my initial sequence that I transmitted.

Next is encoding for L i L1, so this run of single one. So, I should look at next two bits, this is the encoding what p 1. So, this next two bit says binary number is 2. So, the next two bits are going to be my actual value which is been encoded and what is this next two bit, that is 1 0. So, L1 is 2 next after i encode L1. I am encoding, I am using next three bits to encode m. So, what is m? M1 is 2 0 1 0 is 2 again. So, I have to find now find L2. So, there is a run of single 0. So, I should look at next two bit that is 2. So, these next two locations of the number written at next two bits will give me the number which has been encoded. So, that is 1 1. So, L2 is 3. Next three bits are used for m2. So, m2 is 0. Now, here there is 0 run of 1. So, I should look at next bit that basically says I should look at next one bit. So, that is going to tell me that is 1. So, here L3 is 1 and next three bits will give me value of m.

So, M3 is 5 is a run of 1. So, I should look at next two bit that is 2. So, those next two bits will give me the encoding of the number and this is two. So, L 4 is 2 and next three bits are corresponding to m. So, m 4 is 6. Here I look at run of 1. So, I look at next two bit that is 3. So, the next three bits will give me the value of L5 and that is 6. So, L5 is 6 and then next three bits are used for encoding of m and that is 2. So, m5 is 2 and again here there is a run of 1. So, look at next two bit that is 2. So, again look at next two bits. So, L6 comes out to be 2 and the next three bits corresponds to m 6 which is 0. So, now, what I have done is, I have actually decoded the string into initial string and L1 m1 L2 m2 L3 M3 L4 m4 L5 m5 and L6 m6. So, that is what I did. This was my L1 m1 L2 m2 L3 M3 L4 m4 L5 m5 and L6 m6 and this was my initial n w bits, right.

So, I have this. Now, what do I need to do once I know these L is m is now I need to find out what has the corresponding string Y1 corresponding to this L1 m2. What is the substring Y2 corresponding to this L2 m2. So, I need to find these substrings and what is the procedure? So, I start with the initial dictionary which is this. I look at a match of length 2 which starts at m equal to 2. So, m equal to 2 corresponds to p equal to 6. So, I look at location number 6. So, what is that 1 2 3 4 5 6. So, I look at this. So, there is a

match of length 2 at location index 6 because I missed two. So, then the match is starting at this location match of two will be 0 0. So, what I have found out is the string Y1 is 0 0.

Now, what is the next step? The next step is I need to add this string to my initial dictionary and I will get read of the oldest two entries. So, my new dictionary will be now 1 0 1 0 0 1 0 0, right. Now, I look at this. I need to find Y2. So, this is a match of length 3 and p is m is 0. So, m is 0 means p equal to 8, m is 0 means p equal to 8. So, I have a match here of length 3, right and starting at this location, now how I find match of length 3. I only have two bit here. So, clearly the match starts from this location and then, the first bit here has to be 0 because the match starting at this location, now match starting at this location is 0 0. So, the next bit has to be 0, right. So, we have found out that the string that is matched Y2 is 0 0 0. So, what do we need to do is, we append this new string to our dictionary and remove the three oldest entries. So, this we continue essentially. So, if you look at initially we had these eight bits. We found out that this was the match and our dictionary at the time was new dictionary was this. Similarly we found out that the next match Y2 was 0 0 0. So, we got rid of the next three bits and this was our new dictionary.

Now, this process is going to be continued for the remaining. So, look at this one here. L1 is 1 and m is 5. So, then p corresponds to 3. So, we look at first second third that is this. So, we look at this location and we look at match of 1 and what is the match of 1 that is 1. So, we add that one here and we get rid of this entry. So, our new dictionary now is this next we look at this. So, here m corresponds to 6. So, that corresponds to p equal to 2. So, look at this location 2 which would be. So, our dictionary here was let me clean up little bit. So, dictionary was this. Our dictionary was this 0 1 0 0 0 0 0 1. So, if you look at p equal 2, that is this right and we have a match of length 2 that is 1 0. So, we have a match of length 2 that is 1 0. We append this to your dictionary and remove the next bit. So, your new dictionary is actually this. Your dictionary is this.

Now, look at this one. L is 6 and m is 2. So, m is 2 meaning this corresponds to p is 6. So, look at the sixth entry 1 2 3 4 5 6 and look at this. So, we have to look at match of length 6. Now, first match is 1. That is this second match is 1. This third match is this. So, that is this now the forth match is 1. So, that is this fifth match is this at is this and sixth match is 0. So, this is the string that has been added. Now, my new dictionary will be and I will get rid of 6 bits. So, I will get read of 1 2 3 4 5 6. So, my new dictionary is

this one. This is my new dictionary. Now, here L is 2 and m is 0. M is 0 meaning p is equal to 8. So, I need to find the string of length two matching here. So, this is 0. So, this has to be 0 and next match has to be 0. So, this way you can see I have been able to decode this whole string again.

To recap what we did was initially we received n w bits and then, we subsequently received the coding for L i and m i. So, we first from the comma free encoded bits of L i and m i, we found out what those L1 m1 L2 m2 L3 M3 were then what we did was since we knew L1 m1 L2 m2, we knew exactly how long is the match in the dictionary and where is the location of the match in the dictionary. So, using the knowledge of L i m i, we found out the substring that is matched with what was there in the dictionary.

Now, once we match the substring, we added that substring to the dictionary and removed L i bits, the oldest L i bits from the dictionary. So, our new dictionary was this y i appended with oldest L i bits removed once we have this new dictionary and then, we moved up to next value of L1 and m i and again we repeat it the same procedure. We looked at the location where the match was and we looked at the length of the location match. From there we found out what was the string that got matched. Then, we are appended that string to the dictionary removed L i bits from the old dictionary and we progressed until we got what we decoded all the bits. So, this is how basically conceded. It is a sliding window version where the dictionary is sliding over the bits that what transmitting.

So, with this we conclude our discussion on Lempel Ziv 77 algorithm, encoding and decoding. In the next class, we will talk about Lempel Ziv Welch algorithm.

Thank you.