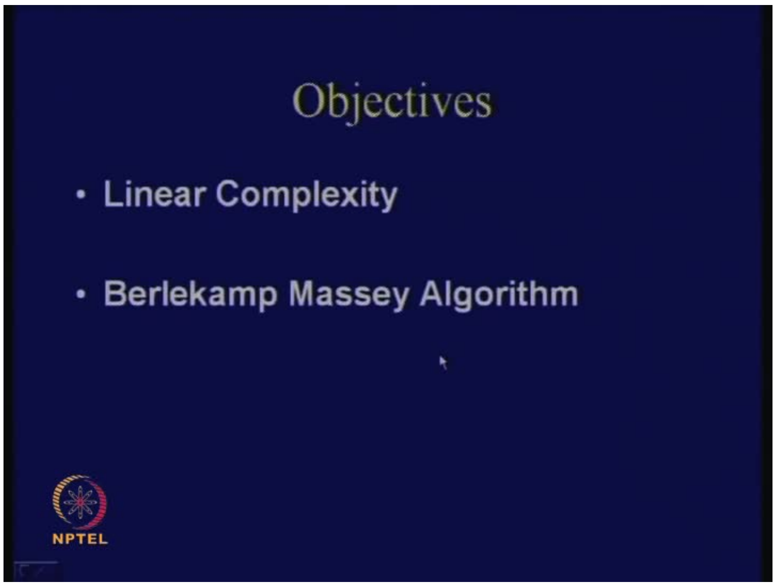
Cryptography and Network Security
Prof.D.Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Model No # 01
Lecture No # 20

Stream Ciphers

In today's class, we will continue with stream ciphers. Yesterday, you have seen some small examples to understand that stream. The problem was I was interested in reconstructing the stream cipher that is given in the sequence. Can I again reconstruct the corresponding LFSR? In today's class, we can essentially talk about two topics - we will discuss about linear complexity, I think I started with this and I am continue with this, and mainly will concentrate on this algorithm, that is, Berlekamp Massey algorithm, which is used to solve this problem. It is a very phenomenal and seminar paper. So, we will discuss on this. And it is quite important.

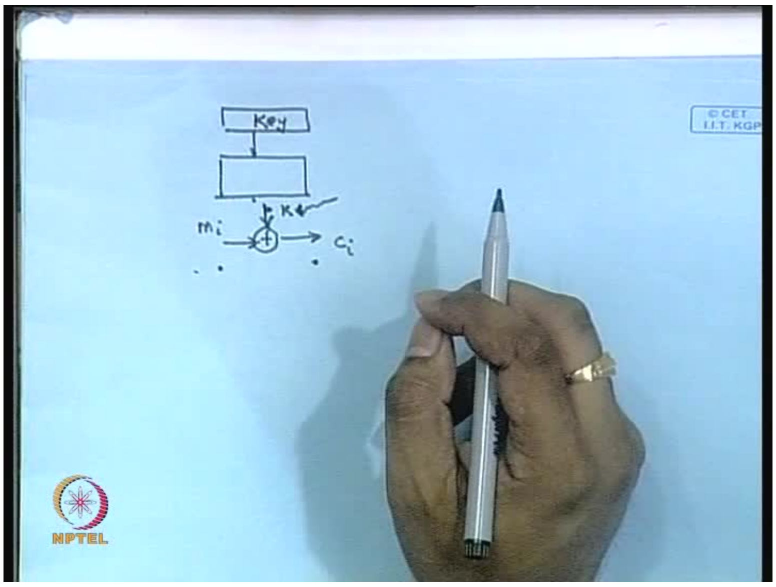
(Refer Slide Time: 00:43)



Objectives

- Linear Complexity
- Berlekamp Massey Algorithm

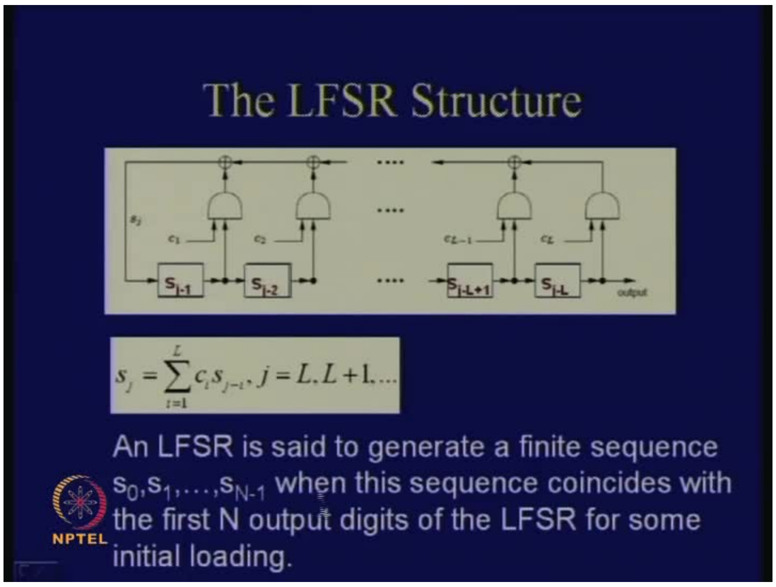
(Refer Slide Time: 01:04)



I hope you understand why we are studying this? Because consider a very simple stream cipher. Essentially have got a key stream k, for example, and suppose you are just taking M_i and obtaining the corresponding cipher text C_i. In context models, like for example, chosen plaintext or non-plaintext models, where the corresponding value of M_i and the corresponding value of C_i, you know the value of K. But the point is that, given this value of K, can you also know the corresponding initial secret key with which it is started. This is the key stream, not the actual key. So, there is an internal algorithm here and there is an input secret key, which is been provided as an input. Right? There is a secret k or key, which is the actual secret of the cipher, therefore, the observer is essentially observing at this point. From there, he is interested in constructing the LFSR and also the key.

We are considering a single LFSR system and trying to understand that whether a single LFSR system, which has got nice pseudorandom properties, can be reconstructed using Berlekamp Massey algorithm. If you remember the problem, I can essentially decompose or rather write all these key bits as linear combination of the internal secrets. How many secret values are there? For N length LFSR, there are actually two N secrets. You do not know the initial seat that gives you N bits. You also do not know the corresponding connection polynomial. Another thing which you do not know is the length of the LFSR. Therefore, you will find that can you actually write them as the system of linear equations and solve them. But this is an unwilling process.

(Refer Slide Time: 03:13)



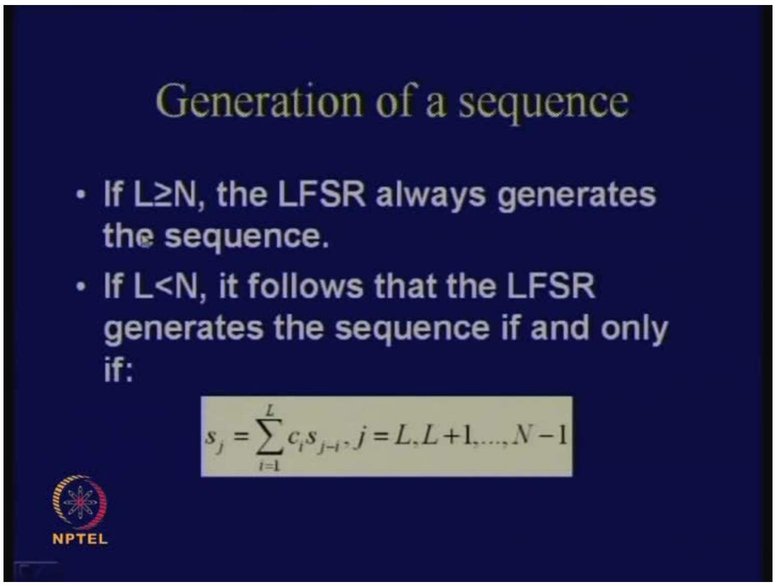
The LFSR Structure

$$s_j = \sum_{i=1}^{L} c_i s_{j-i},\ j = L, L+1, \ldots$$

An LFSR is said to generate a finite sequence $s_0, s_1, \ldots, s_{N-1}$ when this sequence coincides with the first N output digits of the LFSR for some initial loading.

Berlekamp Massey algorithm gives a very simple and elegant technique in order to solve this problem. With this motivation, we will study this problem. First of all, this is a recap of at the LFSR structure, so you see that we have got S_j minus 1 to S_j minus L. Please pay attention because there is certain amount of formalism required to understand the basic principle of the algorithm. There is some amount of inverse mathematics. Therefore, S_j minus 1, S_j minus 2 to S_j minus L plus 1 and S_j minus L. How many flops are there? There are L flip flops or L storage element. So, the corresponding feedback S_j you get as a linear combination of these points. But whether you are taking the feedback or not depends upon a control bit. There are also L control bits here. You take this and multiply with this corresponding feedback and you exhort that with the previous thing.

(Refer Slide Time: 04:26)



Generation of a sequence

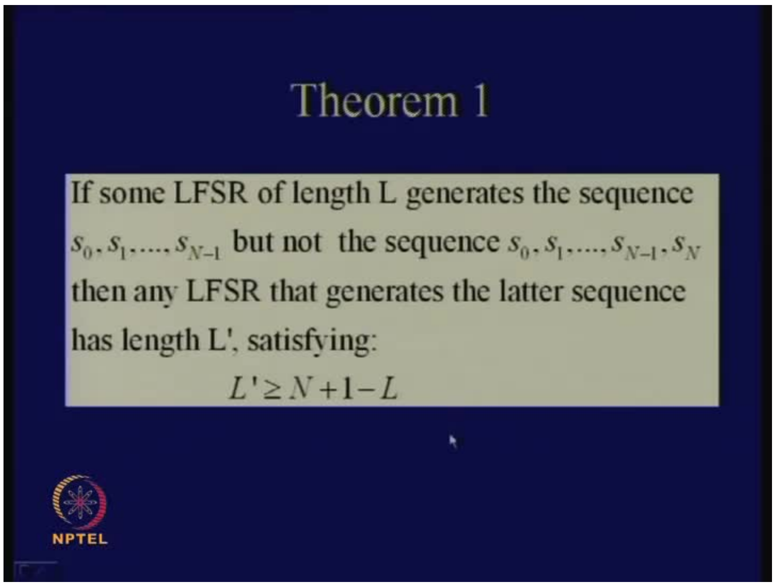
- If L≥N, the LFSR always generates the sequence.
- If L<N, it follows that the LFSR generates the sequence if and only if:

$$s_j = \sum_{i=1}^{L} c_i s_{j-i}, \; j = L, L+1, \ldots, N-1$$

They were simple structure of the LFSR. The LFSR was said to generate a finite sequence, say S_0 S_1 to S_{N-1}. When this sequence coincides with the first N output digits of the LFSR, then the sequence is being generated by the corresponding LFSR. Now, you note that if L is greater than or equal to N, that is, if the length is quite larger compared to the length or the sequence that you want to generate, then generating is trivial. You can simply streaming the data and you can obtain the corresponding output. But the problem becomes when L is smaller than N, that is, the sequence is a larger sequence. We will be concentrating on this part of the problem, that is, when L is smaller than N. It follows that LFSR generates the sequence, if and only if this is true. Why I have started from j equal to L?

So you know what to read out this particular equation. It says that S_j is equal to the sigma of C_i multiplied with S_j minus I, and your j starts from L. why? Because before that you have just streaming the data, therefore, the feedback effect does not come before this. j equal to L point. The problem starts when only when j is equal to L, and after that, this has to be a linear combination of the previous L values. This is a very important equation which we have to keep in mind, that is, S_j is equal to sigma i equal to one to L C_i S of j minus I. To remember, it is just the coefficients multiplied with the previous L values of the LFSR.

(Refer Slide Time: 06:04)



Theorem 1

If some LFSR of length L generates the sequence $s_0, s_1, \ldots, s_{N-1}$ but not the sequence $s_0, s_1, \ldots, s_{N-1}, s_N$ then any LFSR that generates the latter sequence has length L', satisfying:

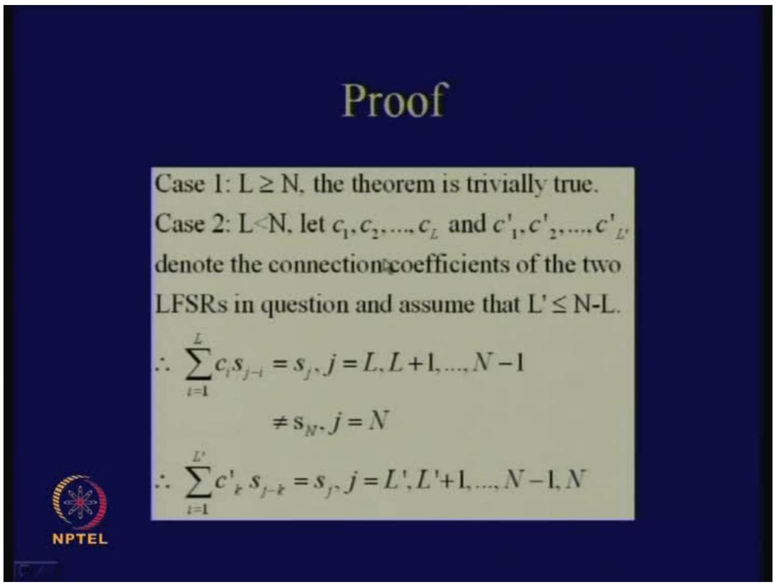
$$L' \geq N + 1 - L$$

First of all, let us consider this theorem. It says that if some LFSR of length L generates the sequence S_0 to S_{N-1}, but not the sequence S_0 to S_{N-1} and S_N, I have added one more value to the sequence, then any LFSR that generates the later sequence, if it has got length of L dash, then you can actually prove that L dash is greater than equal to N plus 1 minus L. If you remember the last day, we were doing a simple example, where we saw that till some point, we are easily able to construct the LFSR. But the point starts when you are going beyond that. For example, if I start with 0, if you remember the previous day examples we started with 0 0, you can do with the 0 length LFSR. So, when you are doing from 0 0 1, then immediately your length jumps to three. Then you require a three stage LFSR. This is exactly the same thing. This was a trivial example where you see that in the particular example your value of N is actually equal to…, N plus one is actually equal to 3. Because N was equal to 2 and L was equal to 0. So, although I have said you that L dash is greater than equal to N plus 1 minus L… First of all, let us try to prove this result. That is, L dash is greater than equal to N plus one minus L. In order to prove you observed that one thing from this theorem statement itself, that is, N minus L plus 1. So, you can write this as N minus L plus 1. What happens when N is smaller than l?

This is actually trivially true. Then, length has to be greater than equal to 0. But the problem will be essentially therefore considering when N is actually greater than equal to L. When N is greater than equal to L, it is the interesting problem. In that case, first of all, we will start with contradiction. Let us contradict this statement. If I contradict this statement, that means, L dash greater than equal to N plus 1 minus L is contradictory.

Therefore I am assuming that L dash be less than N plus 1 minus L. Therefore L dash is less than equal to N minus L, so I can write this as L dash is less than equal to N minus L.

(Refer Slide Time: 08:54)



Proof

Case 1: L ≥ N, the theorem is trivially true.

Case 2: L<N, let $c_1, c_2, \ldots, c_L$ and $c'_1, c'_2, \ldots, c'_{L'}$ denote the connection coefficients of the two LFSRs in question and assume that L' ≤ N-L.

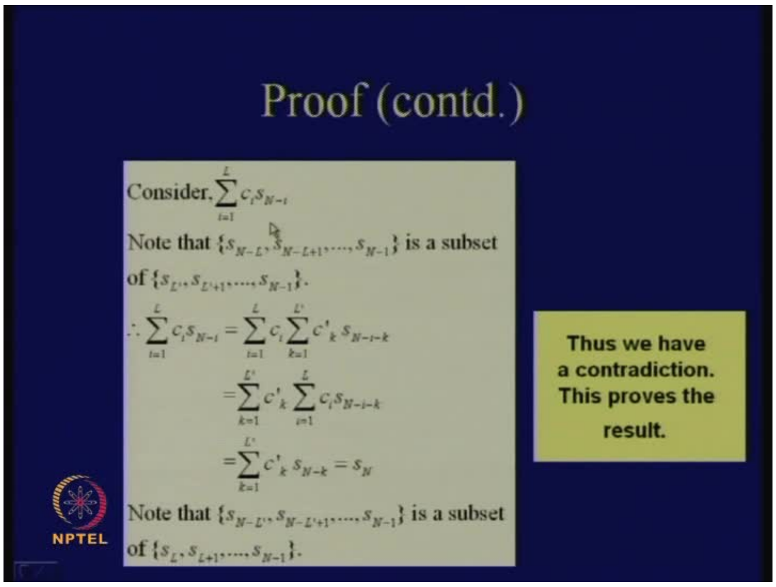
$$\therefore \sum_{i=1}^{L} c_i s_{j-i} = s_j, j = L, L+1, \ldots, N-1$$

$$\neq s_N, j = N$$

$$\therefore \sum_{k=1}^{L'} c'_k s_{j-k} = s_j, j = L', L'+1, \ldots, N-1, N$$

This is by contradiction. We have just contradicted the statement of the theorem. And what we will see that if this is true then we end up with something which is wrong, that is, we are violating the initial starting point. What are the starting points? Let us see this. This is the case 1, that is, L greater than equal to N. The theorem is trivially true for case 2. Let us consider two LFSR's. The two LFSR's has got, say for example, C_1 to C_L and $C_{1'}$ dash to C_L dash. These are the corresponding coefficients of the two LFSR's. We have assumed that L dash is less than equal to N minus L, that is, by contradicting the theorem statement. This is cleared that in this corresponding polynomial LFSR, the first LFSR has generated the sequence till N minus one. Therefore, I can write this equality but this is not equal to S_N. Why? Because it is not generating the corresponding S_N value. That is the N plus one th value in the sequence. But this LFSR is generating all the values, therefore, you start from L dash and continue till N. This equality holds, therefore, these two equations are clear to us. This is a simple statement of where we have started with, that is, the initial theorem statement basically. This LFSR is generating till N minus 1 values but this one is generating till the nth point, that is, nth value. But this one is not generating the nth value, therefore, this inequality. Is it okay?

(Refer Slide Time: 10:58)



Proof (contd.)

Consider, $\sum_{i=1}^{L} c_i s_{N-i}$

Note that $\{s_{N-L}, s_{N-L+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_{L'}, s_{L'+1}, \ldots, s_{N-1}\}$.

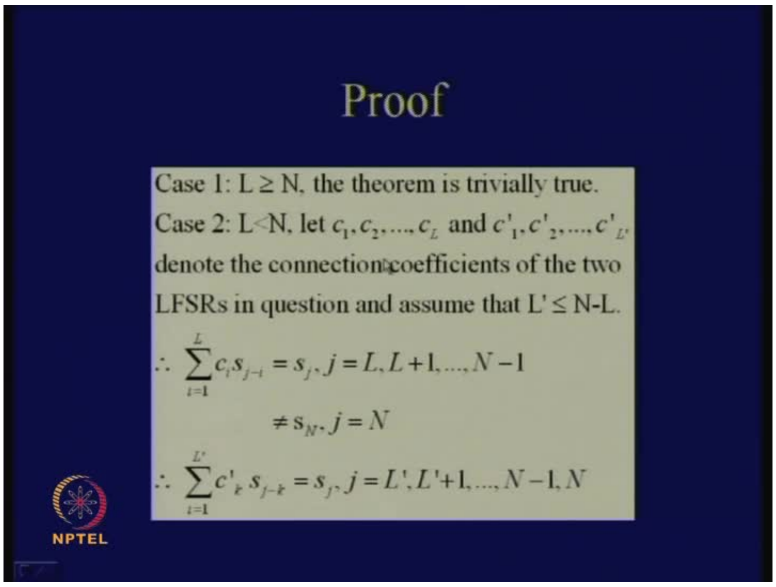
$$\therefore \sum_{i=1}^{L} c_i s_{N-i} = \sum_{i=1}^{L} c_i \sum_{k=1}^{L'} c'_k s_{N-i-k}$$
$$= \sum_{k=1}^{L'} c'_k \sum_{i=1}^{L} c_i s_{N-i-k}$$
$$= \sum_{k=1}^{L'} c'_k s_{N-k} = s_N$$

Note that $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_L, s_{L+1}, \ldots, s_{N-1}\}$.

Thus we have a contradiction. This proves the result.

Yes. Now, let us start considering this particular value.

(Refer Slide Time: 11:15)



Proof

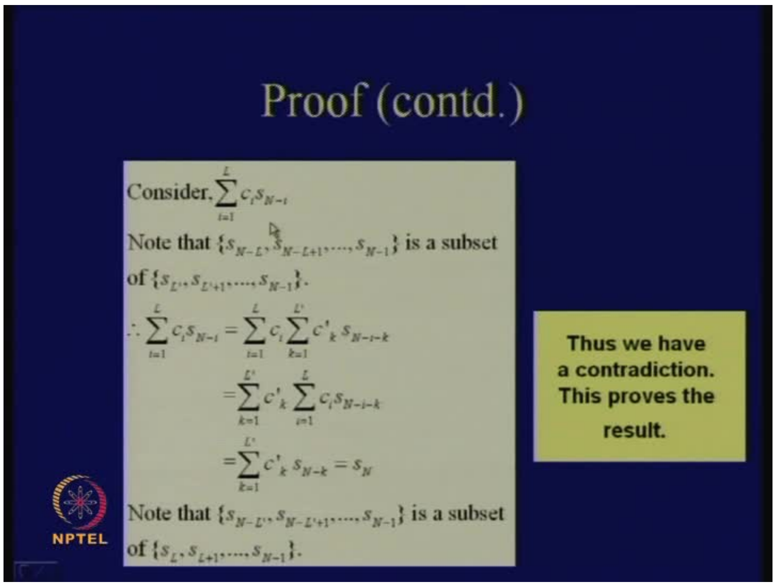
Case 1: L ≥ N, the theorem is trivially true.

Case 2: L<N, let $c_1, c_2, \ldots, c_L$ and $c'_1, c'_2, \ldots, c'_{L'}$ denote the connection coefficients of the two LFSRs in question and assume that L' ≤ N-L.

$$\therefore \sum_{i=1}^{L} c_i s_{j-i} = s_j, j = L, L+1, \ldots, N-1$$
$$\neq s_N, j = N$$
$$\therefore \sum_{i=1}^{L'} c'_k s_{j-k} = s_j, j = L', L'+1, \ldots, N-1, N$$

What we will do is essentially this -- you start with first LFSR coefficients, that is, consider sigma C$_i$ S$_{N-1}$. Therefore, I am calculating this particular value when j is equal to N. I am trying to evaluate this and I will show that if this is true, then this is actually equal to S$_N$. Therefore that contradicts my initial starting point. You understand the idea? So, how do I start to prove this? You can see that sigma if I just write sigma i equal to 1 to L C$_1$ S$_{N-i}$,

(Refer Slide Time: 12:23)



Proof (contd.)

Consider, $\sum_{i=1}^{L} c_i s_{N-i}$

Note that $\{s_{N-L}, s_{N-L+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_{L'}, s_{L'+1}, \ldots, s_{N-1}\}$.

$$\therefore \sum_{i=1}^{L} c_i s_{N-i} = \sum_{i=1}^{L} c_i \sum_{k=1}^{L'} c'_k s_{N-i-k}$$

$$= \sum_{k=1}^{L'} c'_k \sum_{i=1}^{L} c_i s_{N-i-k}$$

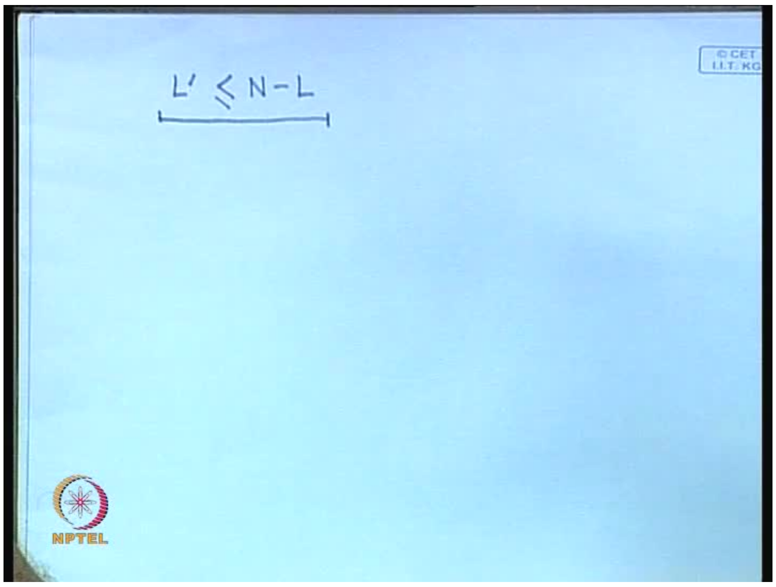
$$= \sum_{k=1}^{L'} c'_k s_{N-k} = s_N$$

Note that $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_L, s_{L+1}, \ldots, s_{N-1}\}$.

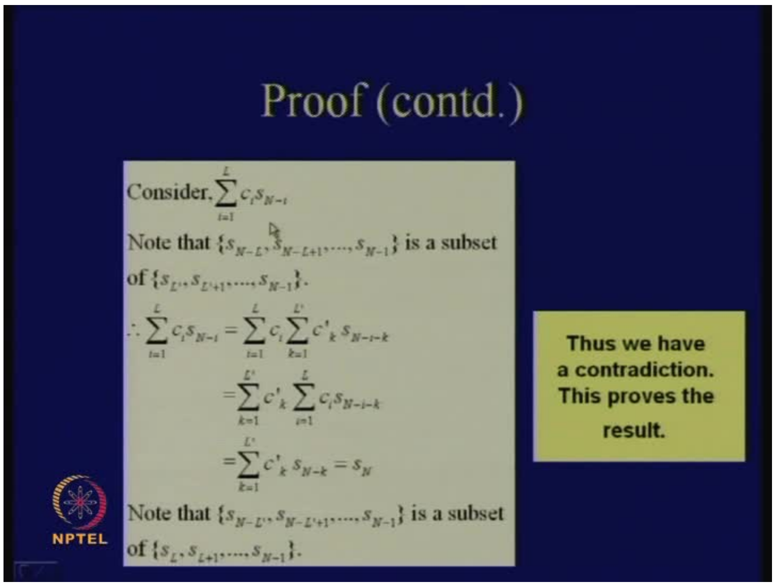
Thus we have a contradiction. This proves the result.

We can actually substitute this value. Why can we substitute this value? Do you understand this? Because in order to substitute, you note one thing that this has to lie in the range of j, otherwise this definition does not hold right. Therefore, you see that here you consider this particular sequence. This says, the sigma i equal to1 to L C_i of S_{N-I} and we note one thing that what are the two extreme ends of this sequence? In that case, in this corresponding sigma value, it starts with S_{N-1} and continues till S_{N-L}. We have assumed that N minus L is greater than equal to L dash.

(Refer Slide Time: 12:54)



(Refer Slide Time: 13:16)



Proof (contd.)

Consider, $\sum_{i=1}^{L} c_i s_{N-i}$

Note that $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_{L'}, s_{L'+1}, \ldots, s_{N-1}\}$.

$$\therefore \sum_{i=1}^{L} c_i s_{N-i} = \sum_{i=1}^{L} c_i \sum_{k=1}^{L'} c'_k s_{N-i-k}$$
$$= \sum_{k=1}^{L'} c'_k \sum_{i=1}^{L} c_i s_{N-i-k}$$
$$= \sum_{k=1}^{L'} c'_k s_{N-k} = s_N$$

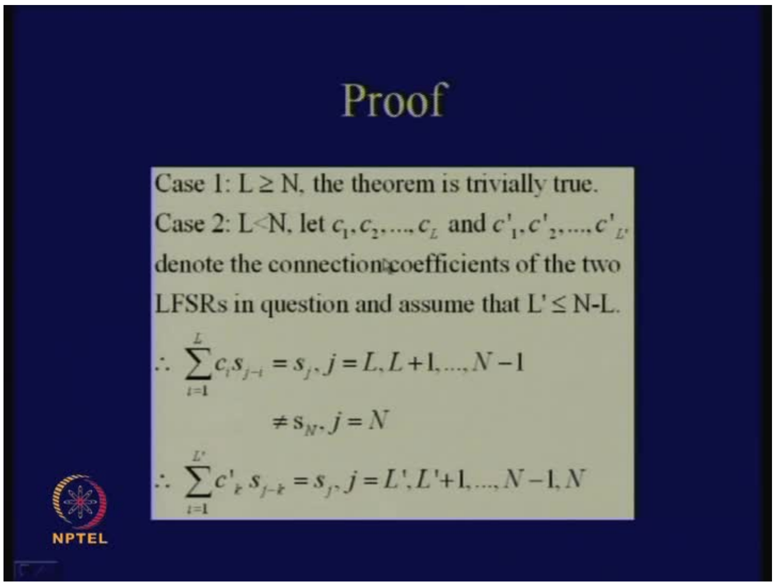
Note that $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_L, s_{L+1}, \ldots, s_{N-1}\}$.

Thus we have a contradiction. This proves the result.

N minus L is actually greater than equal to L dash, therefore, this particular sequence – S_{N-1} to S_{N-L} is actually a subset of S_L dash to S_{L-1}. Why? Because L dash is smaller than N minus L. This is essentially a subset of this particular sequence.

If I consider this sequence from S_L dash to S_{N-1}, then L dash, because of contradicted L dash, was less than equal to N minus L. Therefore, this is actually a smaller value as compared to S_{N-L}. Therefore, this is a bigger sequence as compared to this.

(Refer Slide Time: 14:36)



Proof

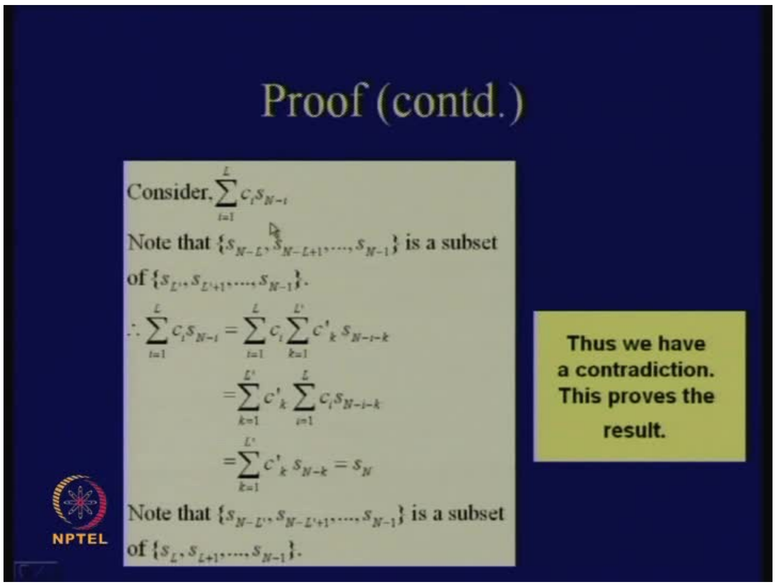
Case 1: $L \geq N$, the theorem is trivially true.

Case 2: $L<N$, let $c_1, c_2, \ldots, c_L$ and $c'_1, c'_2, \ldots, c'_{L'}$ denote the connection coefficients of the two LFSRs in question and assume that $L' \leq N-L$.

$$\therefore \sum_{i=1}^{L} c_i s_{j-i} = s_j, j = L, L+1, \ldots, N-1$$
$$\neq s_N, j = N$$
$$\therefore \sum_{i=1}^{L'} c'_k s_{j-k} = s_j, j = L', L'+1, \ldots, N-1, N$$

Therefore, this is a sub part of this. For all the values of S_{N-i}, where i runs from 1 to L, I can substitute the second equation. I can write them in terms of the coefficients of the second LFSR. Therefore that is exactly what I do -- I take this S_{N-i} and substitute by this value. You see that this is equal to sigma, k equal to one to L dash C_K dash S_{N-i-k}. Therefore, this follows straight away from this equation, so you see that if I am interested in computing S_{N-i}, then I will substitute here instead of j, I will write here N minus i. Hence, this becomes equal to sigma C_k dash S_{N-i-k}. Therefore, that is exactly you can substitute here.

(Refer Slide Time: 14:59)



Proof (contd.)

Consider, $\sum_{i=1}^{L} c_i s_{N-i}$

Note that $\{s_{N-L'}, s_{N-L+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_{L'}, s_{L'+1}, \ldots, s_{N-1}\}$.

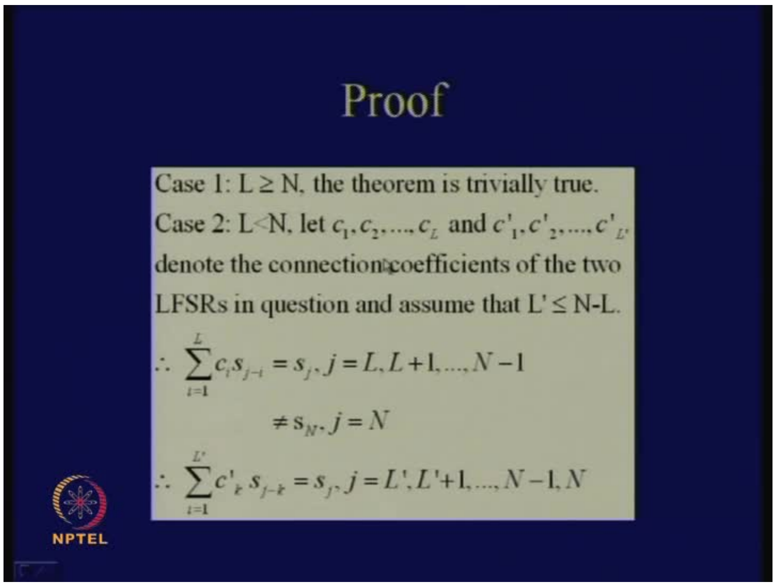
$$\therefore \sum_{i=1}^{L} c_i s_{N-i} = \sum_{i=1}^{L} c_i \sum_{k=1}^{L'} c'_k s_{N-i-k}$$
$$= \sum_{k=1}^{L'} c'_k \sum_{i=1}^{L} c_i s_{N-i-k}$$
$$= \sum_{k=1}^{L'} c'_k s_{N-k} = s_N$$

Note that $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_L, s_{L+1}, \ldots, s_{N-1}\}$.

Thus we have a contradiction. This proves the result.

It becomes equal to C_k dash S_{N-1-k}. Till this part, is it clear? Now, you can interchange these two sigma values. So, you can bring this one here and bring this one here and again you have got this. If you consider the next sigma, it is C_i S_{N-1-k}. Here, I can use this particular equation and instead of this, I can write S_{N-k}. Note that you can again write this because S_{N-L} dash till S_{N-1} is a subset of S_L to S_{N-1} as L dash is less than equal to N minus L. Therefore, if you write this… Now, you see that what you have essentially obtained here is nothing but equal to S_N.

(Refer Slide Time: 16:06)



Proof

Case 1: L ≥ N, the theorem is trivially true.

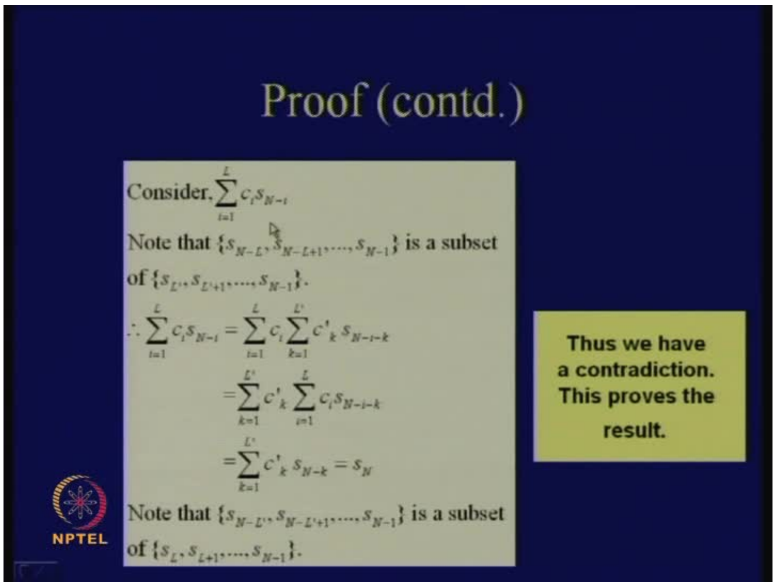
Case 2: L<N, let $c_1, c_2, \ldots, c_L$ and $c'_1, c'_2, \ldots, c'_{L'}$ denote the connection coefficients of the two LFSRs in question and assume that L' ≤ N-L.

$$\therefore \sum_{i=1}^{L} c_i s_{j-i} = s_j, j = L, L+1, \ldots, N-1$$

$$\neq s_N, j = N$$

$$\therefore \sum_{i=1}^{L'} c'_k s_{j-k} = s_j, j = L', L'+1, \ldots, N-1, N$$

(Refer Slide Time: 16:40)



Proof (contd.)

Consider, $\sum_{i=1}^{L} c_i s_{N-i}$

Note that $\{s_{N-L}, s_{N-L+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_{L'}, s_{L'+1}, \ldots, s_{N-1}\}$.

$$\therefore \sum_{i=1}^{L} c_i s_{N-i} = \sum_{i=1}^{L} c_i \sum_{k=1}^{L'} c'_k s_{N-i-k}$$

$$= \sum_{k=1}^{L'} c'_k \sum_{i=1}^{L} c_i s_{N-i-k}$$

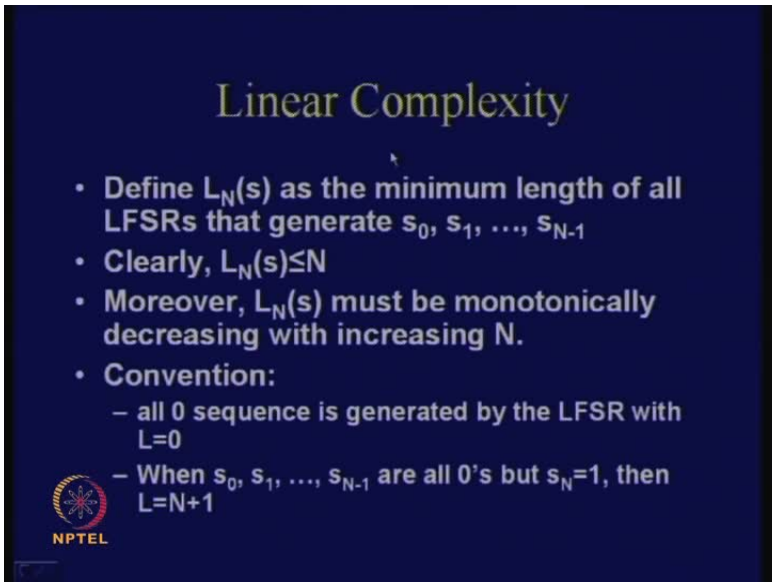
$$= \sum_{k=1}^{L'} c'_k s_{N-k} = s_N$$

Note that $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ is a subset of $\{s_L, s_{L+1}, \ldots, s_{N-1}\}$.

Thus we have a contradiction. This proves the result.

Therefore, what you had essentially contradicting is the initial starting point, that is, the first LFSR cannot generate the S_N digit. Where did we go wrong? We essentially assumed this, which was wrong. Therefore, this proves the theorem, that is, L dash is actually not less than equal to N minus L but it is actually greater than equal to N minus L plus 1 or N plus 1 minus L. Did you understand the principle? I mean, you can always go back and look at the proof in details but this is the idea that if there is a particular LFSR, which is unable to generate sequences from S_0 to S_N, we generate the sequence S_0 to S_{N-1}.

(Refer Slide Time: 17:19)



Linear Complexity

- Define $L_N(s)$ as the minimum length of all LFSRs that generate $s_0, s_1, \ldots, s_{N-1}$
- Clearly, $L_N(s) \leq N$
- Moreover, $L_N(s)$ must be monotonically decreasing with increasing N.
- Convention:
  - all 0 sequence is generated by the LFSR with L=0
  - When $s_0, s_1, \ldots, s_{N-1}$ are all 0's but $s_N=1$, then L=N+1

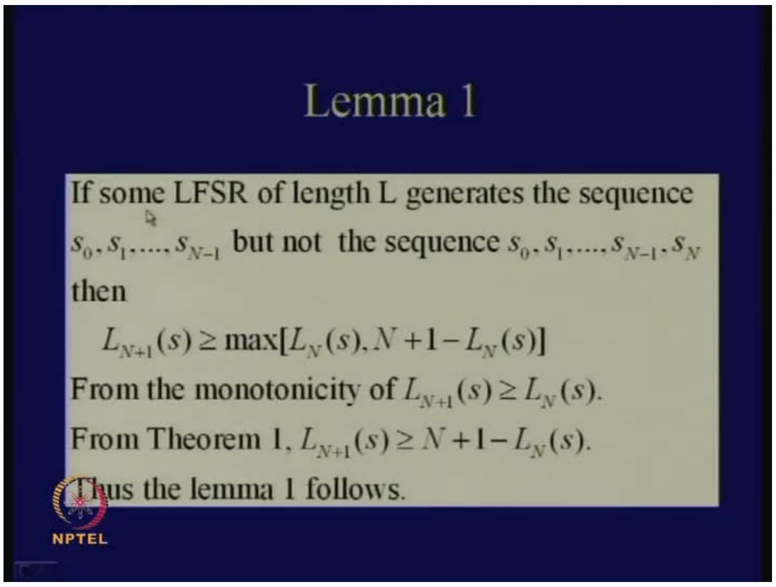
Then you need to add on to that length. Therefore, if you add on to that length and the length becomes L dash, where the previous length was L, then there is an definite relation between L dash and L, and that is what we have proved in this theorem. Now, you can actually better understand the linear complexity problem. What is the linear complexity problem? This is what I defined -- this is the minimum length of all the LFSR's from which we generated the sequence S_0 to S_{N-1}. So, clearly you can see that $L_N(S)$ will be less than equal to N. Why?

(( ))

Yes. Therefore, if I am considering S_0 to S_{N-1}, any N bit LFSR can definitely generate it.

The problem is can I obtain lesser than that? Another thing you note that moreover $L_N(S)$ must be monotonically decreasing. Actually this only monotonically non-decreasing, so there is a mistake. It only monotonically non-decreasing with increasing value of N. Moreover, $L_N(S)$ must be actually monotonically non-decreasing with increasing N. Why it is true? Because you will straight away contradict the initial hypothesis. So, we will start with certain conventions. The conventions are that the all 0 sequence is generated by the LFSR, whose length is 0. We are trying to develop a recursive contraction. There are always some initial starting points for any recursive algorithm. These are the starting points, that is, all 0 sequence is generated by the LFSR, whose length is L equal to 0. And if, S_0 to S_{N-1} are all 0 and your S_N is equal to 1, the length which is required is actually equal to N plus 1.

(Refer Slide Time: 19:33)



Lemma 1

If some LFSR of length L generates the sequence $s_0, s_1, \ldots, s_{N-1}$ but not the sequence $s_0, s_1, \ldots, s_{N-1}, s_N$ then

$$L_{N+1}(s) \geq \max[L_N(s), N+1-L_N(s)]$$

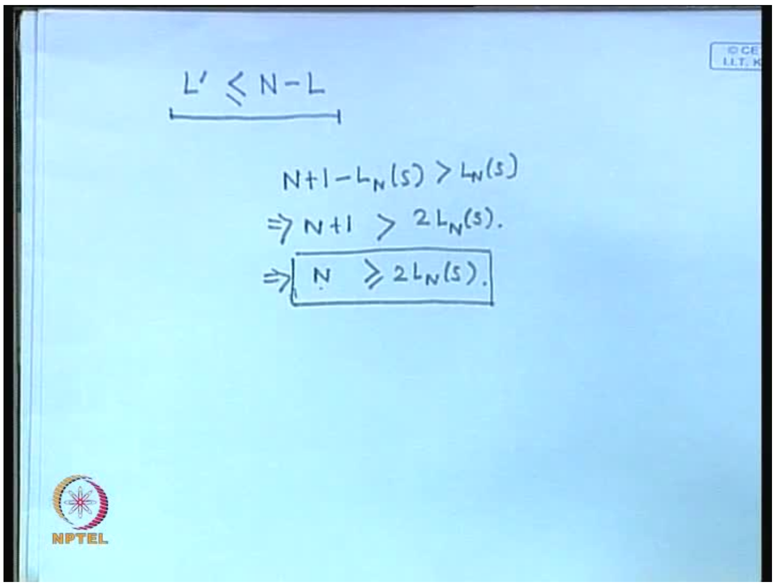
From the monotonicity of $L_{N+1}(s) \geq L_N(s)$.

From Theorem 1, $L_{N+1}(s) \geq N+1-L_N(s)$.

Thus the lemma 1 follows.

This we saw for our example. If you remember 0 0 1, 0 0 could have been generated by a 0 length LFSR but 0 0 1 was generated by a three-stage LFSR. Therefore, this is what it says exactly and you can actually… These are the conventions. Now let us consider another lemma. So, lemma is if some LFSR of length L generates the sequence S_0 to S_{N-1} but not the sequence S_0 to S_{N-1} S_N, that is essentially you see that this S_N is not being generated, then you can actually show that a linear complexity for N plus one for this sequence is actually greater than max of $L_N(S)$ or N plus 1 minus $L_N(S)$. This is actually quite trivial. This follows from the previous idea that we know it is monotonic, and therefore, L_N $L_{N+1}(S)$ has to be greater than equal to $L_N(S)$. We also prove this result. So, it has to be greater than the maximum of these two. Therefore, it is either greater than equal to the maximum of this or whichever is the maximum value, it has to be greater than that. So $L_{N+1}(S)$ is has to be actually greater than equal to maximum of $L_N(S)$, N plus 1 minus $L_N(S)$. This is an interesting lemma, which will help us to recursively compute the linear complexity.

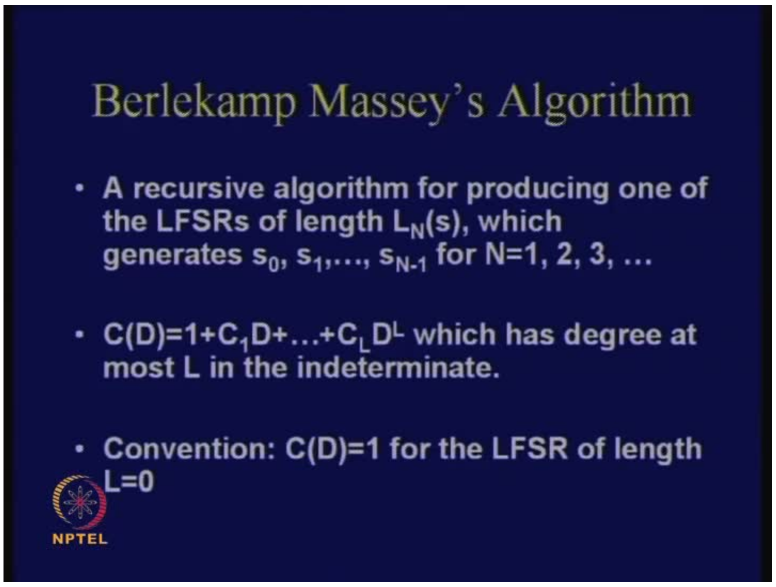
(Refer Slide Time: 21:24)


$$\underbrace{L' \leq N - L}$$

$$N+1-L_N(s) > L_N(s)$$
$$\Rightarrow N+1 > 2L_N(s).$$
$$\Rightarrow \boxed{N \geq 2L_N(s).}$$

So, you note one thing that when do you require an updating, I mean if N plus 1 minus $L_N(S)$ is actually greater than $L_N(S)$, then you required to add on to the length. And when this happen, it means that N plus 1 is greater than 2 $L_N(S)$. So, I can write this as N is greater than equal to 2 $L_N(S)$. For rest of the cases, updating is not required. That is, when it is less than 2 $L_N(S)$, then updating is actually not necessary.

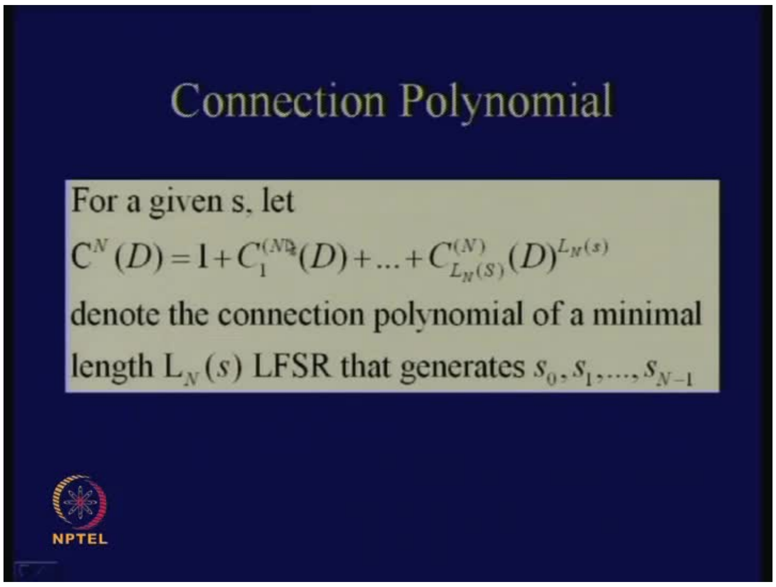
(Refer Slide Time: 22:30)


Berlekamp Massey's Algorithm

- A recursive algorithm for producing one of the LFSRs of length $L_N(s)$, which generates $s_0, s_1, \ldots, s_{N-1}$ for N=1, 2, 3, …
- $C(D)=1+C_1D+\ldots+C_LD^L$ which has degree at most L in the indeterminate.
- Convention: C(D)=1 for the LFSR of length L=0

So, the length gets updated only, depending upon certain cases. But whenever there is an update, the update will happen only when N is actually greater than equal to 2 $L_N(S)$. Do you understand why? Because of the monotonicity again. Yeah. Now, we are actually more or less trying to understand the Berlekamp Massey algorithm. It is a recursive algorithm which produces one of the LFSR's of length $L_N(S)$, that is, minimum length. $L_N(S)$ is minimum length, which generates the sequence. We generate the sequence S_0 S_1 to S_{N-1} for any integer value of N. For this again, I let us look back at the connection polynomial.

(Refer Slide Time: 23:35)



Connection Polynomial

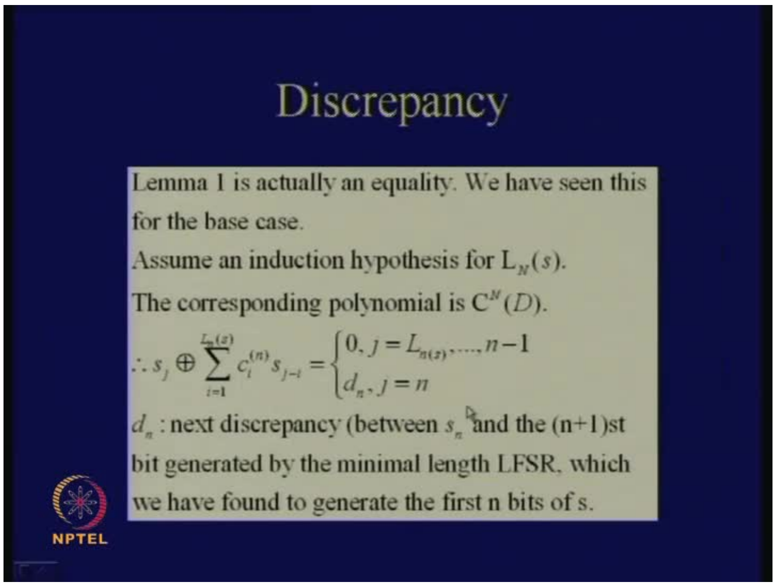
For a given s, let

$C^N(D) = 1 + C_1^{(N)}(D) + \dots + C_{L_N(S)}^{(N)}(D)^{L_N(s)}$

denote the connection polynomial of a minimal length $L_N(s)$ LFSR that generates $s_0, s_1, \dots, s_{N-1}$

So, you had C D equal to one plus C_1 D plus, and so on till C_L D power L. This was my connection polynomial, which has degree at most L in the indeterminate. The indeterminate in this case is D, in the variable D. So, the convention is that second areas starting convention follows from the previous thing, that is, C D is equal to 1 for the LFSR of length L equal to 0. If length L is equal to 0, then we will assume that C D is equal to one. This is just convention. Therefore, for a given sequence S, I will write that C and D is equal to 1 plus C_1 N D plus, and so on, till $C_{LN(S)}$ N D to the power of $L_N(S)$. This is just a rewriting of the previous connection polynomial. Only thing is that I have specified that the length is actually equal to $L_N(S)$, and all these sequences if you note carefully are actually denoted like C_1 C_2 and C 3 so on till $C_{LN(S)}$. But there at the top, I have written mean values to indicate that this is a generated of the sequence till S_{N-1}. That is from S_0 to S_N S_{N-1}. This LFSR essentially generates the sequence S_0 S_1, and so on, till S_{N-1}. So, what we are interested in calculating in that case? C_{N+1}D. Why we are interested in calculating the value of C_{N+1} D because C_{N+1} D will give us the corresponding coefficients, which will generate the sequence S_0 S_1 to S_N. Now, you see that…

(Refer Slide Time: 24:55)



Discrepancy

Lemma 1 is actually an equality. We have seen this for the base case.

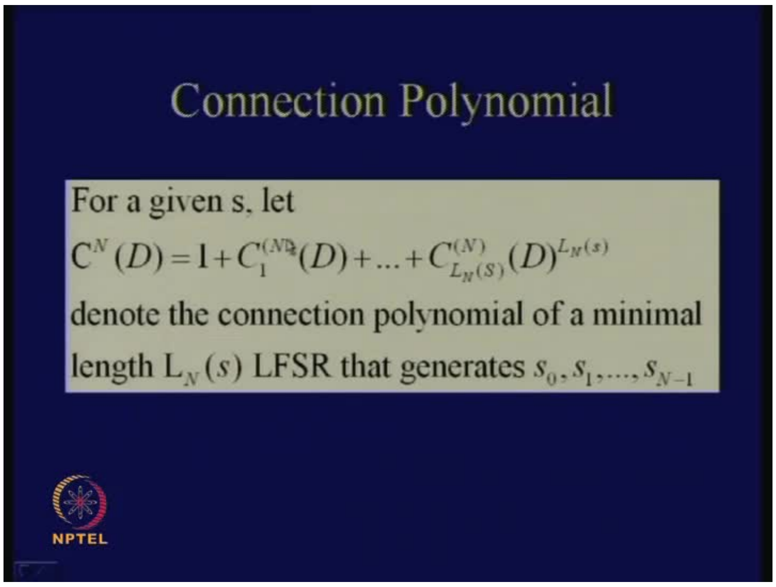
Assume an induction hypothesis for $L_N(s)$.

The corresponding polynomial is $C^N(D)$.

$$\therefore s_j \oplus \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} = \begin{cases} 0, j = L_{n(s)}, \ldots, n-1 \\ d_n, j = n \end{cases}$$

d_n : next discrepancy (between s_n and the (n+1)st bit generated by the minimal length LFSR, which we have found to generate the first n bits of s.

(Refer Slide Time: 25:00)



Connection Polynomial

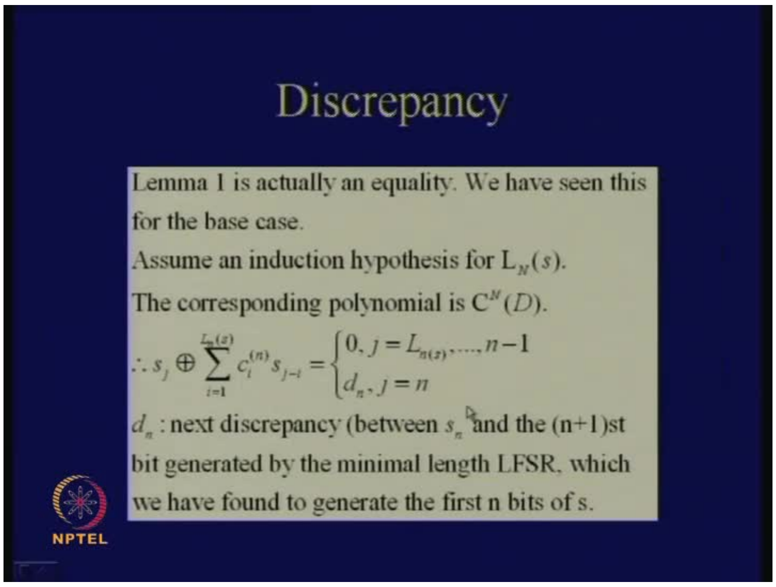
For a given s, let

$$C^N(D) = 1 + C_1^{(N)}(D) + \ldots + C_{L_N(S)}^{(N)}(D)^{L_N(s)}$$

denote the connection polynomial of a minimal length $L_N(s)$ LFSR that generates $s_0, s_1, \ldots, s_{N-1}$

Before going into the next thing, we will try to understand certain things. That is, we will try to prove a recursive constriction of this particular polynomial, that is, C_N D, and at the same time, we will also show that lemma one that we stated…. We essentially till now proved an inequality, that is, L_{N+1} is was actually greater than equal to the maximum of $L_N(S)$ or N plus 1 minus $L_N(S)$. Actually that greater than equal to should be replaced by equality, that is, it will be equality instead of a greater than equal to. In order to prove that, we will essentially develop a proof, this merges these two proofs. So, we will prove that equality by induction and at the same time we will give you a proposal for a polynomial, which will be a candidate for $C_{N+1}D$.

(Refer Slide Time: 26:15)



Discrepancy

Lemma 1 is actually an equality. We have seen this for the base case.

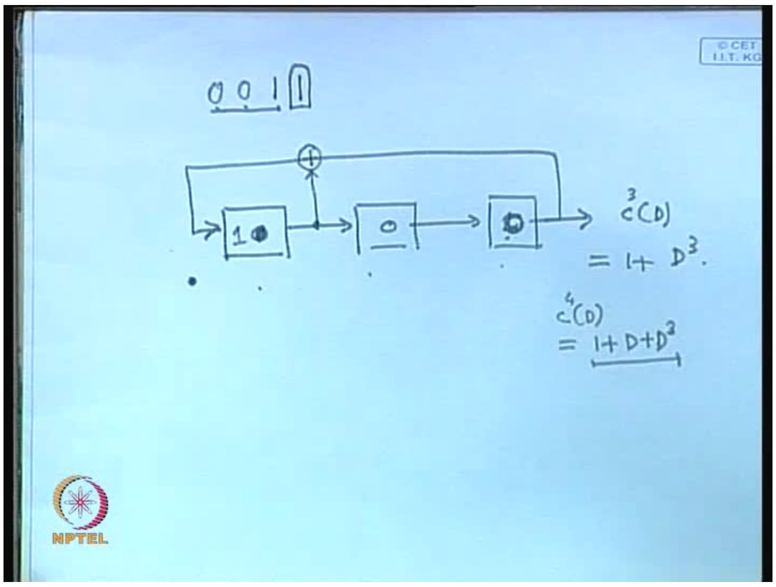
Assume an induction hypothesis for $L_N(s)$.

The corresponding polynomial is $C^N(D)$.

$$\therefore s_j \oplus \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} = \begin{cases} 0, & j = L_{n(s)}, \ldots, n-1 \\ d_n, & j = n \end{cases}$$

d_n : next discrepancy (between s_n and the (n+1)st bit generated by the minimal length LFSR, which we have found to generate the first n bits of s.

(Refer Slide Time: 26:35)



This is a constructive way of proving. This is not an existential proof but a constructive proof. So, we will not only prove the existence but we will also show you how it is constructed. In order to do that, we will define something called discrepancy, which we have already seen. Why? If you remember the previous example, we were able to calculate the values till a particular point. For example, if you remember, we had 0 0 1 1, that 0 0 1 can be quite trivially constructive. How? You can just take three-stage LFSR and you can just write the feedback. The feedback polynomial could be simply a shift register like this, so you can take 0 0 and 1 and you can stream out the value of 0 0 1.

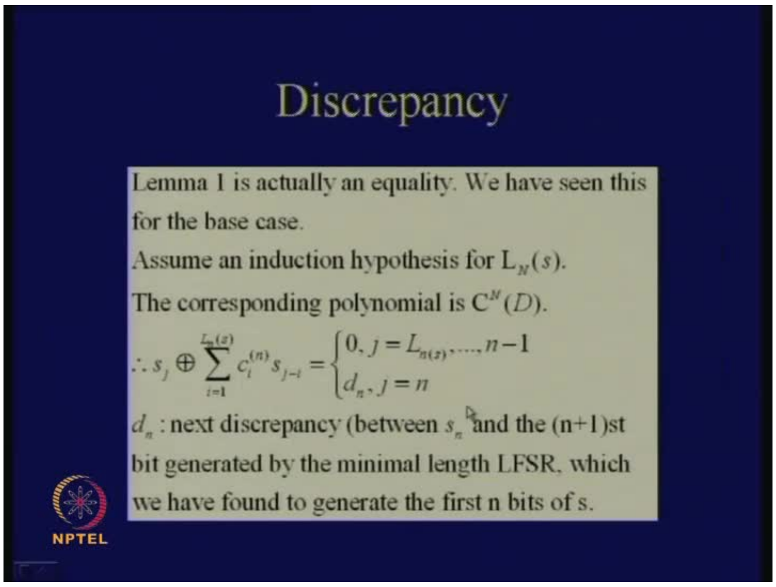
What is the corresponding polynomial here? C D is equal to 1 plus D cube. But the moment you see that you consider this particular 1, the next thing that if get feedback fed back? it is actually a 0. Right? When you want a 1 here, you are actually getting a with this particular LFSR, and you are getting back 0. That means, there is a discrepancy. This is the idea of discrepancy. What are you actually getting back and what you want –exhort between these two. The moment you see that you actually get back 1 and you want is 1, you know that you have to make certain modifications in this structure. I will just do a simple modification, I will introduce exhort here, and take this feedback.

In that case, this LFSR will generate the sequence 0 0 1 and 1. If you remember, the previous thing what we did is we actually solve this and found this. So, you see that you get 0 0 1, that is quite trivial, but the next thing is an exhort between 0 and 1, so you get back 1. Till this point is it fine? What is the updated polynomial now? C D. If I write in terms of C and D, this was C_3 D but this is C_4 D. What is C_4 here? 1 plus D plus D cube. Therefore, you see that you are able to construct the corresponding polynomial but when you are doing this algorithm, you will take care of the fact that there is a discrepancy at this point, and accordingly modify the polynomial. And when there is no discrepancy, there is no modification required. You can just go ahead with the previous polynomial -- previous construction of the LFSR.

Yes.

(( ))

(Refer Slide Time: 29:4)



Discrepancy

Lemma 1 is actually an equality. We have seen this for the base case.

Assume an induction hypothesis for $L_N(s)$.

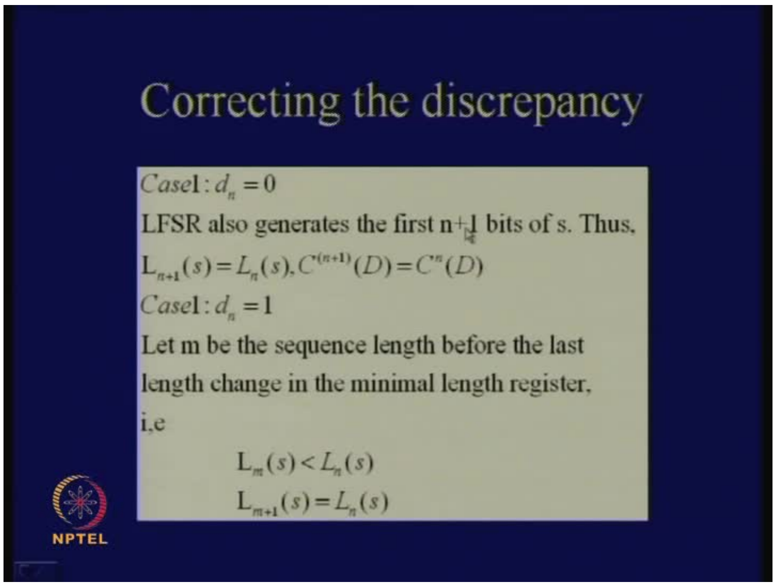
The corresponding polynomial is $C^N(D)$.

$$\therefore s_j \oplus \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} = \begin{cases} 0, j = L_{n(s)}, \ldots, n-1 \\ d_n, j = n \end{cases}$$

d_n : next discrepancy (between s_n and the (n+1)st bit generated by the minimal length LFSR, which we have found to generate the first n bits of s.

It will be 1 0 0. Yeah, I am streaming out from this point, so it will be 0 here, 0 here, and 1 here. But anyway the rest of the discussion holds. So, I am getting back first 0 0 and then 1. Right? Therefore, you see that whenever there is a discrepancy introduced here, you are only required to modify this polynomial. Now, you come to this particular definition of discrepancy -- it says that lemma one is actually equality, and we have seen this for the base case but we will prove it for for the further cases, using induction. Why I have said it holds for the base case? Because, I have already proved you that for 0 0 1 sequence, that was exactly equal to 3, because you started with L equal to 0 and then it was actually equal to N plus 1 minus 0. So, it was actually equal to N plus 1. For the base case, the equality holds, for the rest of the cases, we will prove it by induction. Our hypothesis would be that this holds for L_N(S), and we will try to prove this for N L plus 1 S. Therefore, the corresponding polynomial in this case is C and D. This is the phenomenon, which is according to when you need an update. When do you need an update? When this S_j exhort with what you exactly obtain back, this is what you want and this is what you obtain, if you take an exhort you get 0 for all the cases till N minus but for j equal to N minus you get a non 0 value. So, you get 1 here. This D N is called the next discrepancy and you need an update only when D N is equal to 1.

(Refer Slide Time: 31:32)



Correcting the discrepancy

$Case1 : d_n = 0$

LFSR also generates the first n+1 bits of s. Thus,

$L_{n+1}(s) = L_n(s), C^{(n+1)}(D) = C^n(D)$

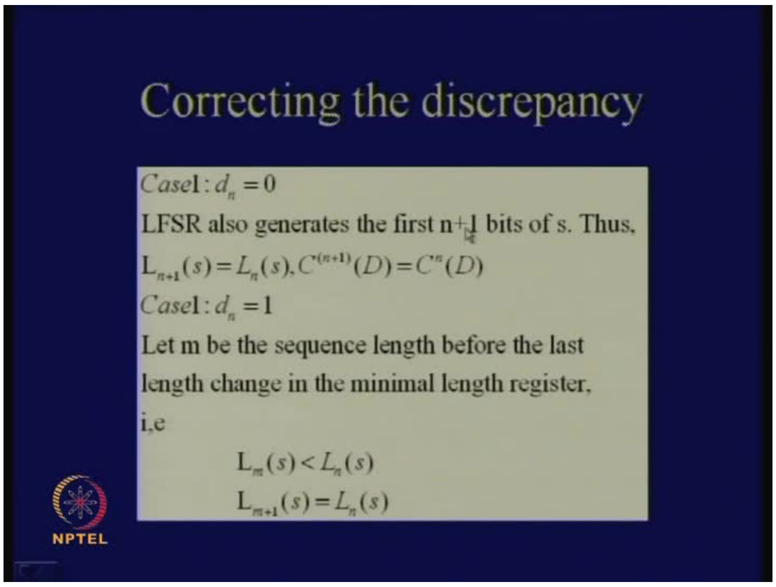
$Case1 : d_n = 1$

Let m be the sequence length before the last length change in the minimal length register, i.e

$$L_m(s) < L_n(s)$$
$$L_{m+1}(s) = L_n(s)$$

Therefore, you see that this is the discrepancy between S_n and N plus 1, first bit, which is generated by the minimal length LFSR, which we have found to generate the first N bits of S. You understand the idea behind discrepancy? This is the difference, that is, exhort between what you actually obtain and what you wish to obtain. Therefore, when D_n is equal to 0, then the LFSR also generates the first N plus 1 bit, and therefore there is nothing to be done. So, N plus 1 S is equal to $L_n(S)$, the linear complexity will remain the same. C n plus 1 D also remains the same as that of C n D. Problem happens when D_n is equal to 1. There, we need a correction of this discrepancy. So you note one thing that in order to do that, we will consider the sequence length before the last. Therefore, let m be the sequence length before the last length change in the minimal length register. Before this, whichever was the last change when you did for the L f length, that is, I mean, so m is a variable which is the sequence length before the last length change in the minimal length register. Therefore, last time when you change the length, what was the sequence length? That value is being held by the variable N. These are recursive algorithm, so you are keeping on updating one after the other. You are modifying the length sometimes, you are not modifying the length sometimes. Therefore, when was the last time? Suppose, you are maintaining this as a form of a table, you go back and see when is the last time when you changed the value of length and find out that what was the corresponding sequence length.

(Refer Slide Time: 32:57)



Correcting the discrepancy

$Case1: d_n = 0$

LFSR also generates the first n+1 bits of s. Thus,

$L_{n+1}(s) = L_n(s), C^{(n+1)}(D) = C^n(D)$

$Case1: d_n = 1$

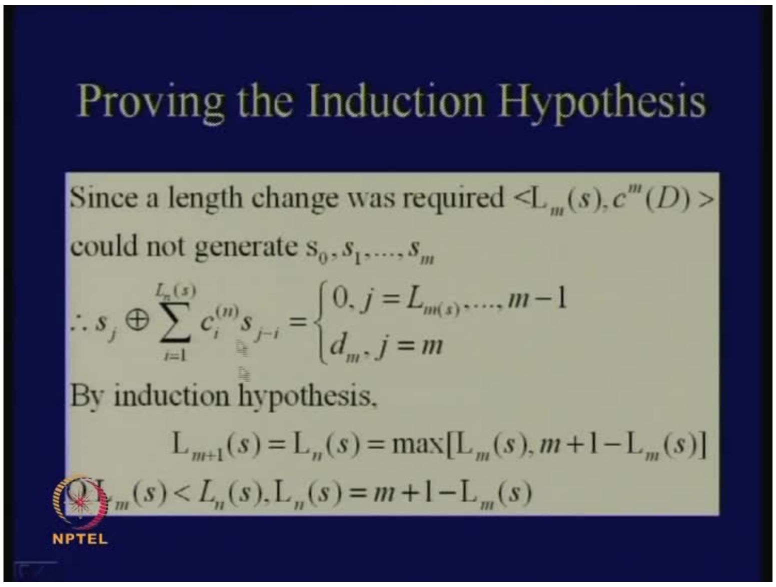
Let m be the sequence length before the last length change in the minimal length register,

i.e

$$L_m(s) < L_n(s)$$
$$L_{m+1}(s) = L_n(s)$$

That is suppose m, so immediately you can understand that these two things hold, that is, $L_m(S)$ is less than $L_n(S)$ because of monotonicity. And $L_{m+1}(S)$ is actually equal to $L_m(S)$ because you have updated your m value to satisfy this. So this was the last change. I mean, m is the sequence but this is the length, so you have changed the length, which means that you have gone from m to m plus 1. Therefore $L_n(S)$ must be greater than L_m because it was the last change. m is the greatest integer before the last change. I mean, m is the sequence actually, so $L_m(S)$ is the corresponding length of the LFSR, and $L_{n+1}(S)$ must be equal to $L_n(S)$ because this is the length that we are talking about.

(Refer Slide Time: 34:07)



Proving the Induction Hypothesis

Since a length change was required $<L_m(s), c^m(D)>$ could not generate $s_0, s_1, \ldots, s_m$

$$\therefore s_j \oplus \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} = \begin{cases} 0, j = L_{m(s)}, \ldots, m-1 \\ d_m, j = m \end{cases}$$

By induction hypothesis,

$$L_{m+1}(s) = L_n(s) = \max[L_m(s), m+1-L_m(s)]$$

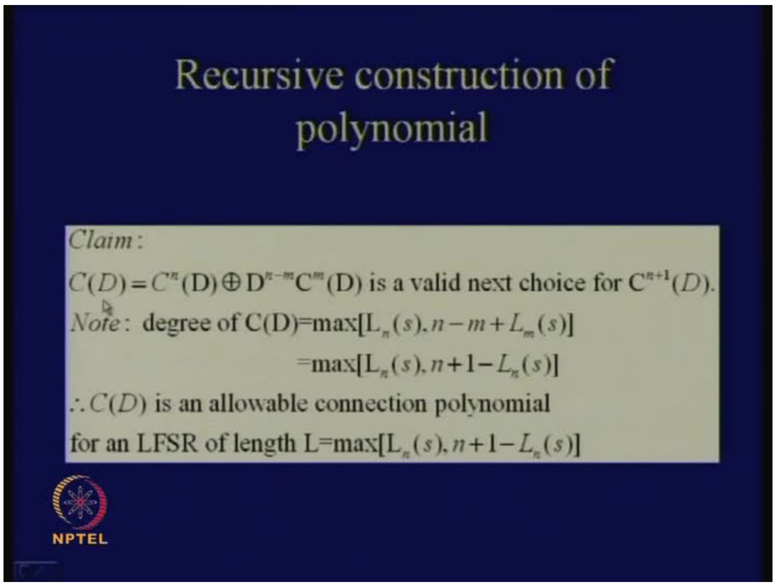
$\because L_m(s) < L_n(s), L_n(s) = m+1-L_m(s)$

Therefore, since a length change was required, so you see that only at j equal to m. There must be a discrepancy. But otherwise, this would have been held fine right. Therefore the discrepancy was 0 in all these cases. Therefore since, the length change was required, this LFSR, that is, $L_m(S)$ and with the corresponding polynomial. This will be a capital. So, C_m D, this could not generate the sequence from S_0 to S_m. That is why you change the length. So, what is the sequence length here? m plus 1. So, in order to generate the m plus one th sequence length, you had made a change in the value of m.

Therefore, immediately, you can understand that the corresponding discrepancy for all these values till m minus 1 was equal to 0, but when j was equal to m, the discrepancy was a non-0 value. By induction hypothesis, you can understand that $L_{m+1}(S)$ is equal to $L_n(S)$. This we have already discussed. By induction hypothesis, $L_n(S)$ is actually equal to max of $L_m(S)$ and m plus 1 minus L l m S. You can apply this and note that $L_m(S)$ is actually less than $L_m(S)$. Therefore, rather this, this q is something, which is generated the software. So this is actually a since. Since $L_m(S)$ is less than $L_n(S)$, therefore $L_n(S)$ is actually equal to m plus one minus $L_m(S)$. So, $L_n(S)$ is actually m plus 1 minus $L_m(S)$.

Now, please remember this value till this point, that is, $L_n(S)$ was equal to m plus 1 minus $L_m(S)$. So after this, we will give you a proposal for the next candidate. Till now, we have assumed that we know C m D and we would like to find out C n plus 1 D, which generates the sequence till S_n.

(Refer Slide Time: 36:27)



Recursive construction of polynomial

Claim:

$C(D) = C^n(D) \oplus D^{n-m} C^m(D)$ is a valid next choice for $C^{n+1}(D)$.

Note: degree of $C(D) = \max[L_n(s), n-m+L_m(s)]$

$= \max[L_n(s), n+1-L_n(s)]$

$\therefore C(D)$ is an allowable connection polynomial

for an LFSR of length $L = \max[L_n(s), n+1-L_n(s)]$

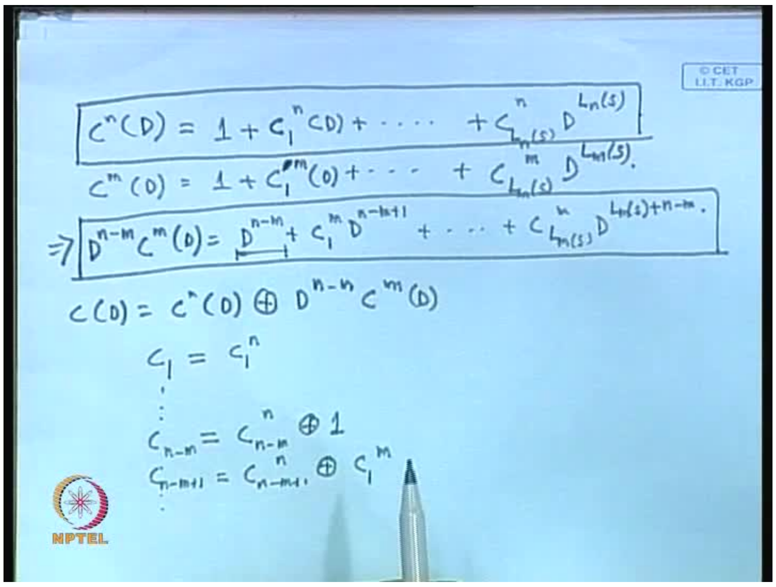
NPTEL

So, this the corresponding construction. The claim is that if you exhort C n D with C m D, which is multiplied by D to the power of N minus m, this is the valid next choice for C n minus one. So, you see that we have taken this and exhort with this. The idea of this exhort is to correct the corresponding discrepancy value. At the point of your problem, this gives you a discrepancy, this also gives you a discrepancy, and both these discrepancies actually cancelled each other.

That is the idea. But let us see this more closely to understand this phenomenon. So, what is the degree of C D? The degree of C D would be the maximum of this degree. What is the degree of this? L_n(S). And, what is degree of this? L_m(S). But n minus m is multiplied, so it is n minus m plus L_m(S). Note that n minus m plus L_m(S) can be actually written equal to n plus 1 minus L_n(S). Why? Because of the previous equality that we found out. If you remember, the previous equality L_n(S) was equal to m plus one minus L_m(S). If you substitute this value here, you obtain this. Therefore C D is actually an allowable connection polynomial because maximum length is inside this. Now, only one thing remains to prove that C D does the connection. Because if C D does the connection, then what you have proved is the that the length, that is the next length, that is actually equal to maximum of L_n(S) and n plus 1 minus L_n(S).

So, you have proved that L_{n+1}(S) is actually equal to maximum of L_n(S), n plus 1 minus L_n(S). And therefore the induction gets proved. We get the proof by induction. But till now, we have actually not proved one thing -- C D actually does the correction. Therefore, what we now need to prove is that C D does the correction, which means it generates the sequence digit S_n, and at the same time does not disturb the previous sequences. So, in order to understand this, let us observe the value of C n D.

(Refer Slide Time: 38:58)



$$\boxed{C^n(D) = 1 + C_1^n(D) + \cdots + C_{L_n(S)}^n D^{L_n(S)}}$$

$$C^m(D) = 1 + C_1^m(D) + \cdots + C_{L_m(S)}^m D^{L_m(S)}.$$

$$\Rightarrow \boxed{D^{n-m} C^m(D) = D^{n-m} + C_1^m D^{n-m+1} + \cdots + C_{L_m(S)}^m D^{L_m(S)+n-m}.}$$

$$C(D) = C^n(D) \oplus D^{n-m} C^m(D)$$

$$C_1 = C_1^n$$

$$\vdots$$

$$C_{n-m} = C_{n-m}^n \oplus 1$$

$$C_{n-m+1} = C_{n-m+1}^n \oplus C_1^m$$

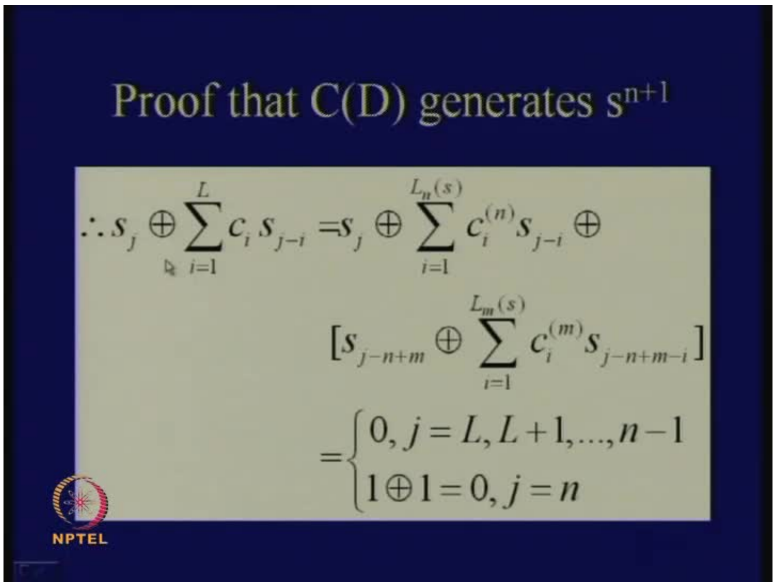
$$\vdots$$

© CET
I.I.T. KGP
NPTEL

What was C n D equal to? It was equal to one exhort with c_1 n D. This was small c plus and so on till $C_{Ln(S)}$ n D to the power of $L_n(S)$. And what was C m D equal to? One plus C_{m1} dash or you can write 1 plus C_1 m, one plus C_1 m D plus till $C_{Lm(S)}$ m D to the power of $L_m(S)$. Now, if you multiply this with D power of n minus m C m D, then what do you obtain? It is D to the power of N minus m plus C_1 m. What does this D becomes? N minus m plus 1 plus $C_{Lm(S)}$ m D to the power of $L_m(S)$ plus N minus m. So, you note this equation and you note this equation. Your C D is an exhort between these two polynomials. By my definition, C D was C n D exhort with D to the power of n minus m C of m D. Now, I can enumerate all the corresponding coefficients for C D in terms of these two coefficients.

How can I write? I mean the coefficient will be correspondingly C_1, it will be equal to C_1 n and so on till C_{n-m}. C_{n-m} will be C_{n-m} n exhort with 1 because you will have correspondingly this particular term will give you a coefficient of 1 and so on. I mean you can write C_{n-m+1}, this will be equal to C_{n-m+1} an exhort with C_1 m. So, subsequently you have till this particular point, therefore, after a point there will be an exhort. Now, note that if I need to find out the corresponding discrepancy value, I shall be interested in computing this value -- S_j exhort with C_i S_{j-1} sigma, where I runs from 1 L.

(Refer Slide Time: 42:00)



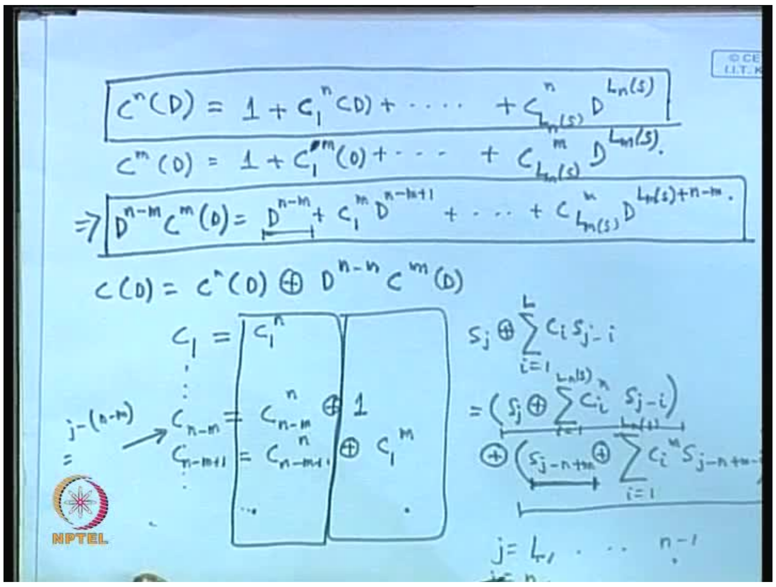
Proof that C(D) generates s^{n+1}

$$\therefore s_j \oplus \sum_{i=1}^{L} c_i s_{j-i} = s_j \oplus \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} \oplus$$

$$[s_{j-n+m} \oplus \sum_{i=1}^{L_m(s)} c_i^{(m)} s_{j-n+m-i}]$$

$$= \begin{cases} 0, j = L, L+1, \ldots, n-1 \\ 1 \oplus 1 = 0, j = n \end{cases}$$

(Refer Slide Time: 42:28)



$$\boxed{c^n(D) = 1 + c_1^n(D) + \cdots + c_{L_n(s)}^n D^{L_n(s)}}$$

$$c^m(D) = 1 + c_1^m(D) + \cdots + c_{L_m(s)}^m D^{L_m(s)}.$$

$$\Rightarrow \boxed{D^{n-m} c^m(D) = D^{n-m} + c_1^m D^{n-m+1} + \cdots + c_{L_m(s)}^m D^{L_m(s)+n-m}.}$$

$$c(D) = c^n(D) \oplus D^{n-m} c^m(D)$$

$$c_1 = c_1^n$$

$$\vdots$$

$$c_{n-m} = c_{n-m}^n \oplus 1$$

$$c_{n-m+1} = c_{n-m+1}^n \oplus c_1^m$$

$$\vdots$$

$$j-(n-m) =$$

$$s_j \oplus \sum_{i=1}^{L} c_i s_{j-i}$$

$$= (s_j \oplus \sum_{i=1}^{L_n(s)} c_i^n s_{j-i}) \oplus (s_{j-n+m} \oplus \sum_{i=1} c_i^m s_{j-n+m-}$$

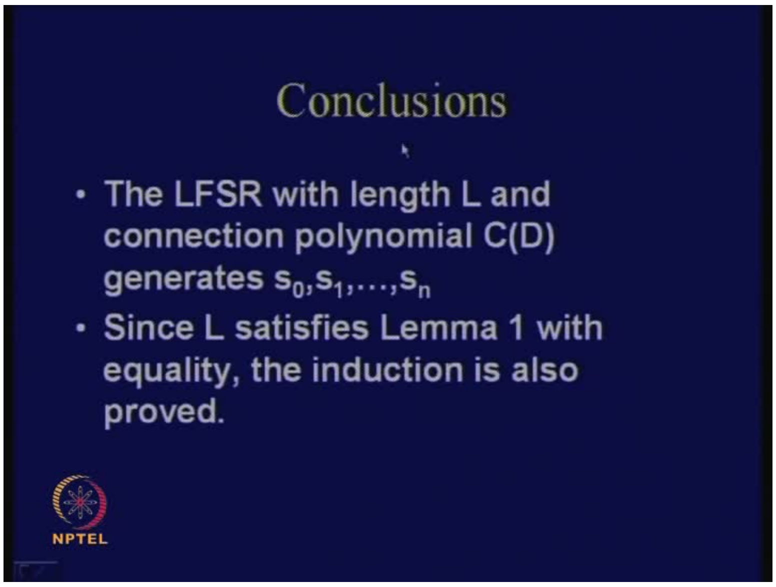
$$j = L, \ldots n-1$$

I need to ensure that this is actually equal to 0 for j running from L to n, for all the values. So you note that I can now write this in two separate parts because if you remember, the corresponding coefficient C$_i$, that is, these coefficients, I can write as an exhort of this exhort with this. Therefore, this gives you the first part and this gives you the second part. If I am interested in computing S$_j$ exhort with C$_i$ S$_{j-I}$, where I runs from say, 1 to l. Then this will be equal to S$_j$ exhort with sigma C$_i$ n S$_{j-I}$. This is one part. and the other part will be the corresponding part from S$_{j-n+m}$ plus the other part of the sigma. What is the other part of the sigma? I hope you have understood why I am writing j minus n plus m. Because at this point I am writing out the corresponding coefficient for

C_{n-m}. C_{n-m} is this and the corresponding seed value will be j minus n plus j minus. I mean j minus n minus m, so that is exactly this value. Therefore, the rest part of the sigma comes in, and you have got C_i m and $S_{j-n+m+I}$. Note that this I will run from 1 to $L_m(S)$ and this will run from 1 to $L_n(S)$. For all the previous values, that is, for j running from L to n minus 1, this value will be 0, because this LFSR was properly generating all these values. But what about this? The suffix here is j minus n plus m. When a j runs from L to n minus 1, you substitute here n, n minus 1, then you get n minus 1, n minus 1 minus n plus m. so that is n minus 1.

Therefore this sequence is actually generating the first m digits of the sequence of its corresponding LFSR. Therefore, this was actually properly generating, this value is 0 value. This a 0 and this is a 0, so you get 0 for j running from L to n minus 1. And what about when j equal to n? This will generate a 1 because of the discrepancy and this will also generate a 1 because of the discrepancy, and both the discrepancies will get cancelled out. Therefore we say that the corresponding coefficient C D actually generates the entire sequence S_{n+1}. So, this is not so trivial proof, therefore, please go back and look at the proof again and try to work with some small examples.

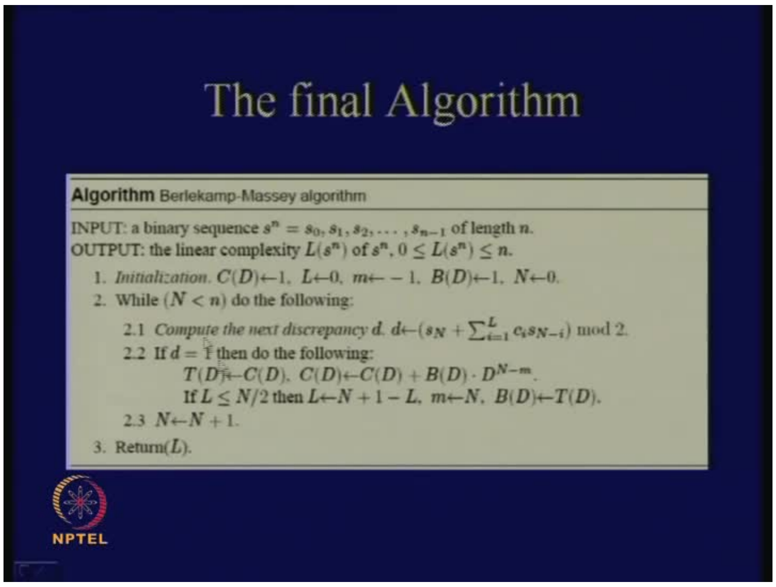
(Refer Slide Time: 45:53)



Conclusions

- The LFSR with length L and connection polynomial C(D) generates $s_0,s_1,\ldots,s_n$
- Since L satisfies Lemma 1 with equality, the induction is also proved.

(Refer Slide Time: 46:29)



The final Algorithm

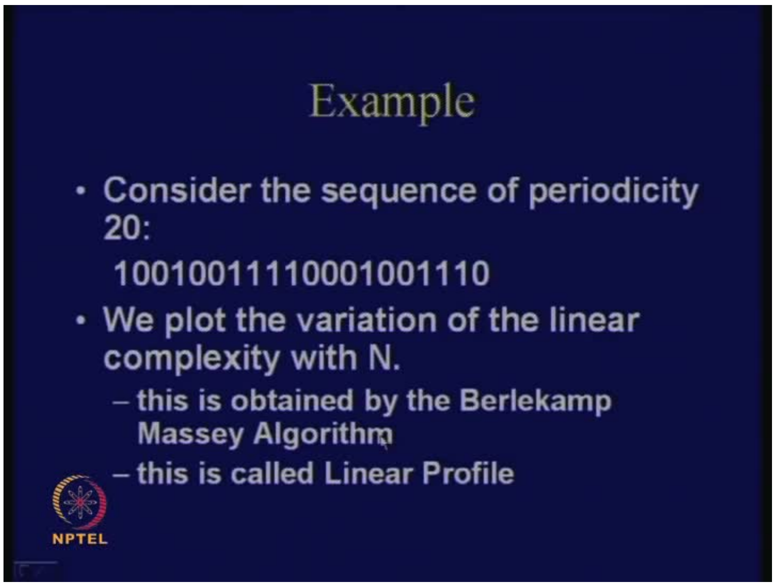
Algorithm Berlekamp-Massey algorithm

INPUT: a binary sequence $s^n = s_0, s_1, s_2, \ldots, s_{n-1}$ of length n.
OUTPUT: the linear complexity $L(s^n)$ of s^n, $0 \leq L(s^n) \leq n$.

```
1. Initialization. $C(D) \leftarrow 1$, $L \leftarrow 0$, $m \leftarrow -1$, $B(D) \leftarrow 1$, $N \leftarrow 0$.
2. While $(N < n)$ do the following:
    2.1 Compute the next discrepancy d. $d \leftarrow (s_N + \sum_{i=1}^{L} c_i s_{N-i}) \bmod 2$.
    2.2 If $d = 1$ then do the following:
            $T(D) \leftarrow C(D)$, $C(D) \leftarrow C(D) + B(D) \cdot D^{N-m}$.
            If $L \leq N/2$ then $L \leftarrow N + 1 - L$, $m \leftarrow N$, $B(D) \leftarrow T(D)$.
    2.3 $N \leftarrow N + 1$.
3. Return(L).
```

The conclusions are that the LFSR with length L and connection polynomial C D generates the sequence S_0 to S_n. Since L satisfies lemma 1 with equality, the induction also gets proved. So, you saw that L was actually equal to L_n(S), maximum of L_n(S), n plus 1 minus L_n(S). Therefore, we are proving the equality for n plus 1 S. I am not going into the details of the algorithm, which you can follow from this discussion. You must note one thing that what you are doing is that you are maintaining some temporary variables like B(D) and m and things like that. m I have already defined, so B(D) is another temporary variable and T(D) is another temporary variable, and you note that whenever there is a D equal to 1, so you are calculating the discrepancy. Whenever this D is equal to 1, you are doing certain operations and when do you require modifying to the length?

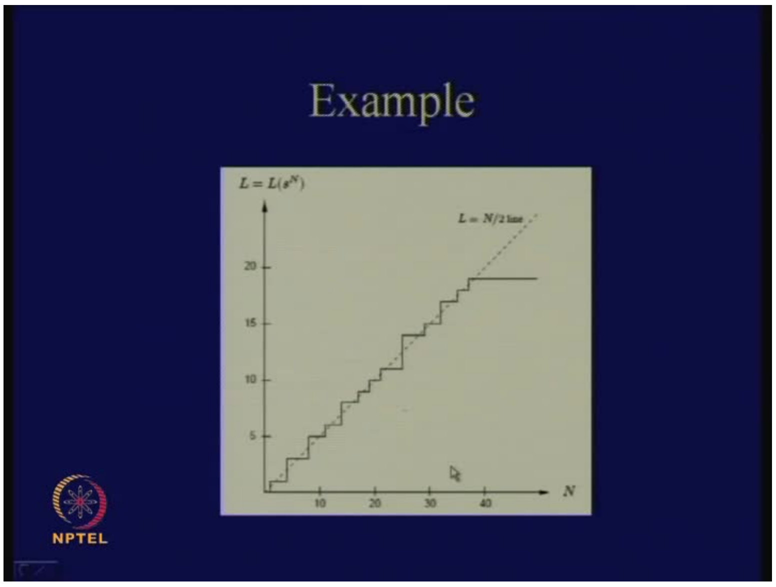
(Refer Slide Time: 47:12)



Example

- Consider the sequence of periodicity 20:
  10010011110001001110
- We plot the variation of the linear complexity with N.
  - this is obtained by the Berlekamp Massey Algorithm
  - this is called Linear Profile

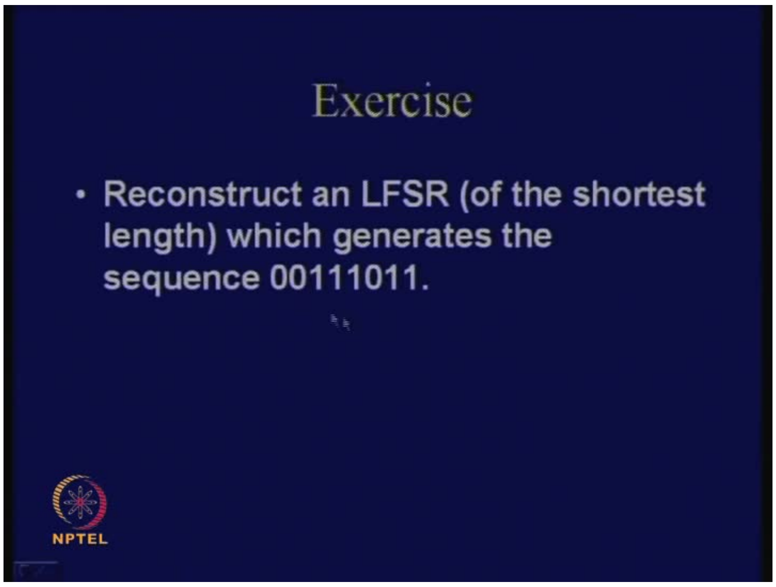
(Refer Slide Time: 47:38)



Example

When L is less than equal to n by 2. I described why it is so and only at that particular time you require to update the value of L, otherwise the L is fine. You can work through the details of this thing but I will give you some examples. I will tell you one thing that until and unless you work with your hand you will not be so clear. Therefore, please go back and work with some toy examples. Consider this sequence of periodicity 20, therefore this is for example, a sequence. You can actually plot the variation of the linear complexity with n by calculating this using Berlekamp Massey algorithm. And this is actually called the linear profile, so you can actually plot them and it will look like this. If this is line corresponding to L equal to n by 2, then whenever L is less than equal to n by 2, the modification has taken place. Otherwise, when L is actually greater than n by 2, the modification of L is not equal. You get the step function at those places. So, you can little bit look in more closely

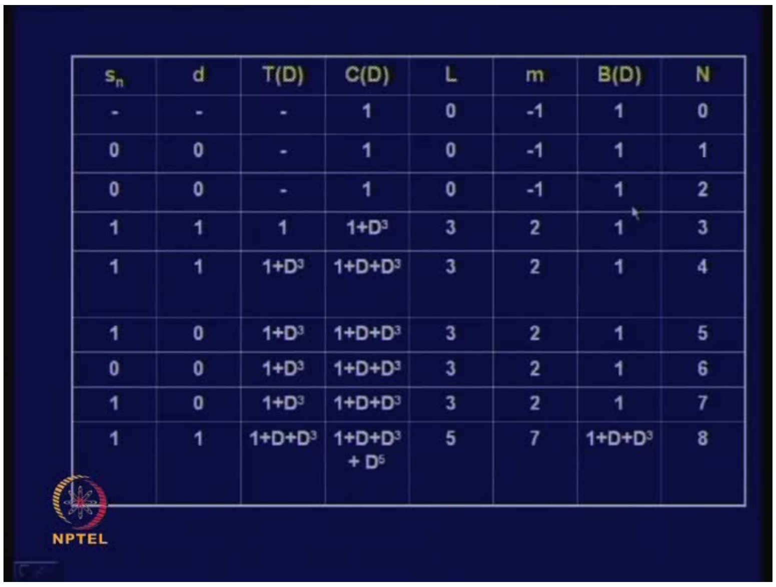
(Refer Slide Time: 48:04)



Exercise

- Reconstruct an LFSR (of the shortest length) which generates the sequence 00111011.

(Refer Slide Time: 48:20)



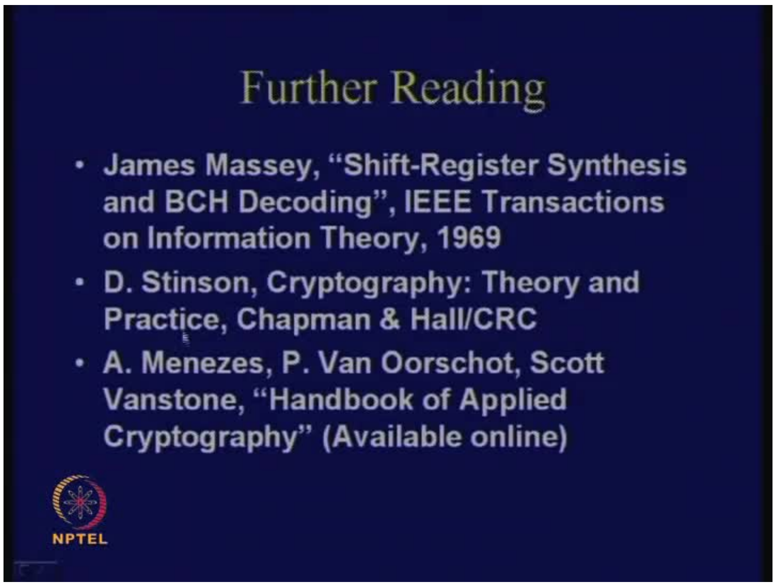
s_n	d	T(D)	C(D)	L	m	B(D)	N
-	-	-	1	0	-1	1	0
0	0	-	1	0	-1	1	1
0	0	-	1	0	-1	1	2
1	1	1	$1+D^3$	3	2	1	3
1	1	$1+D^3$	$1+D+D^3$	3	2	1	4
1	0	$1+D^3$	$1+D+D^3$	3	2	1	5
0	0	$1+D^3$	$1+D+D^3$	3	2	1	6
1	0	$1+D^3$	$1+D+D^3$	3	2	1	7
1	1	$1+D+D^3$	$1+D+D^3 + D^5$	5	7	$1+D+D^3$	8

To obtain certain interesting properties, I will conclude with the example with which I started. Therefore, this is the sequence. Yesterday we saw that a four-state LFSR was unable to generate the sequence. So, in order to solve the problem, let us apply the Berlekamp Massey algorithm. To do that, you will store that in the form of a table like this. So you see that S n d T(D) C(D) L m B(D) and N are some variables up there, and we will start operating this table. This is the sequence, which you wish to generate from 0 0 1 1 1 0 1 1. You start with a value of C(D) equal to 1, that was my convention, and your L value was 0, and B(D), I mean, m is actually in this case you start with initialize with minus 1, and B D is the corresponding polynomial which generates the sequence. This was my C m D in my discussion in the analysis. So, n is equal to 0 means still now you have generated till 0. That is, you have generated nothing till this point. This is just a initialization of the algorithm. So what you do next is you get 0. What is the discrepancy? It is 0, therefore, you do not require to do any other thing. Therefore, you see that all the things are kept intact. Next what you get is again a 0, you have a discrepancy of 0 and therefore you do not do anything. But the next thing you get is a 1. The moment you get 1, your discrepancy is equal to 1, because you feeding back 0 and you are getting what you want, that is, 1. Therefore, the exhort is 1. So, we require updating the corresponding value of C(D). So you see that what you do is that you exhort the previous value of C(D), that is, 1 with the corresponding value of B(D) but you multiply with D to the power of N minus m.

What is your N here? Two. And your m value is minus 1, so minus 1 is 2 plus 1, 3, so you get 1 plus D cube. You see that 1 plus D cube, which has the length of 3 should be able to generate this particular sequence 0 0 1. And this we have already seen right with our hand exercise. Therefore, what is the value of m? Equal to two. So, m is the previous sequence before the length got changed. The length got changed here from 0 to 3 and what you have generated previously was till 3 (2). Therefore, m is actually equal to the previous value, therefore in this case, its 2, and the corresponding polynomial was 1. Now, you have generated till 3, that is, the I mean 0 1 2 3. So, the next thing, which you get is 1, and you again find that discrepancy here, which means that you need to change the value 1 plus D plus, I mean, you need to change the value of C(D).

So what you do is that you take N and you see that this is m, therefore 3 minus 2 is 1. So, you multiply this with 1 plus D cube and you also exhort the previous value of B(D) but multiplying that with D. Therefore, you get 1 plus D plus D cube, so you get 1 plus D plus D cube and your length… You see that you are not updating the length. Can you tell me why? Yeah, because of that inequality. Therefore, in this case, you do not require to update this value and therefore you go ahead with it. So, you have generated till this point. Therefore, you see that it works fine here for these things because it generates the discrepancy of 0. There is no other updating required for these stages. But at this point we were unable to construct with a 3-bit LFSR. And you see that the discrepancy is in this case is 1. Why? Because 1 plus D cube, so if you exhort the previous this value and this value, that is the previous sequence values to get 1 on 1 exhort to actually feeding back 0 but what you want is a 1. What is the discrepancy value? Its equal to one, therefore, you need to modify the value of C(D). Therefore, you take 1 plus D plus D cube and exhort that with the previous value of B(D). What is the previous value of B D? It is equal to one multiplied with D to the power of 7 minus 2, that is, D to the power of 5. So, you take 1 plus D plus D cube plus D to the power of 5 and you get L equal to 5. So, you see that you have modified the length again because 3 is actually smaller than 7 by 2. Therefore, you have modified the value of this to 5, and therefore you see that 5-stage LFSR with this particular polynomial is able to generate the entire sequence.

(Refer Slide Time: 53:26)



Further Reading

- James Massey, "Shift-Register Synthesis and BCH Decoding", IEEE Transactions on Information Theory, 1969
- D. Stinson, Cryptography: Theory and Practice, Chapman & Hall/CRC
- A. Menezes, P. Van Oorschot, Scott Vanstone, "Handbook of Applied Cryptography" (Available online)

Therefore, by using Berlekamp Massey algorithm, you are able to calculate the corresponding minimal length of the LFSR, which will generate the sequence and also find out the corresponding polynomial. And actually, you can see from the algorithm statement the complexity of this algorithm is o N square. Therefore, this is a quadratic algorithm to generate the corresponding sequence. This is quite efficient in that case. So, references that I have followed are as follows. These are standard references but I will suggest you to go back and read this paper. Its freely downloadable, it is got shift registers synthesis and B C h decoding. Just concentrate on the first part of this paper. This gives you a description but this actually tops in prime fields and I generalize this proof for g f 2, I mean not generalize, many more specific. This is a classic paper, it IEEE transactions on information theory paper but you understand… So, what you have learnt from this example is that a single LFSR system is not good.

(Refer Slide Time: 54:25)

Next Days Topic

- Stream Ciphers (contd.)

If you have a single LFSR-based stream cipher, then you can actually do a known plaintext attack. You can obtain the key stream and from that, you can actually reconstruct the LFSR. You can know everything about LFSR. Given two N bit sequence, you can construct the entire thing, therefore, you need a multi LFSR system. In the future classes, in the next day classes, we will still continue with stream ciphers and we will try to understand more detailed, I mean, better constructions of stream ciphers using LFSR's, and also not LFSR's.