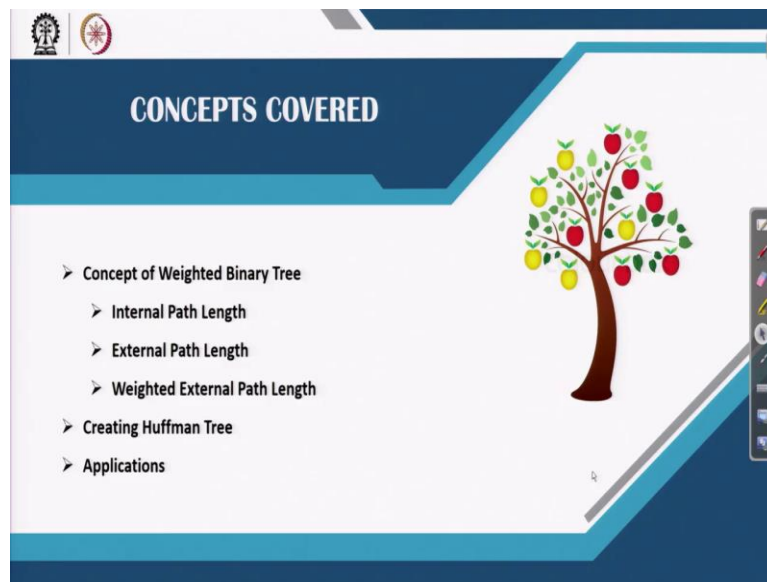


Data Structures and Algorithms Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 34
Huffman Tree

We are discussing Binary Tree and we have discussed several important Binary Trees. Last, in the last lecture we have discussed about Heap Tree. Today, we will discuss another very important Binary Tree. It has again many applications, particularly in the field of communication. Earlier, this Huffman Tree was extensively used. Now it is also used. This Tree is in the application of compression, text compression, data compression and message encoding, encryption, many applications are there. So let us study the Huffman Tree in this lecture.

(Refer Slide Time: 01:14)



So, this lecture includes few basic concepts prior to study the Huffman Tree. That means it is the concept of Weighted Binary Trees followed in the Huffman Tree. First we will see exactly when a Binary Tree is termed as Weighted Binary Tree. And for Weighted Binary Tree, there are two important terminologies are there. This is called path length.

There are two types of path lengths. It is called the internal path length and external path length. So, we will study first Weighted Binary Tree and for a Weighted Binary Tree, the internal path and external path length. Now, there is an another definition of path length it is called the

weighted external path length, so we have to discuss about weighted. Now, this discussion will make sense to understand Heap, Huffman Tree finally.

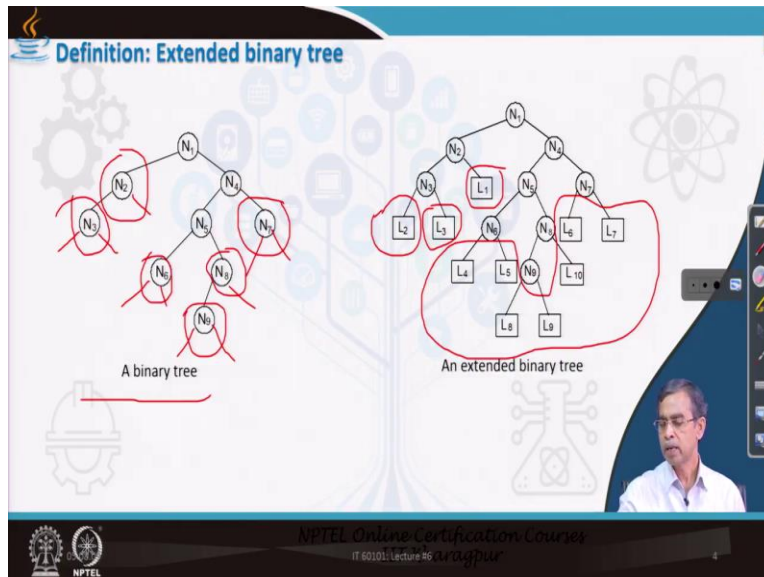
So Actually Huffman Tree is basically is a Binary Tree, which is having minimum value of weighted external path length. So now, we will, now we shall learn about how a Huffman Tree,, that is a Tree, a Binary Tree with minimum weighted external Path length can be created? And finally this lecture slides will be concluded giving an application of Huffman Tree. So, this is agenda that we have planned in this (cou), in this lectures.

(Refer Slide Time: 2:53)



So now, let us first discuss about Weighted Binary Tree. We know exactly different type of Tree, Binary Search Tree, we know Heap Tree. We know other Tree like say Expression Tree in some sense, right? Now we will discuss about Weighted Binary Tree. So, Huffman Tree is basically, essentially Weighted Binary Tree but in addition to, it has some extra properties. Now, let us first discuss about Weighted Binary Tree concepts.

(Refer Slide Time: 3:20)



Now here, in this picture, first let us consider, this is the usual concept of Binary Tree. So, a Binary Tree is a collection of nodes and each node has maximum two children that is the concept. Now, it can be any Tree, Binary Search Tree or Heap Tree, all are Binary Tree actually, right? Okay, so this is an example of Binary Tree with n number of nodes in it. I should not say that whether it is a Binary Search Tree or it is Heap Tree or I am not telling like this. It is just simply a Binary Tree. Okay, fine?

So, it is ordinary Binary Tree. The usual Binary Tree. Now, there are two types of nodes as you know. The nodes is basically some nodes is called the leaf nodes in Binary Tree. The nodes are leaf nodes means the nodes which does not have any children.

As we see in this example, so this is a leaf node, this is leaf node, this is leaf node and this is leaf node. Now, after these leaf nodes are there, there are few nodes which basically has the single child. Now, in this example, as you see, this is basically the node which has the single children and this is the node has the single children.

Now, so the nodes are single children or leaf node because their link field, either all the link fields or any one link fields are empty. For example, this N_3 , it has two link field, the left and right. It is basically empty. It has right link is empty. It has both left link is empty. It is empty

and it is empty and it is empty. Now, what is the concept is that, all the empty link can be filled with address to a dummy node.

So this dummy node is called the external node. Now, for example, as we see, N2 has one dummy node. This is called the L1. It is L2 and L3 for the another leaf node, and so these are, so these are the different dummy nodes we have already added. So, dummy node actually you see, dummy node is just like header node that we have already learned about while we were discussion linked list.

So it is a header node is a dummy node because this node does not store any value in it. All the other, on the other hand, any other nodes other than this dummy node store some value in it. So, it is basically the matter. Now, again you can ask. You may have some curiosity, why we are going to add dummy nodes? I will tell, its application while we learn about the dummy nodes matters a lot actually.

Anyway, let us first learn about internal node. Internal nodes are basically not dummy node. Internal nodes are nodes which contain some what is called the value in it, some actual value in it. On the other hand, the external nodes, they are dummy nodes, they do not store any value in it at the moment.

Anyway, so, we can discriminate two types of node in a Binary Tree namely internal node and external nodes. Internal nodes are can be either degree two or degree one or degree zero. On the other hand, all dummy nodes, all external nodes, they are of degree zero because they do not point to any nodes. That is right. Fine. Now let us start about other concept that is required in order to have the full understanding of Huffman Tree. So, so far we have learned about internal node and external nodes.

(Refer Slide Time: 7:23)

Definition: Path length

Path length: Path length of a node in a tree is defined as the number of edges that has to be traversed from the node to the root node.

Path length of $N_1 = 0$
Path length of $N_2 = 4$
Path length of $L_1 = 4$
Path length of $L_8 = 5$

The slide features a tree diagram with nodes labeled N_1 through N_9 and L_1 through L_8 . Node N_1 is the root. Nodes $N_2, N_3, N_4, N_5, N_6, N_7, N_8, N_9$ are internal nodes, and nodes $L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8$ are leaf nodes. A red path is highlighted from node N_6 up to the root N_1 , passing through N_5, N_4, N_3, N_2 . The slide also includes a small video inset of a man in the bottom right corner and the NPTEL logo at the bottom.

Now, here is the idea about, okay so, internal node and external node, we have learned about. Now we have to discuss about path length. So, path length, basically, (sta) for every node, we can calculate the path length. Path length is basically starting from that node to the root node, so how many node you can visit?

Now here, for example, the path length of this node N_6 . So, we have to visit this, this, this so path length is 4. Now, here, N_9 for example. N_9 , right? So, for this N_9 , starting from the N_9 and this one and this one, so here if we, so N_9 is the node here. We follow this, this, this, so how many node we have visited excluding the N_9 of course? So 1, 2, 3, 4 or actually if you see, how many edges are there starting from the root node to the node whose path length is to be calculated?

So here, this is the number this is the edge 1, 2, 3 and 4, so number of edges number of edges can be calculated in little bit different way starting from this node to the root node, how many nodes are there? So, 1, 2, 3, 4, 5. 5 nodes are there so 5 minus 1 is basically the number of edges. It is always true, for example, L_8 . So, number of edges are actually 1, 2, 3, 4, 5 and the number of nodes that you can traverse from that node to the root node is basically 6 so that is why it is there.

Now I have listed here as you can see in this calculation, the path length few nodes. So this way every, for each node, we can calculate the path length of that node. Whether this node is internal node or external node, it does not matter. The path length for every node can be calculated by using this formula or this method actually. So this concept is fine.

(Refer Slide Time: 9:53)

Definition: Internal and external path lengths

Internal path length: Internal path length (let it be denoted as I) of a binary tree can be defined analogously as the sum of path lengths of all internal nodes in the tree.

$N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8, N_9$

$$I = 0 + 1 + 2 + 1 + 2 + 3 + 2 + 3 + 4 = 18$$

External path length: External path length (let it be denoted as E) of a binary tree is defined as the sum of all path lengths, summed over each path from the root node of the tree to an external node.

$L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8, L_9, L_{10}$

$$E = 2 + 3 + 3 + 4 + 4 + 3 + 3 + 5 + 5 + 4 = 36$$

The diagram shows a binary tree with root node N_1 . Internal nodes are N_1 through N_9 . External nodes are L_1 through L_{10} . The tree structure is as follows: N_1 is the root, with children N_2 and N_3 . N_2 has children N_4 and L_1 . N_3 has children N_5 and N_6 . N_4 has children L_2 and L_3 . N_5 has children N_7 and N_8 . N_6 has children L_4 and L_5 . N_7 has children N_9 and L_6 . N_8 has children L_7 and L_8 . N_9 has children L_9 and L_{10} .

Now, let us come to discussion about internal path length and external path length. As a name implies, you can understand what it does mean? Internal path length is basically sum of all path lengths of all internal nodes. Now, this way, here if you see, all internal nodes are N_1, N_2, N_3 like N_9 , so now, this is basically the path length of all internal nodes, sum of the path length of internal nodes.

So, this is the, this formula, this basically gives you the calculation of internal path lengths. For example, for this Tree, the internal path lengths is 18. Similarly, the external path lengths also can be calculated. The external path length is basically sum of the path length of all external nodes. As we can see in this Tree, L_1 to L_{10} are the external nodes and this basically sum of the path lengths of all external nodes, this gives to 36.

So, these are the concept or basically they are matrix, for the measurement of internal path length and external path length. So, we have learned about internal node, then external nodes, then path length, internal path length and external path lengths.

(Refer Slide Time: 11:29)

The slide features a title "Properties: Relation between internal and external path lengths" at the top. Below the title, a text box states: "In a binary tree with n internal nodes, if I denotes the internal path length, then external path length, $E = I + 2n$." To the right is a binary tree diagram with 9 internal nodes labeled N_0 through N_8 and 12 external leaf nodes labeled L_1 through L_{12} . On the left side of the slide, there are three lines of text: " $n = 9$ ✓", " $I = 18$ ", and " $E = 18 + 2 \times 9 = 36$ ". The slide also includes a small video inset of a person in the bottom right corner and the NPTEL logo at the bottom left.

Now, we have to discuss about few more concept here. That concept is basically very important. Here, it is basically one theorem. For any tree, this is valid actually. This theorem is valid. This theorem is that, if I is the internal path length and E is the external path length for a given Binary Tree with N number of nodes, then there is a very nice relation between the external path length and internal path length. So, this relation says that if E is external path length and I is the internal path length then E is equals to I plus $2n$ where n is the number of nodes.

Now, we can verify this formula for any Binary Tree adding external nodes into it and then verify. It is always valid. Now here, for an example, for in this Tree, as you see, total number of internal nodes or number of nodes actually, it is in this Tree is 9. So, total number of nodes is 9 and we have calculated internal path length for this Binary Tree is 18 and now you see, external path length also we have calculated 36.

Now, you can check that this E is equals to this also. That mean E is equals to I plus $2n$ is valid. Now this is an example that I have given. You can take any arbitrary Binary Tree, add external nodes into it, calculate the path lengths, internal as well as external and then verify this formula. So, this is an important theorem actually. The theorem gives us the relation between internal path lengths and external path lengths of a Binary Tree.

(Refer Slide Time: 13:25)

Definition: Weighted binary tree

Suppose, all internal nodes have unit values but each of the external nodes are assigned a (non-negative) number called *weights*.

The external path length with these weights are called *external weighted path length* (or, simply *weighted path length*) and the corresponding binary tree is termed as *extended weighted binary tree* (or, simply *weighted binary tree*).

if W_i is a weight of an external node n_i and its path length is L_i , then, weighted path length, P is:

$$P = \sum_{i=1}^n W_i L_i$$

NPTEL Online Certification Courses
IT 60101: Lecture #6: *Tree*

Now, let us come to the next discussion about weighted Binary Tree. Now, in case of the previous discussion, we have considered the (dummy) dummy node or external nodes where we didn't consider any weight. Now, we can do something again, each external nodes, let us put some weight of it. That weight can be any value, any number. Floating point value may be integer or whatever it is there but it is a number. Let it be number only.

So, all external path lengths are assigned it weights. That weight you can put in any order whatever it is there. So, any random arbitrary value can be given in the weight. So if there is an external node E_1 , let its weight be W_1 and like like, so every nodes are with weights. Now, if it is like this then weight of that external node is multiplied by its external path length is called the weighted path length for that node.

Now, taking the sum of all the weighted path lengths corresponding to all external nodes gives you called the external weighted path length. Now, we have learned about internal path length, external path length. Now, we introduce another new concept, it is called the external weighted path lengths. So, external weighted path length is the sum of the weighted path lengths. Weighted path lengths means it is basically weighted path length of all external nodes.

So, this basically can be accessed by this formula. This formula says that if W_i is the weight of i 'th external nodes and L_i is the external path length of i 'th external nodes then sum of all weights

and its weighted path lengths is basically gives you the weighted external path length. So, this is basically called external weighted path length or simply it is called weighted path length.

Now, so we have discussed about that a Binary Tree, if it includes the external nodes with weight then we can say that it is an extended Binary Tree or simply we can say that extended weighted Binary Tree or more simply we can say that Weighted Binary Tree. Now, this is the concept of Binary Tree a special kind of Binary come into the picture. Again I repeat it is a Binary Tree. It is not necessary to be complete or full Binary or not necessary to be in the Binary Search or Heap or nothing, it is just Binary Tree, but only thing is that, it has the external nodes. All external nodes are with some weights and it is basically the weighted Binary Tree.

(Refer Slide Time: 16:41)

Weighted path length: Illustration

$$P = \sum_{i=1}^n W_i L_i$$

$$P = WL_1 + WL_2 + WL_3 + WL_4 + WL_5 + WL_6 + WL_7$$

$$= 2 \times 2 + 5 \times 2 + 3 \times 4 + 4 \times 4 + 1 \times 3 + 7 \times 3 + 6 \times 3$$

$$= 4 + 10 + 12 + 16 + 3 + 21 + 18$$

$$= 84$$

NPTEL Online Certification Course
IT 60101 Lecture 46 Aravind

Weighted path length: Illustration

$$P = \sum_{i=1}^n W_i L_i$$

$$P = W_1 L_1 + W_2 L_2 + W_3 L_3 + W_4 L_4 + W_5 L_5 + W_6 L_6 + W_7 L_7$$

$$= 2 \times 2 + 5 \times 2 + 3 \times 4 + 4 \times 4 + 1 \times 3 + 7 \times 3 + 6 \times 3$$

$$= 4 + 10 + 12 + 16 + 3 + 21 + 18$$

$$= 84$$

NPTEL Online Certification Courses
© 2020. Created by NPTEL

Now, let us consider some examples to understand this concept better. I am trying to explain you the weighted Binary Tree here, so let us consider these are the some nodes, internal nodes with any value. I do not bother about it. They are not important actually. Only important is that external nodes here. Now, these are the basically as you see in this example, these are all external nodes and for each external node, there is a weight, so it is at the weights. Weights. So there are total seven external nodes are there. Seven weights are involved.

Again new node, internal nodes whatever the value they have, it is really does not matter, we will not bother about it. We will, it matters only the weights of all external nodes. These weights are arbitrary or according to some right application you can assign them. We will see exactly how the weight can be assigned to the external nodes later on when we will discuss the application.

Now once the weights are known to each external nodes, then we are not ready to calculate weighted path length. So, weighted path length means, weight multiplied by the path length, so W_1 into 2. Here, W_2 into 2. W_3 is 2 into 2, 5 into 2. So here, you see, for this Binary Tree or Weighted Binary Tree rather, for this Weighted Binary Tree, we have calculated this is the weighted path length.

So P is the weighted path length. As I said, it is the formula that we have to use here in for this and this is the total calculation and it says that for this Binary Tree, the weighted path length is

84. Now, this concept is important to understand the concept of Huffman Tree. So we have learned about internal nodes. We have learned about external nodes.

We have learned about path lengths, internal path lengths, external path lengths. We have learned about weight about internal nodes and then we have learned about the weighted path lengths. It is basically this formula. Now let us learn about it actually the problem. What exactly the problem that we can think about with his Binary Tree. That means weighted Binary Tree.

(Refer Slide Time: 20:29)

Problem: Building a weighted binary tree with minimum weighted path length

Given a set of weights (for the external nodes), how a weighted binary tree can be constructed, so that it has minimum weighted path length

(a) T_1 $P(T_1) = 38$

(b) T_2 $P(T_2) = 44$

(c) T_3 $P(T_3) = 34$

NPTEL Online Certification Courses
IT 60101: Lecture #6: Huffman Coding

Here is the problem. Problem is that, given some weights to all external nodes, we have to create a Weighted Binary Tree so that its weighted path length is minimum. Now this is a problem if I give it to you then how you can solve this problem? What I, I repeat again, there are suppose in this example seven external nodes are there and all external nodes are with some weights.

So what I can say that seven weights are given to you. Now we can have many Binary Trees with these seven weights as the external nodes in fact. Out of these many Binary Trees, rather Weighted Binary Trees, we have to have one Binary Tree, Weighted Binary Tree, which is having minimum weighted path length.

Now, let us consider an example so that we can understand about it. Now suppose the values of the weights are 2, 3, 5, 9, so only four external nodes are possible. Now, with these four external nodes, there are many Binary Trees possible. That is weighted Binary Trees possible, so I have

given such three such Binary Tree in this slide. So this is the 1, now let it be T1. So, this is the first weighted Binary Tree T1 and what is the path length of this Weighted Binary Tree, weighted path length, this 38.

This is the another Binary Tree where the weights are basically like this. External nodes are arranged in this way. This is the say Weighted Binary Tree T2 and its path is 44. This is another Tree. Here it is like, here if you see this Tree and this Tree has the same structure but all external nodes are connected in a different manner and then T3 is the another Weighted Binary Tree whose path length is 34.

Now what we can check there, out of these three Binary Tree where all of them have the same weighted external nodes. But they are having different weighted Binary Tree and corresponding different weighted Binary Tree, they are having different weighted path length and out of which, this is the Tree which has the minimum weighted path length out of these three.

Now, question again is that, is there any more Tree possible with the same? Yes. There are many more Weighted Binary Tree with these 4 weights possible. So, how many? There is a huge number. There is a calculation that how many different weighted Binary Trees possible with this one, there is a formula is there. Anyway but whatever it is, it is very large number actually and this if the number of weights increases then number of Weighted Binary Tree that is possible with so many weights are also highly increased and therefore, finding out which is the minimum Weighted Binary Tree with minimum weighted path length is almost impossible to solve problem in real time.

So, this is a very, is a problem and is a is a very nice problem for the computer scientist in fact how to solve this problem. So, here the problem is that given a set of weight, you have to create one Weighted Binary Tree so that this weighted Binary Tree has the minimum weighted path length. This is the problem and this is a very nice problem. The problem was solved by the famous scientist, his name is Huffman and that is why this Tree, the Tree which basically can be formed according to the Huffman's Algorithm is called the Huffman Tree.

(Refer Slide Time: 23:32)

The slide features a central title "Huffman Algorithm" in blue text. To the left is an icon of a coffee cup with steam. The background is a light blue tree with various icons at its branches. The slide is part of an NPTEL presentation, as indicated by the logo and text at the bottom.

NPTEL Online Certification Courses
IIT Kharagpur

The slide is titled "Huffman algorithm" in blue text. It contains a list of three bullet points in a light orange box. The background is a light blue tree with various icons at its branches. The slide is part of an NPTEL presentation, as indicated by the logo and text at the bottom.

- An elegant solution to the problem of finding a binary tree with minimum weighted external path length was given by D. Huffman.
- The tree that can be constructed with the Huffman's algorithm is guaranteed to be a minimum weighted binary tree and to honour the inventor, such trees are alternatively termed as *Huffman tree*.
- The algorithm is an iterative method with greedy strategy.

NPTEL Online Certification Courses
IT 60101: Lecture #6 | Kharagpur

Now, we will see exactly how the Huffman Tree can be obtained. Now, Huffman Tree is followed by an Algorithm Actually and this Algorithm is a famous algorithm, it is called the Huffman Algorithm. This Algorithm is in fact very simple Algorithm. Now if I give you the problem to find your own algorithm, you may take many days to find the algorithm.

But anyway, now, let us see first Huffman Algorithm. This is the Huffman Algorithm proposed in very long time in around 1950 years so he proposed this algorithm, this algorithm based on the paradigm called the Greedy. Greedy means so try to find the shortest weighted path, Weighted Binary Tree whatever the way.

Use the shortest Weighted Binary to find another shortest Binary Tree and finally the shortest Binary Tree will with all nodes will be considered so this is called the greedy strategies. Now let us understand that is the greedy strategies that is followed in Huffman Algorithm. So, in the next few slides, we will try to understand about the Huffman Algorithm.

(Refer Slide Time: 24:41 }

The algorithm

Input:
 n weights W_1, W_2, \dots, W_n .

Steps:

1. Sort the weights in ascending order.
2. Obtain a sub-tree with two minimum weights as the weights of external nodes.
3. Include the weighted path length of the sub-tree so obtained into the list of weights.
4. Repeat the procedure until the list contains single weight.

NPTEL Online Certification Courses
IT 60101: Lecture #6: Dr. Duggir

So, for this algorithm, the input is the set of weights. The weights can be distinct or it all can be whatever. There maybe two weights maybe same value absolutely not an issue. These are weights in any value, arbitrary values. So, all set of weights. This algorithm consider n number of weights are input, so with the same number of weights, we have to build a Weighted Binary Tree and that Weighted Binary Tree should have the minimum weighted path length.

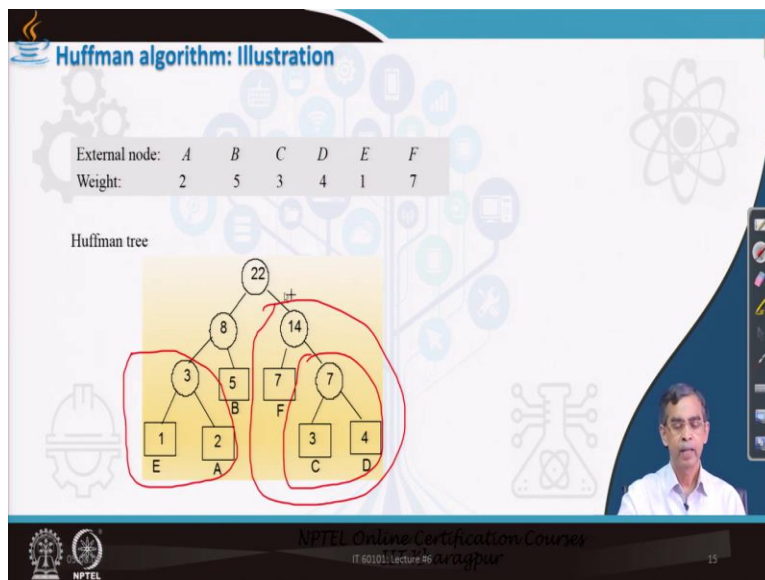
So, the algorithm is very simple. It has only four step. The first step is, you first have all the weights in sorted order, ascending order so this is the first stage that we have mentioned here. Sort the weights in ascending order from smaller value to the larger value. Then, first create a

subtree with two minimum weights. So, take any two minimum weights and then create a subtree. In this subtree, it has only one internal node and two external nodes and this...

Then the weighted path length of this subtree can be stored into its internal node and use this internal node into the list of weights. So this basically add into these weights. I will discuss this example with an example so that you can understand. So, this is the idea is that take two minimum weights and then create the subtree. Add the subtree into the list of weights, so this is basically this one.

Then repeat the same procedure until the all weights are included into the Tree and this will completes the Huffman Tree building. Now better we can explain these things with an example and then we will, if required we can come back to this step again.

(Refer Slide Time: 26:54)



And here is an example. These are the weights. These are the basically name of the leaf node or external node you can say. External nodes so like that, like A, B, C, D, E, F are the six external nodes and they are assigned with weights in any order. Then Huffman Tree that can be obtained given these are the weights are like this. So, E should be placed here, A, B, F and everything everything so these are the basically you see weights are arranged as an external node into a Binary Tree, weighted Binary Tree and this basically is an example of weighted Binary Tree.

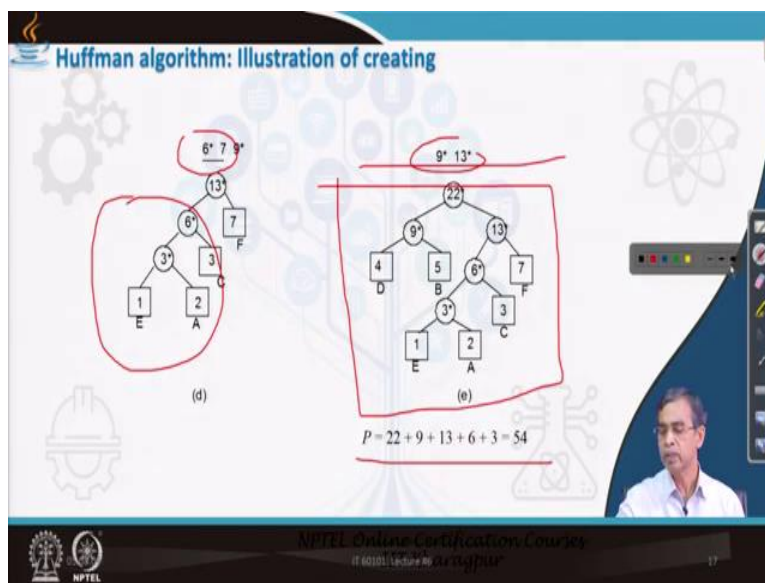
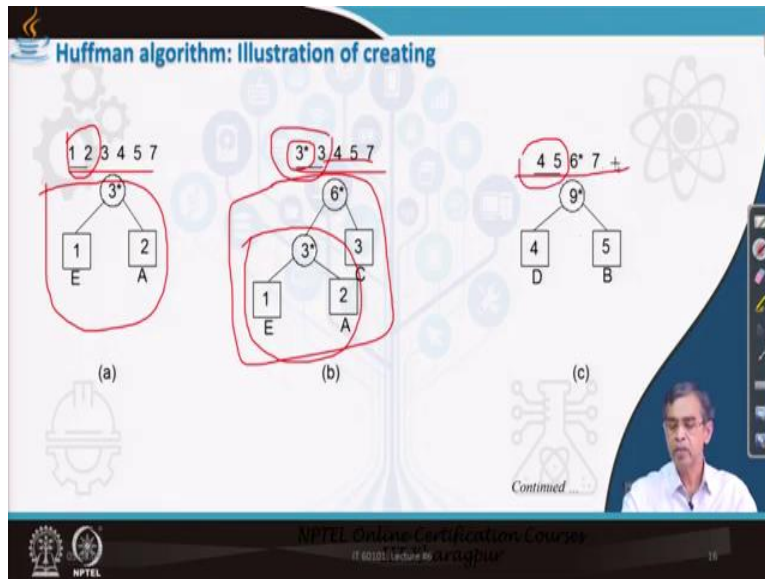
Now here, you can note see what is the value that internal nodes hold good. Internal nodes means, I am telling you again, let us see. Say suppose this is the one subtree. In fact, each subtree if you consider is a Weighted Binary Tree. Now this is the subtree which contains only one internal nodes and two what is called the external nodes with weight. So, this is a Weighted Binary Tree. Now, this three is basically is the weighted path length because if you see weight into path length, weight into path length, sum of weight path length so this three.

Now if you again consider, this is another Weighted Binary Tree, this basically the path length, weighted path length of this Binary Tree. Now this is another subtree. It is also another weighted. So this basically path length, weighted path length of the weighted Binary Tree. And continuing these things, as you see this whole is basically Weighted Binary Tree and then 22 is basically sum.

Now, what is the weighted path lengths of the entire Tree that can be obtained by summing up of this 3 plus 8, 22 plus 14, 17 so sum of all internal nodes can give rise to give the total weighted path lengths of this Weighted Binary thing. For example. 3 plus 8 plus 28 plus 14 plus 7 so it is around 51 like so 51 is a not 51, 30 and then 40, 54. So, 54 is the weighted path length, total weighted path length of this Binary Tree, Weighted Binary Tree.

Okay, fine. So now, let us see how whether I do not know, whether this is the Weighted Binary Tree with minimum weighted path length or not. Anyway, first, this is another question and second is that how we got this Tree? Whether it is a randomly we build the Tree or we follow any algorithm? So, definitely we should follow Huffman Algorithm to have the Weighted Binary Tree with minimum path length. Let us first proceed it.

(Refer Slide Time: 30:13)



Here is the procedure that you can follow. This is Huffman Algorithm. First, we just see we ordered the element into the sorted order so we just in sorted order. Then we create the subtree with the two minimum weights. So, in this case, two minimum weights this one and this one 3 star. So 3 is basically is the new weights so for this Weighted Binary Tree is concerned, it is sub-weighted Binary Tree. With this, 3 star should be added into this list.

So, adding 3 star into this. Then weights are like this. This is basically the weights from the subtree and these are the weights of the other remaining external nodes. Again we repeat the two

minimum weights in this case, so taking these two and then we can create another Binary Tree, Weighted Binary Tree where this is the path that we have already done and this 3 is added so this is the this one 6 is new result so this 6 will be added here.

6 is added here. Again sort, then 4, 5. 4, 5, 9. This 9 will be added and this will continue. This will continue. And so this will continue 6, 7, 9. Then taking 6 and 7. 6 is the previous node we have created. 7 is the new and then 13 is created, 13 is added. 9 star 13 is so 22 so this basically finish because this is now complete. There is no more elements are there and this basically creates the Huffman Tree according to Huffman Algorithm and the total path length that we can calculate 54.

Now, previously I showed an example of a Binary Tree where I calculated path length with same set of external nodes and 54. This means that this is on according to Huffman Algorithm and earlier also, according to algorithm Huffman Algorithm, but two nodes, two Trees are not same. So, this means that Huffman algorithm is not necessary to give only unique Weighted Binary Tree.

This is because if two weights are of same value then we can take this one and that one and then it is not unique so that is why, it is not necessary that Huffman Tree that you will get it unique like, it is but all are with minimum weighted path length actually. It is not unique but they can give you, guaranteed to give you the minimum value so Huffman Algorithm is guaranteed to give you the minimum value always. So, we have learned about Huffman Tree Algorithm and we have discussed that it is not necessary to be unique as here the two Trees are not unique.

(Refer Slide Time: 33:21)

Huffman algorithm

Algorithm BuildHuffmanTree (W)

1. For $i = 1$ to N do // Read the external nodes and initialize the PARRAY
2. $ptr \leftarrow \text{GetNode}(\text{NODE})$
3. $ptr \rightarrow \text{WEIGHT} = W[i]$ // To read the weight for the external node
4. $ptr \rightarrow \text{LCHILD} = \text{NULL}$ // Link fields of external nodes are empty
5. $ptr \rightarrow \text{RCHILD} = \text{NULL}$
6. $\text{PARRAY}[i] = ptr$ // Store the node into the PARRAY
7. EndFor
8. $i = 1$
9. While ($i < N$) do
10. Sort $\text{PARRAY}(i, N)$ // Sort the weights from i to N in ascending order
11. $ptr \leftarrow \text{GetNode}(\text{NODE})$ // Get a node for internal node
12. $ptr \rightarrow \text{DATA} = \text{NULL}$
13. $ptr \rightarrow \text{LCHILD} = \text{PARRAY}[i]$ // Include the lowest weight as the left sub-tree
14. $ptr \rightarrow \text{RCHILD} = ptr \rightarrow \text{PARRAY}[i + 1]$ // Include the second lowest weight as the right sub-tree
15. $ptr \rightarrow \text{WEIGHT} = (ptr \rightarrow \text{LCHILD}) \rightarrow \text{WEIGHT} + (ptr \rightarrow \text{RCHILD}) \rightarrow \text{WEIGHT}$ // Path length of sub-tree
16. $\text{PARRAY}[i + 1] = ptr$ // Store the pointer to the sub-trees into the list of weights
17. $i = i + 1$
18. EndWhile
19. $\text{ROOT} = \text{PARRAY}[N]$ // Root of the Huffman tree
20. Stop

NPTEL Online Certification Courses
IT 60101: Lecture 06: Kharagpur

This is an algorithm that you can follow to build the Huffman Tree and this is basically based on Huffman Algorithm and this algorithm is required if you want to go for coding. So, writing program again is another exercise for you.

(Refer Slide Time: 33:34)


Application of Huffman Tree

NPTEL Online Certification Courses
IIT Kharagpur

Application of Huffman tree: Optimum coding for message

One very useful application is to obtain an optimal set of codes for symbols S_1, S_2, \dots, S_n which constitute messages. Each code is a binary string (combinations of 0's and 1's) which will be used for transmission of messages.

Suppose, there are n symbols to constitute a message. We are to code these symbols by means of strings of bits. One simple way to do this is to code each symbol by a r -bit string where

$$2^{r-1} < n \leq 2^r$$



NPTEL Online Certification Course
© 2020. Created by Dr. Rajan

Optimum coding for message: Fixed length coding

- Suppose a message consists of one or more of the 11 symbols, say S_1, S_2, \dots, S_{11} .
- Then the number of binary bits required to encode each symbol is 4 (since $2^3 < 11 \leq 2^4$) and a coding scheme is presented.

Symbol	Binary code	Symbol	Binary code
S1	0000	S7	0110
S2	0001	S8	0111
S3	0010	S9	1000
S4	0011	S10	1001
S5	0100	S11	1010
S6	0101		

Number of bits required to code a message of 100 symbols = 400



NPTEL Online Certification Course
© 2020. Created by Dr. Rajan

Now, let's come to the application of Huffman Tree. Now one application is very useful in coding. Suppose A to Z, 26 characters are to be coded, so we can consider five bits for each and then consider the codes. So here, we can have the eleven symbols which we want to make them put and we can use the four bits to do the coding and this is this kind of coding is called using the Binary binarization or Binary coding it is called or Binary encoding and this coding is basically for each symbol, the number of Binary bits that is required to code is same, that is why they are called fixed length coding.

So here, we get some example of fixed length coding like this one. Now suppose in a text, you have these 11 symbols only. Now, we can store this text in the form of four bits character for

each, because each symbol is there and then we can represent the document there. Then using this symbol, suppose there are total 100 symbols appear to total storage or total bits that is required is basically 400. Now, question is that, using the same set of symbols, but using some different coding scheme, can we store the same document with fewer bits? Less than 400, for example.

So, Huffman Algorithm if we can apply to this coding mechanism then we will be able to get some coding of all these 11 symbols and then we can use this symbol to store into this one so that is why the idea about compression is coming. So, compression basically it take the symbols. Those are they are in a document and then use the encryption or coding or encoding and use this encoding formula to store them in a Binary.

And then reverse can be decoding, so encryption decryption or encoding decoding concept it is there opposite to that. Anyway, let us come to the question about how Huffman Algorithm can be applied to this coding scheme so that we can have the better compression. So, this basically idea about it is an opposite idea or another alternative idea than the fix length coding, it is called the variable length coding.

(Refer Slide Time: 36:00)

Optimum coding for message: Variable length coding

Symbol	Probability	Weight	Code
S1	1/4	16	00
S2	3/16	12	10
S3	1/8	8	111
S4	1/8	8	010
S5	1/16	4	1101
S6	1/16	4	01110
S7	1/16	4	01111
S8	1/16	4	0110
S9	1/32	2	11001
S10	1/64	1	110000
S11	1/64	1	110001

Number of bits required to code a message of 100 symbols = 243

NPTEL Online Certification Courses
IT 60101: Lecture 46: Huffman

So, variable length coding is here I said they are the different symbols, 11 symbols and all the symbols are coded with this one. Now, here you see some codes are symbols like this way that

they are not fix length unlike the previous fix length coding. This is a variable length coding. Now here, if you see there are coding is given in such a way that according to the frequency of occurrence of the symbols.

Now some symbols are highly frequent. For example, in case of English alphabets A, E, I, O, U, they are highly frequent symbols, so they should be given coding in such a way that only few bits have to be there. So, those are the highly frequent symbols, they can be coded with fewer bits than the other which are less frequent, they can be given large number of bits and this way, the variable length coding and this, basically ultimately can be rewarding and here you can see earlier for fix length coding, it took 400 but now it is taking 243.

So, this will basically give the gaining. Now this basically, here I want to see that how the Huffman Algorithm can be applied in this context. So Huffman Algorithm is that let us consider the frequency of occurrence, which has the frequency of occurrence can be considered as the weight so here basically, the frequency of occurrence of all the symbol those are involved are the weights so these are the things are there. Now opposite to this weight, we can just, this weight can be considered as in weight of the external nodes and finally the Tree can be created and this Tree is the Weighted Binary Tree. Now see this weighted Binary Tree therefore can be used for coding purpose.

(Refer Slide Time: 38:01)

Optimum coding for message using Huffman tree: Huffman coding

(a) Huffman tree

(b) Huffman coding

NPTEL Online Certification Courses
IT 60101: Lecture 96: Huffman

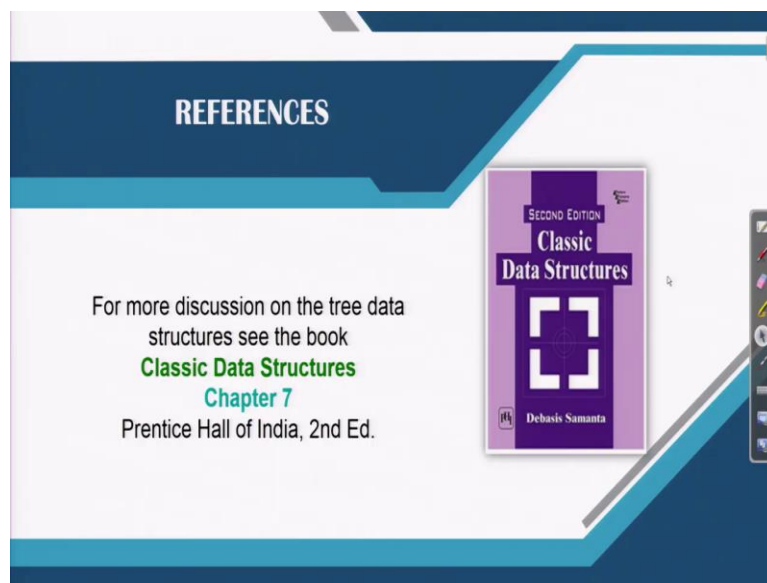
24

Now let us proceed and example. So, this is the example as we can see here. 11 symbols are there and we just take the weights as the frequency of occurrence of them and then this is the Huffman Tree that we can create with the weights as an external, so this basically Weighted Binary Tree we can say with weights of the symbol where symbols are the external nodes.

So 11 external nodes are there, 11 weights are there. With these 11 nodes, this is the Huffman Algorithm applied and Huffman Tree is obtained so this is the Huffman Tree. Now coding is very simple, starting from the root and to each symbol, if we go left then code as encode as 0 and if you go right, so here, this symbol 0, 0 and this symbol 0, 0, 0, 1, 0, so this symbol should be coded as 0, 1, 0 and so on.

Likewise, these basically 1, 1, 1, this one is 1, 0 and so on. So, this is the simple coding mechanism that can be followed and all symbols can be coded and the coding that I have already listed in the table is basically encoding for that. So this is an example that we can consider to coding is in communication, application we can have it.

(Refer Slide Time: 39:27)



Now for more details about the Algorithm and Procedure, you can follow this book for better understanding and programming for the Huffman Tree is given as an exercise for you. You can just write your Java program and all the algorithms that I have discussed, you can follow and

then write the class for this Huffman Tree and then check it how it is working and then finally you can check some application, compression or coding or other application. Thank you.