# Information Theory and Coding

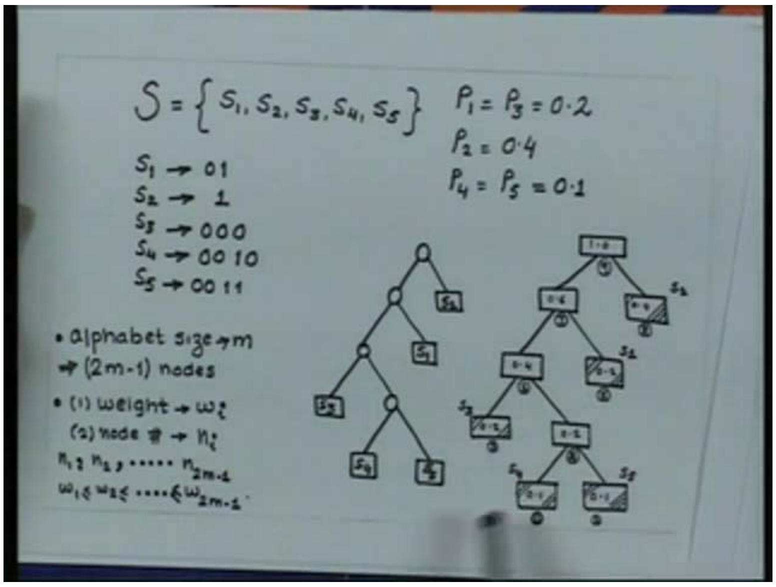
**Prof. S. N. Merchant**
**Department of Electrical Engineering**
**Indian Institute of Technology, Bombay**

## Lecture - 15
## Adaptive Huffman Coding Part – I

Huffman code are optimal for a given source model. So, if the source model changes then we have to recompute the Huffman code, therefore in those situation where the source model is not available or the source model changes. Then we have to go for adaptive Huffman code. In today’s class, we will have a look at the procedure for construction of adaptive Huffman code. Adaptive Huffman code is constructed on the statistics of the signal already encountered.

Theoretically, if we have to encode the k plus one symbol using the statistic of the k serial symbol, then we can recompute the code using the Huffman coding procedure each time a symbol is transmitted. However, this could be a very impractical approach, due to the large amount of computation involved. Therefore, it is necessary that we develop a single part adaptive Huffman coding procedure. Adaptive Huffman code works on the principle that the Huffman code can be represented by a binary tree. So, let us revisit the example, which we have studied earlier and where we had shown how to develop the Huffman code for it

(Refer Slide Time: 02:52)



$S = \{s_1, s_2, s_3, s_4, s_5\}$ $P_1 = P_3 = 0.2$

$P_2 = 0.4$

$P_4 = P_5 = 0.1$

$s_1 \rightarrow 01$
$s_2 \rightarrow 1$
$s_3 \rightarrow 000$
$s_4 \rightarrow 0010$
$s_5 \rightarrow 0011$

- alphabet size $\rightarrow m$
  $\rightarrow (2m-1)$ nodes
- (1) weight $\rightarrow w_i$
  (2) node # $\rightarrow n_i$
  $n_1, n_2, \dots\dots n_{2m-1}$
  $w_1 \leq w_2 \leq \dots\dots \leq w_{2m-1}$

So, here is a example. I have a source consisting of five symbols and the probability of the symbol of the source model is indicated here. For this source model, we can design the Huffman code, which we have done earlier and the code is indicated out here. Now, from this code, we can construct the binary tree representing this code. That will be as shown here, so this is a binary tree representing this code. We have external nodes or leaves for the symbols S 1, S 2, S 3, S 4, S 5.

These are the internal nodes and this is the root node. Now, if the alphabet size is m, then there will be 2 m minus 1 nodes including internal and external nodes. Now, in this case m is equal to 5, so 2 m minus 1 comes out to be none. So, there are 9 nodes. Now, in order to understand the working of adaptive Huffman code, we add two parameters to the binary tree. These parameters are weight of a node and the node number, in order to appreciate the significance of this parameter in the context of adaptive Huffman code. Let us first examine this parameter in the context of the binary tree for a non adaptive Huffman code.

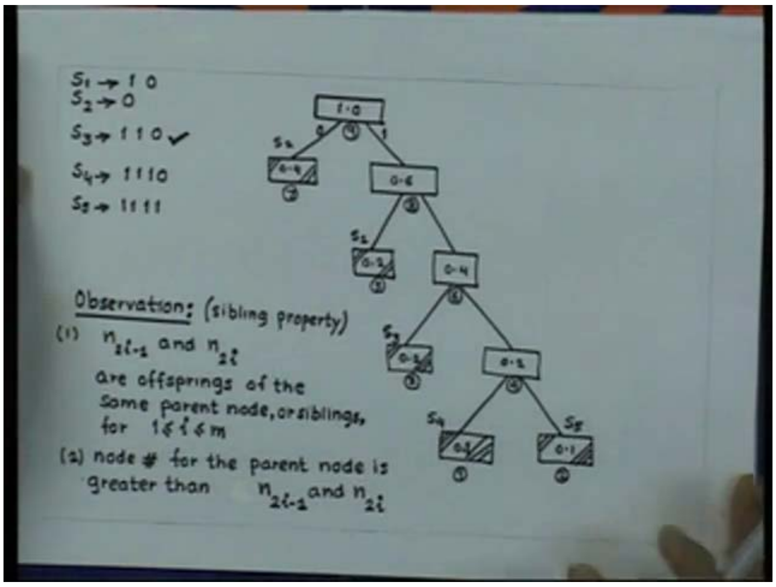
So, for the binary tree under discussion, we can associate two parameters weight and the node number. So, weight is indicated by w i and the node number is indicated as n i. Now, the weight for the external nodes or the leaves corresponding to the symbol is the probability of a occurrence of that symbol. So, the weight of S 2 would be 0.4 and the weight of S 3 0.2. Similarly, the weights of the internal nodes are given by the sum of the weights of the offspring's. So, the weight of the internal node for example, this node will be given by the weight by the sum of the weights of S 4 and S 5; that will be 0.1 plus 0.1 is 0.2.

Now, we will see that in the adaptive Huffman code, the numbering is done in a particular fashion in order to preserve the fixed order of the nodes the numbering. That is adopted for the adaptive Huffman code is as follows. If I have the numbers given to the node as n 1, n 2 up to n 2 m minus 1, then I number the nodes in this fashion, in such a way that the requirement on the weight of the node satisfy this relationship w 1 is less than equal to w 2 is less than equal to w 2 m minus 1.

In the adaptive Huffman code, this numbering is done from left to right and bottom to top. Now, if we adopt the same procedure for the binary tree under discussion. Then we get the following tree out here, I have numbered this node from left to right and bottom to top 1, 2, 3, 4, 5, 6, 7, 8, 9 and in each of this node I have indicated the ways the shaded one correspond to the external node or the leaf corresponding to the symbol. Now, if you look at this tree and look at the numbering which we have done, then it is very obvious from here that this numbering does not satisfy this requirement, where I want w 1 to be less than equal to w 2 less than equal to w n.

Because if I go like this in this manner, I find that w 5 is not less than w 6 and w 7 is not less than w 8. So, it we want to number in the same fashion left to right bottom to top and still satisfy this requirement. Then what I can do is swap some of these nodes to meet this requirement. For example, I can swap the node 5 and 6 and then I can swap the nodes 7 and 8. If I swap the nodes and then reconstruct my tree, what I will get is as shown here.

(Refer Slide Time: 09:18)



$S_1 \rightarrow 10$
$S_2 \rightarrow 0$
$S_3 \rightarrow 110$ ✓
$S_4 \rightarrow 1110$
$S_5 \rightarrow 1111$

<u>Observation:</u> (sibling property)
(1) $n_{2i-1}$ and $n_{2i}$ are offsprings of the same parent node, or siblings, for $1 \leq i \leq m$
(2) node # for the parent node is greater than $n_{2i-1}$ and $n_{2i}$

I get a new tree after the swapping the nodes, obviously the code word for the symbols. Now, change to obtain the code word for a particular symbol, we have to traverse from the root node to the external node or leave corresponding to the symbol. For example, if I want to get the code word for s 3, then I traverse this way. So, I keep on adding 1 whenever I am in the right path and 0 on the left path. So, I get 1 1 and 0 1 1 and 0. Now, if you look at the 3 2 characteristics can be observed. One is that the node number n 2 i minus 1 and n 2 i are off springs of the same parent node or siblings for i greater than equal to 1 less than equal to m.

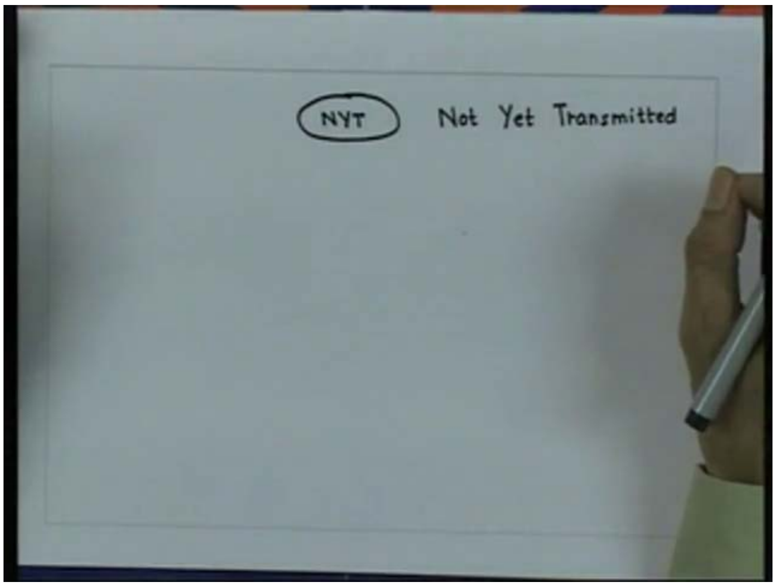
In our case m is equal to 5, so I languish in the range 1 and 5. This property is satisfied another property worth noting is that node for the parent node is always greater than the node for the offspring's n 2 i minus 1 and n 2. Another interesting property of this tree is that the node with higher weights have higher node number. They are closer to the root node. Now, exactly this similar concept which we have discussed for the non adaptive binary Huffman code is extended to the construction of adaptive Huffman code.

The only difference is that the binary tree corresponding to adaptive Huffman code is dynamic. In the sense that both the weights and the node number keep on changing, the weight as defined in the context of adaptive Huffman code has to be modified. Now, so the weights of the external node is simply the number of times the symbol corresponding to that external node or the leaf has been encountered and the weight of each internal node will be the again the sum of the weight of its offspring node. Numbering is done from left to right bottom to top, the root node gets the maximum node number that is 2 m minus 1.

It also satisfies the requirement on the weight of the form w 1 less than equal to w 2 less than equal w 3 less than equal to w 2 m minus 1. This property, this two characteristic which we saw for the binary 3 corresponding to non adaptive Huffman code, these are known are sibling property of Huffman tree or the tree corresponding to the Huffman code. So, similarly, in the case of the binary tree corresponding to the adaptive Huffman code, this sibling property also hold good.

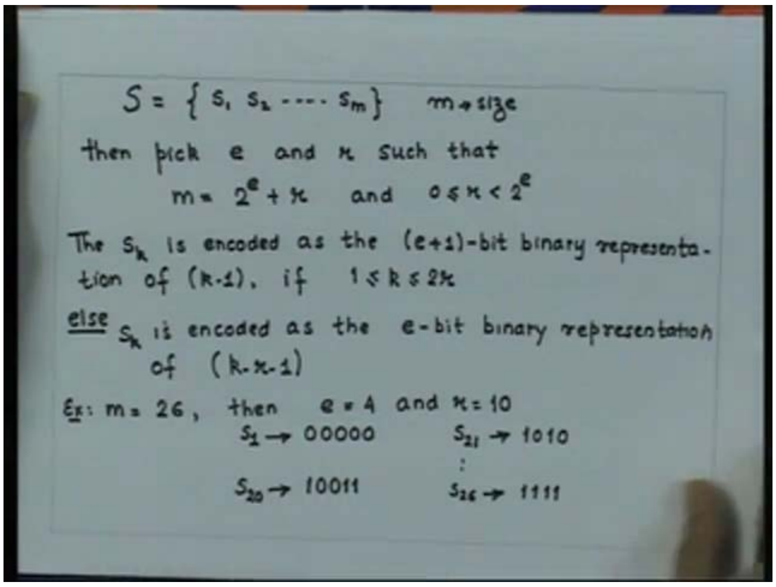
So, in the adaptive Huffman coding procedure, neither the transmitter nor the receiver knows anything about the statistics of the source sequence at the start of transmission. The tree at both the transmitter and receiver consist of a single node and that node corresponds to all symbols not yet transmitted. So, what we do is to start with both at the transmitter and receiver.

(Refer Slide Time: 14:46)


NYT
Not Yet Transmitted

We have a single node and that node is called as NYT, which means not yet transmitted and the weight of this node is 0 as transmission progresses node corresponding to single to symbols transmitted will be added to the tree. The tree is reconfigured using an update procedure, which we will study procedure this update procedure is common to both the encoding and decoding process. Before the beginning of transmission, a fixed code for each symbol is agreed upon between the transmitter and receiver. A simple short code is as follows.

(Refer Slide Time: 16:20)



$S = \{s_1\ s_2 \cdots s_m\}$ $m \rightarrow$ size

then pick $e$ and $r$ such that

$m = 2^e + r$ and $0 \leq r < 2^e$

The $s_k$ is encoded as the $(e+1)$-bit binary representation of $(k-1)$, if $1 \leq k \leq 2r$

else $s_k$ is encoded as the $e$-bit binary representation of $(k-r-1)$

Ex: $m = 26$, then $e = 4$ and $r = 10$

$s_1 \rightarrow 00000$ $s_{21} \rightarrow 1010$

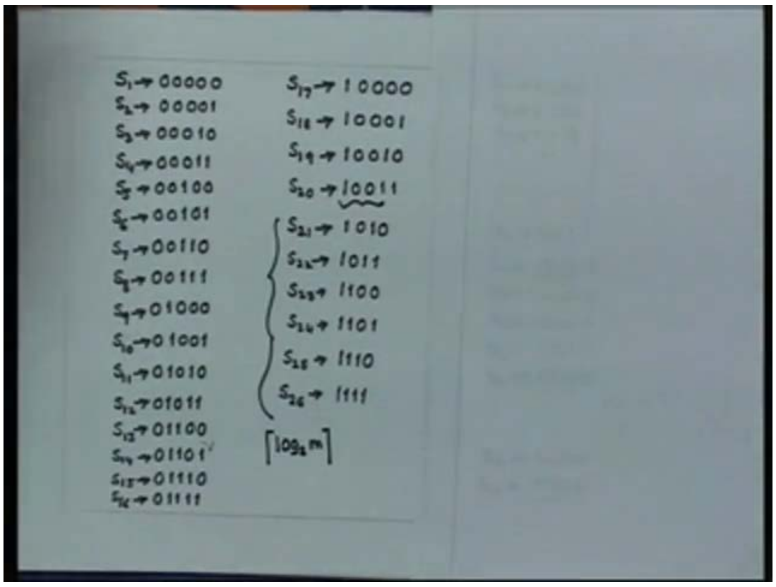
$\vdots$

$s_{20} \rightarrow 10011$ $s_{26} \rightarrow 1111$

If I have a source alphabet consisting of m letters m is the size, then pick the value e and r such that m i S equal to 2 raise to e plus r and r is greater than equal to 0 less than 2 raise to e. Now, the letter S k is encoded as the e plus 1 bit binary representation of k minus 1, if k lies between 1 and 2 r. Else S k is encoded as the e bit binary representation of k minus r minus 1. So, if we have for example, m is equal to 26, for these values of e and r can be calculated as e is equal to 4 and r is equal to 10. Now, so if were to encode the symbol of the letter S 1, since this is the first letter in the alphabet k is equal to 1.

Since, r is equal to 10, 1 is less than 20. Therefore, 1 is encoded as 1 minus 1; that is 0 and 5 digit representation of 0 would be as follows. So, similarly, if I have to encode S 20, which is again less or equal to 20, I have to make a 5 bit representation of 20 minus 1 that is 19. So, that will be 1 1, if i were to encode S 21, 21 is not less than equal to 20 than 21 is encoded as 4 bit representation of 21 minus 10 minus 1. That is 10 and that is equal to 1 0 1 0. Similarly, S 26 would be encoded as 4 bit binary representation of 15. Now, if we look at all the code words for this example from S 1 to S 26, so it could be given as shown below.

(Refer Slide Time: 21:10)



$S_1 \rightarrow 00000$
$S_2 \rightarrow 00001$
$S_3 \rightarrow 00010$
$S_4 \rightarrow 00011$
$S_5 \rightarrow 00100$
$S_6 \rightarrow 00101$
$S_7 \rightarrow 00110$
$S_8 \rightarrow 00111$
$S_9 \rightarrow 01000$
$S_{10} \rightarrow 01001$
$S_{11} \rightarrow 01010$
$S_{12} \rightarrow 01011$
$S_{13} \rightarrow 01100$
$S_{14} \rightarrow 01101$
$S_{15} \rightarrow 01110$
$S_{16} \rightarrow 01111$
$S_{17} \rightarrow 10000$
$S_{18} \rightarrow 10001$
$S_{19} \rightarrow 10010$
$S_{20} \rightarrow \underbrace{1001}1$
$S_{21} \rightarrow 1010$
$S_{22} \rightarrow 1011$
$S_{23} \rightarrow 1100$
$S_{24} \rightarrow 1101$
$S_{25} \rightarrow 1110$
$S_{26} \rightarrow 1111$

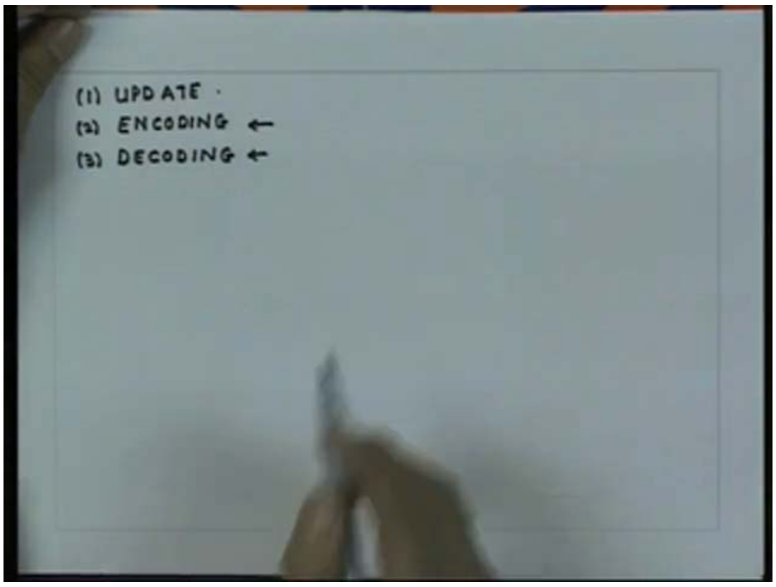
$\lceil \log_2 m \rceil$

One interesting property of this code is that the value of the first four significant bits corresponding to the code word from S 1 to S 20, which are the four most significant bits the value of this from S 1 to S 20 will be less than 10. The code word from S 21 to 26 will go run from the value of that will run from 10 to 30. Now, this property is written like for the decoding procedure in the adaptive Huffman code. Another way of coding this letters of the source could have been used upper flow of log 2 n. In this case, it would have been 5.

So, we would have allotted 5 uniform bits to all the code words from S 1 to S 26, but by adopting the set strategy which we have discussed. We find that the code word for S 21 S 21 look like only 4 bits. So, we get a simple short code in the process, initially all the letters of the code alphabets are placed in the list. That list is called as not yet transmitted list; that list consist of the letters with the predefined code word for those letters, when a symbol is encountered for the first time.

The code for the n y t node is transmitted followed by the fixed code for that symbol from the NYT list a node for the symbol is created. Then that symbol is taken out from the NYT list. Both the transmitter and the receiver starts with the same tree structure the updating procedure used by both transmitter and receiver is identical. Therefore, the encoding and decoding process remains synchronized adaptive Huffman code works based on three procedures.

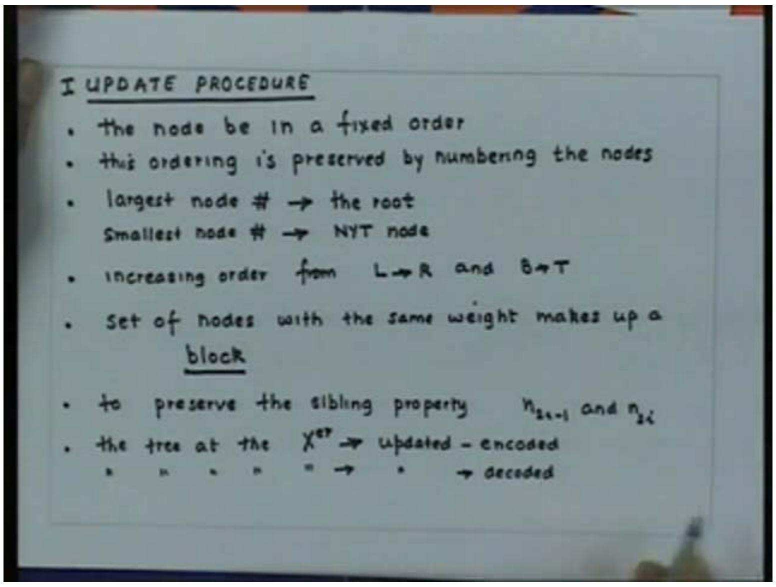
(Refer Slide Time: 24:38)



(1) UPDATE
(2) ENCODING ←
(3) DECODING ←

These three procedures are, update procedure, encoding procedure and decoding procedure, so in order to understand adaptive Huffman code we have to look into all this three procedures. Update procedure is common to both encoding and decoding. So, let us first have a look at the update procedure.

(Refer Slide Time: 25:26)



I UPDATE PROCEDURE

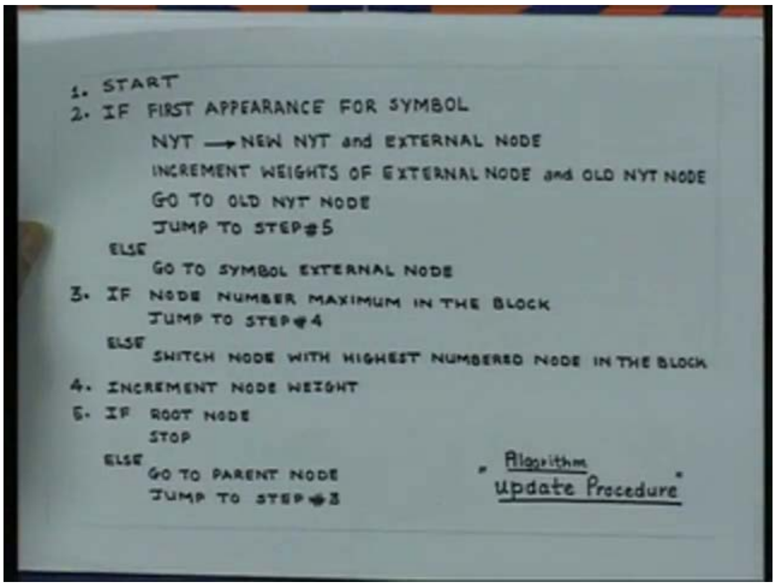
- the node be in a fixed order
- this ordering is preserved by numbering the nodes
- largest node # $\rightarrow$ the root
  Smallest node # $\rightarrow$ NYT node
- increasing order from L $\rightarrow$ R and B $\rightarrow$ T
- Set of nodes with the same weight makes up a **block**
- to preserve the sibling property $n_{2i-1}$ and $n_{2i}$
- the tree at the $X^{er}$ $\rightarrow$ updated - encoded
  " " " " " $\rightarrow$ " $\rightarrow$ decoded

The update procedure requires that the node be in a fixed order and this ordering is preserved by numbering the nodes. The largest number the largest node number is given to the root of the tree and the smallest node number are given to the NYT node. The numbers from the NYT node to the root of the tree are assigned in increasing order from left to right and bottom to top. Now, the set of nodes in the tree with the same weights makes up a block the function of the update procedure is to preserve the sibling property, which we discussed earlier.

That is the node number n 2 i minus 1 and n 2 i are offspring's of the same parent node. The node number the parent node is always greater than the node number corresponding to this offspring. That is the sibling property which is satisfied by the tree corresponding to a Huffman code. So, in order that the update procedure at the transmitter and the receiver both operate with the same information the tree at the transmitter, at the transmitter is updated after each symbol is encoded.

The tree at the receiver is updated after each symbol is decoded. Now, let us look at the pseudo code for the update procedure. So, to start with we have the first appearance for the symbols. Now, the tree at the beginning consists of a single node; that is NYT not yet transmitted node. So, first you check whether in the first appearance of symbols for the symbol if it is the first appearance for the symbol.

(Refer Slide Time: 30:50)



```
1. START
2. IF FIRST APPEARANCE FOR SYMBOL
       NYT → NEW NYT and EXTERNAL NODE
       INCREMENT WEIGHTS OF EXTERNAL NODE and OLD NYT NODE
       GO TO OLD NYT NODE
       JUMP TO STEP#5
   ELSE
       GO TO SYMBOL EXTERNAL NODE
3. IF NODE NUMBER MAXIMUM IN THE BLOCK
       JUMP TO STEP#4
   ELSE
       SWITCH NODE WITH HIGHEST NUMBERED NODE IN THE BLOCK
4. INCREMENT NODE WEIGHT
5. IF ROOT NODE
       STOP
   ELSE
       GO TO PARENT NODE
       JUMP TO STEP#3
```

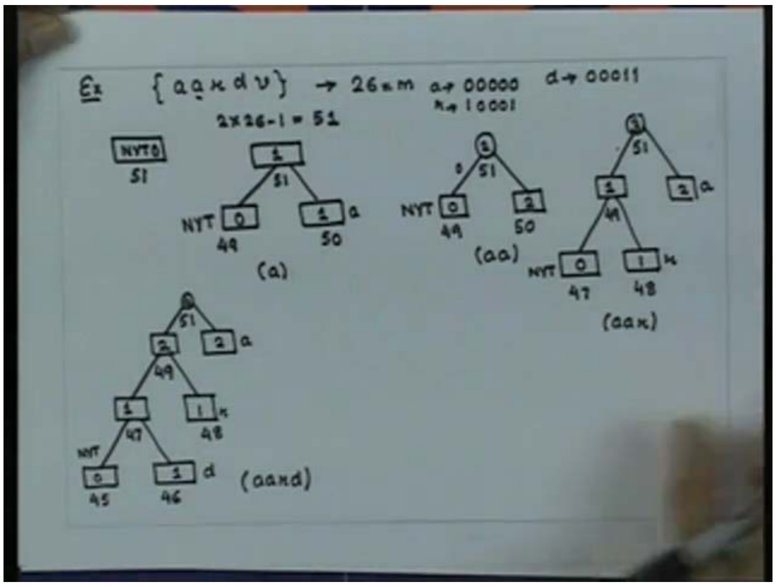
"Algorithm
Update Procedure"

Then NYT node degenerates into new NYT node and external node corresponding to that symbol. Then you increment the weights of external nodes and the old NYT node. The next step is to go to the old NYT node and then jump to check whether it is the root node if the appearance for the symbol is not for the first time. Then go the symbol external node after you reach the external node check the node number of that external node. If the node number is maximum in the block, then jump to step four, but if the node number of that external node is not maximum in the block.

Then switch that node with the highest number node in the block as long as the node with the higher number is not the parent node of the node being filled. After the switching of the node has taken place, it is permitted. Then you increment the node weight check, whether this node is the root node, if it is the root node you stop else go to the parent node of this node, whose weight has been updated.

Repeat the process by going to step three to find out if the node number of that parent node is maximum. If again if it is not maximum, then since it is permanent permitted, otherwise increase the node weight. You keep on iterating this process, unless you reach to the root node, then you stop. Let us look at this update procedure with the help of an example.

(Refer Slide Time: 33:44)



So, suppose we are encoding the message a a r d v, then our alphabet consists of the 26 lower case letters of the English alphabet. Now, the updating procedure is as follows, we begin with only the NYT node. The total number of nodes in this tree will be 2 into 26 minus 1; that is 51. So, we start numbering backwards from 51 with the number of the root node being 51. The first letter to be transmitted is a now as a does not exist in this tree. So, start with this node weight is 0 and the number is 51, so we start with the transmission of the letter a. As a does not exist in the tree, we send a binary code 0 0 0 0 0 for a, and then add a to this tree.

Now, to add a to this tree, this NYT node to start with gives birth to a new NYT node and a terminal node corresponding to letter a. So, this is my old NYT with both a new NYT and a terminal node and the weight of the terminal node will be higher than the NYT node. So, we assign the number 49 to the NYT node and 50 to the terminal node, corresponding to letter a. This weight will be 1, this will be 0 and this weight will be sum of the weight of the occurrence that is 0 plus 1 and the node number here will be 51. Now, the second letter to be transmitted is also a. This time the transmitted code is 1, because a is already existing in this tree.

The node corresponding to a has the highest number that is 50, if we do not consider its parent. So, we do not need to swap this node, so we just increment the weight of this node it becomes 2. We increment the weight of the root node also that will become 2 and

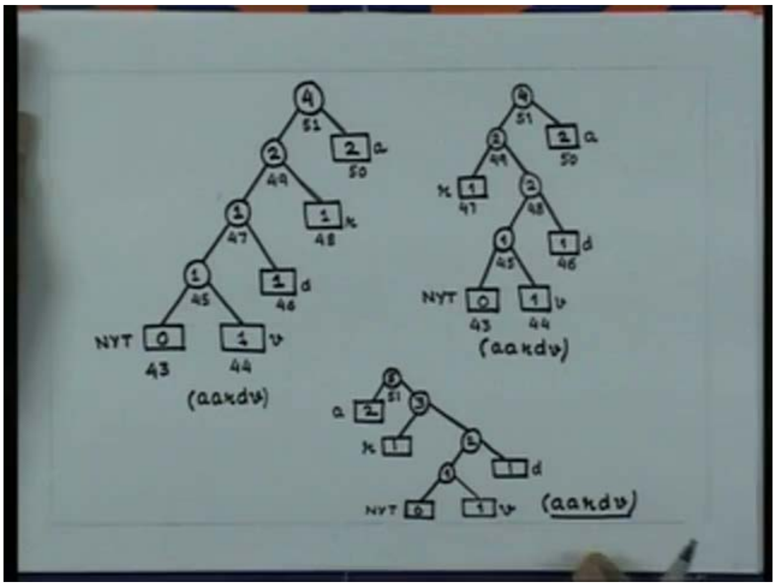
NYT here is 0, 49, 50, 51. So, this a tree on the seat of a on transmission or receipt and this is a tree on the transmission or receipt of a. Now, the next letter to be transmitted is r, this letter does not have a corresponding node on the tree. So, we send the code word for the NYT which is 0, followed by the index of r. So, if we do that this will degenerate into a new NYT node and a terminal node for r.

This is your r a. Now, so we will transmit 0 and the code word for r that is code word for r is 1 0 0 0 1. We will look at the encoding procedure little later on. So, I get this new NYT, I renumber this 49. So, this becomes 47 48, so the weight is 1, the weight is here 0. I increase the weight of my old NYT, it is 1. Now, this has the highest number in its block, so I do not have to do anything. I go to the parent node of it parent node is this that has the highest number 51. So, I increment it by 1 come back and check whether the root node, it is a root node. I stop here so this is the tree, which I get at the end of transmission r.

Now, when I transmit the letter d again the transmission of d is for the first time. So, we have to send the NYT node to send NYT node. We have to send the code as 0 0 followed by the code word for d. The code word for d is 0 0 0 1 1. So, the next step, this will, this node 47 will give birth to a new NYT and a terminal node. This terminal node corresponds to d, if I update the weight of this, this will be 45. This will be 46, then the weight of the old NYT is 1. I look at the parent of this the parent of this is this. Now, that has the highest number in the block, so I just increment the weight that becomes 2. So, this remains as it is this again as it becomes 2.

It has the highest number in the block, so I do not require anything to be done. Now, I go to the root of parent of this node; that is this node. So, 3 it becomes 4 and this number are as follows. So, this a tree which I get it on the receipt of the symbol d. Now, the next transmission is v. Now, the transmission of v, v is not existing in this tree. So, what happens is that we have to send the code word for NYT b which is 0 0 0 followed by the code word for v. Now, so this node will give rise to a new NYT plus a terminal node corresponding to v. So, let us do that.

(Refer Slide Time: 43:39)



So, this node, now will give rise to a new NYT and a terminal node for this. So, I will have the numbering as 43, 44, 45. This is my NYT. This node corresponds to v, which is v 1, this is 0 1 1 2, 41, 47. So, this gets degenerated that was the old one into new NYT and b. I check the weight for this old NYT as one move here to its parent. If you look at the parent of this node, it does not have the highest number in the block.

The highest number in the block is being possessed by the node number 48. So, I have to swap it and then increment. So, if I do that what I get is… So, I have swapped it. If I swap this node and after I have done the swapping, I increment the weight of this node. So, this becomes this is r, this is a weight that is equal to 1 and this becomes 47. This becomes 48, this becomes 2. Now, after I have incremented the width, I look at the parent node of this. The parent node of this is, this now again this node does not have the highest number in the block.

The highest number in the block is possessed by the node number 50. So, I have to swap this node with 50. Then increment the weight, so if I swap the thing 49 with 50, then what I get is I have shown here. I swap the 1, 49 with 50 and then increment the weight of it. So, this becomes 3 after increment, this I go to the parent node of this. So, the parent node of that would be the root node see it S I the highest number. If it is the highest number, so I increment the weight of this node that becomes 5. Then check whether this is the root node, since this the root node I stop.

So, then finally the tree which I get is shown here on the transmission of the receipt of the sequence of the symbols corresponding to a a r d b. Now, if the next symbol to be transmitted or to receive is a, then we send the code as 1. Then look for this whether it has got the highest number, it has got the highest number. In our case this will be 49, this has the highest number. So, we do not do ant swapping and we just increment the weight of this. So, this would become 3 and then check for the parent node of this node. That would be the root node.

Again this is the highest number, so we just increment that and check with the root node. Since it is the root node, we will stop. So, in the next alphabet to be transmitted the next, sorry the next letter which you think to be transmitted was a. Then the next; that is stage, the tree would look exactly like this with the weight of a increased to 3 and this to 10. Now, this updating procedure, which we have studied with the help of my example is common to both encoding and decoding procedure. Now, we will have a look at the encoding process and decoding process, which is the part of the adaptive Huffman code in the next class.