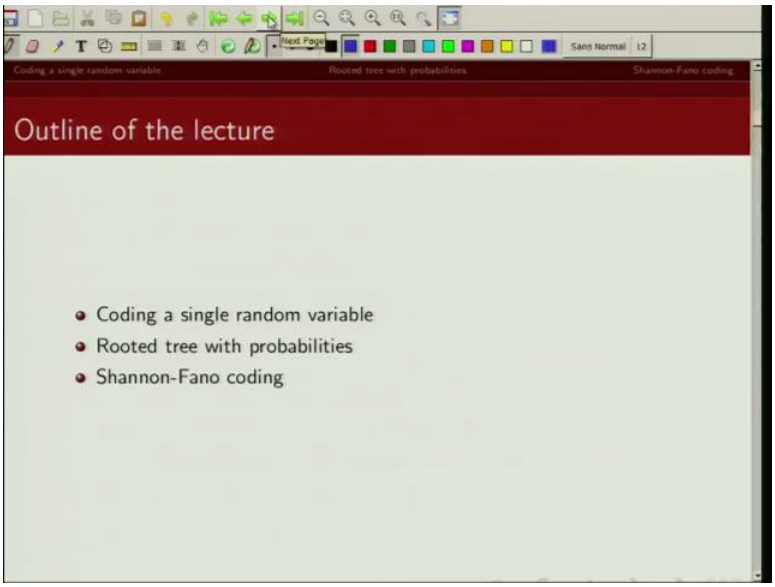
An Introduction to Information Theory
Prof. Adrish Banerjee
Department of Electronics and Communication Engineering
Indian Institute of Technology, Kanpur

Lecture – 06
Block to Variable Length Coding-II: Bounds on Optimal Codelength

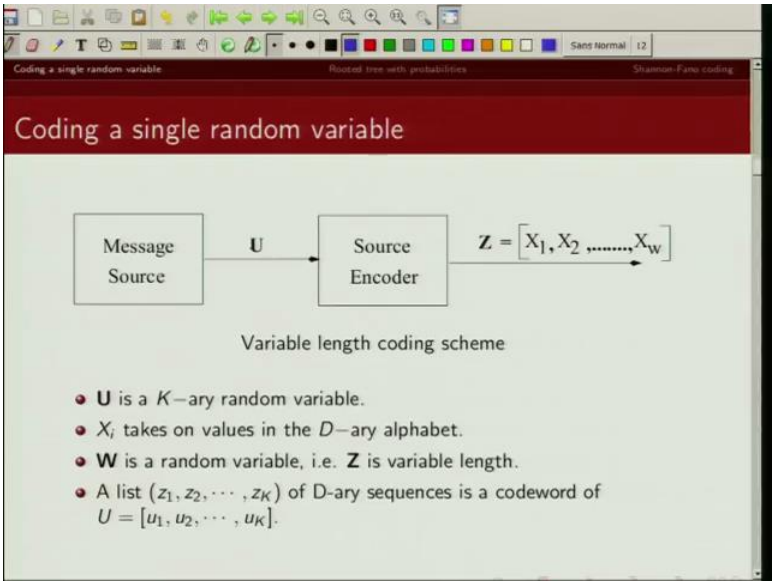
Welcome to the course on an Introduction to Information Theory. So, we will continue our discussion on block to variable length coding and in this lecture we are going to talk about what are the bounds on optimal code length for block to variable length coding.

(Refer Slide Time: 00:33)



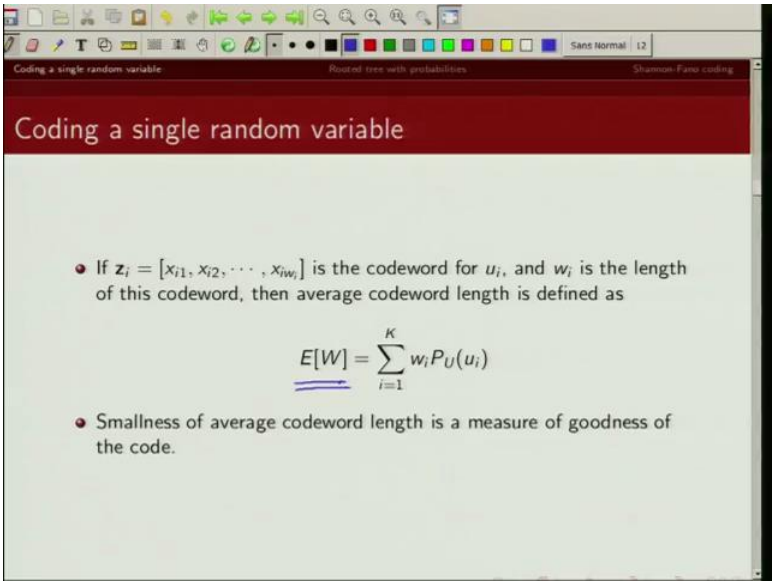
So, to get to this bound we will define what we mean by a rooted tree with probability and then we will derive the bounds on optimal code word length, and next we will talk about a prefix free code.

(Refer Slide Time: 00:56)



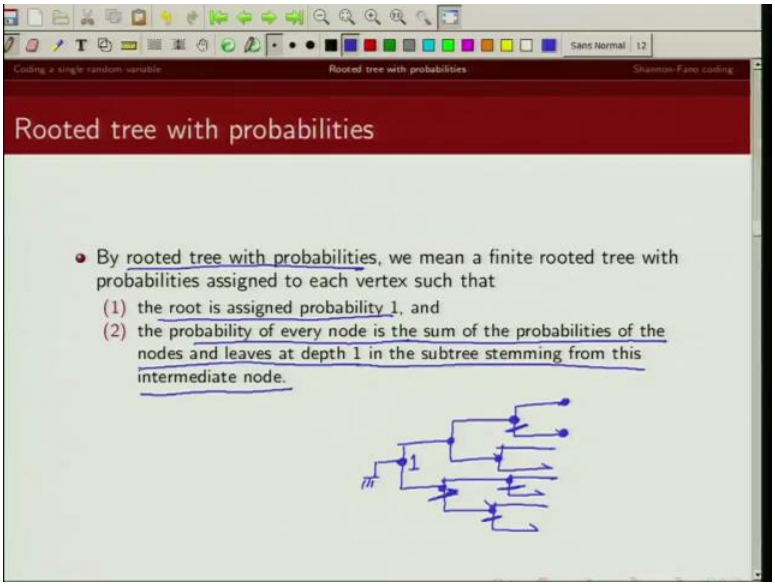
Which is known as Shannon Fano codes, so, just to refresh your memory basically we are considering this problem. We had a message source denoted by u which is a clearly random variable. So, it takes k different possible values and we want to encode this source into code words z which is of variable length and z denotes the length of the code word z and expected value of w, gives us a measure of average code word length. So, we have a list of code words corresponding to this u1, u2, u3, uk and what we are interested is in average code word length being of more or less possible.

(Refer Slide Time: 01:41)



So, what is the smallest value of expected value of w that we can have and that is a measure of goodness of our source compression algorithm? So, to get a bound on this we will require some results and some (Refer Time: 02:00) and some proof. So, we will build up those results and then we will derive the expression for lower bound on expected value of w.

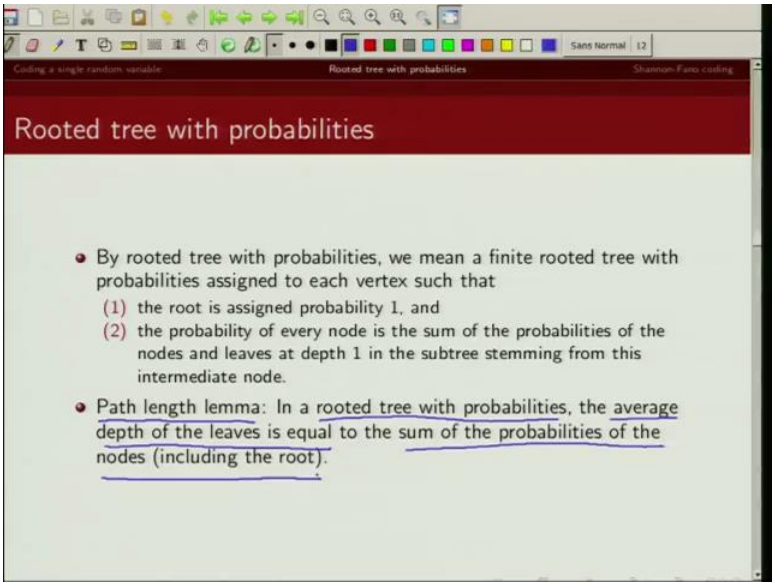
(Refer Slide Time: 02:22)



So, we will define what is meant by rooted tree with probability. So, by a rooted tree with probability we mean it is a finite rooted tree with probability assigned to each vertices, now how do we assign these probabilities this is as follows, the root is assigned probability 1 and probability of every node is the sum of probabilities of the nodes and the leaves at depth 1, in the sub tree stemming from this intermediate node. So, if you have a consider a binary tree, so this is a root node right this will have probability 1 and at any node let us say at any node w we can just draw. So, let us say we are considering this node at depth one. So, it is probability will be given by probability of this node and probability of this node. That is what we meant when we say probability of every node is the sum of probabilities of the nodes and leaves at depth 1 from the subtree stemming from this intermediate node.

So, you look at any node here, you look at nodes and leaves at depth 1 from this node. So, in this example this is this node and this node. So, the probability of this particular node is nothing, but probability of at this node plus probability at this node. Similarly what is the probability at this particular node that is given by probability of this leaf plus probability of this leaf? So, this is how we construct our rooted tree with probability.

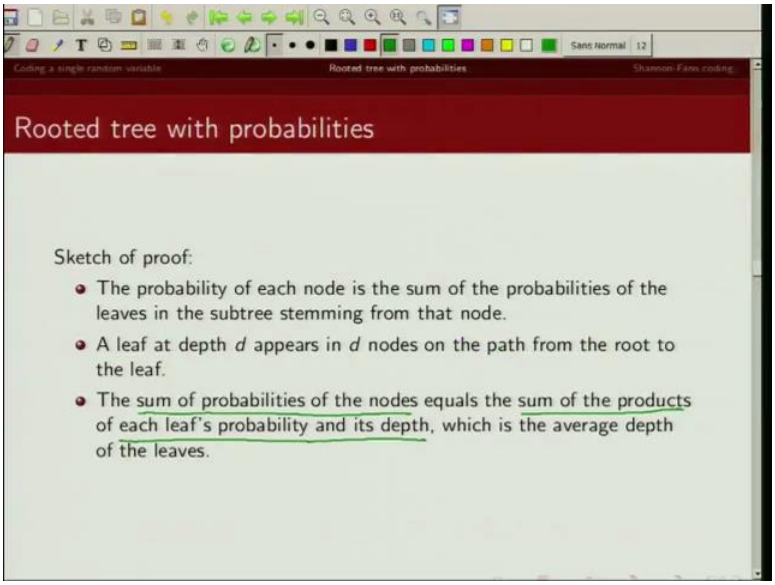
(Refer Slide Time: 04:29)



Now we are going to prove a lemma which is we calling as path length lemma. So, in a

rooted tree with a probability, the average depth of the leaf is equal to sum of probabilities of all the nodes, including the root node. So, what we are saying is the average depth of the leaves in a rooted tree with probabilities equal to sum of probabilities of all the nodes including the root node.

(Refer Slide Time: 05:08)



Let us prove this; the probability of each node is nothing, but sum of probabilities of the leaves in the subtree stemming from that particular node. Now a leaf which is a depth d will appear in d nodes on the path from root to the leaf.

So, what do I mean again I take an example of a binary tree let us just take an example, let say these are my leaves. So, these are the nodes at depth 1; these are the nodes this is a node at depth 1 these leaves are at depth node. Depth 2 this is depth 1 these are nodes at depth 2 these are the leaves at depth 2 this node at depth 3. There is a leaf at depth 3 and these are the leaves at depth 4. Now what I am saying is a leaf at depth d appears in d nodes from the path to root to the leaves. So, let us look at, this is my root node right see. If I look at path and let us look at this particular leaf this particular leaf is at depth 1, 2, 3, 4 then, this leaves are depth 4 from the root. Now how many times now we know the probability of each node is sum of probabilities of the leaves stemming from the subtree stemming from that particular node. So, probability of this particular node is

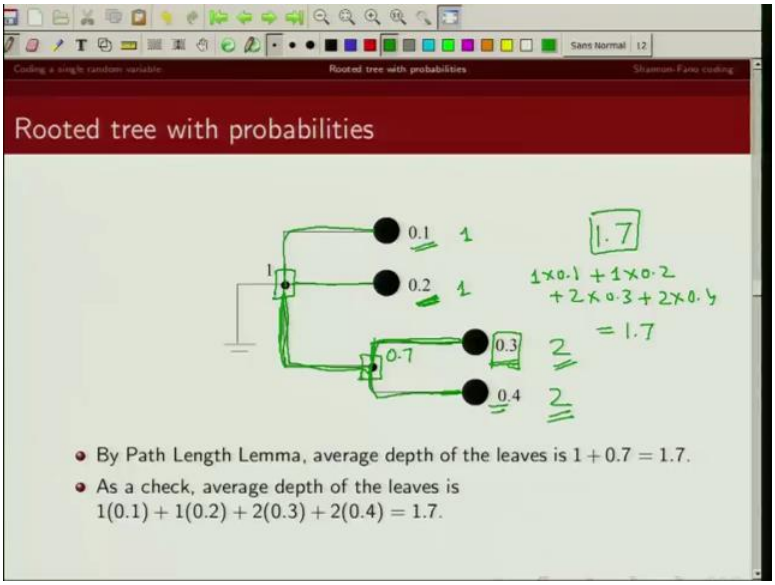
nothing, but probability of this leaf plus probability of this leaf correct similarly what is the probability of this node this is this probability plus this probability.

Now, please note that a leaf at depth d appears in d nodes on the path from leaf to the node. So, probability of occurrence of this leaf this appears here right, this appears here this appears here and this appears here. So, the contribution of the probability of occurrence of this leaf this is appearing in the probability of this node. Now this probability of this node is affecting the probability of this node. So, this probability will also affect this.

Now this probability effecting the computation of probability of this node so, this probability will also appear here. So, you can see that if a leaf is at depth d it is going to appear at d nodes it is probabilities will is going to appear at d nodes from the path from root to the leaf. So, the sum of probabilities of the node is equal to sum of product of each leaf probability and it is depth. Which is precisely the average depth of the leaf now again goes back to the example that we had here. So, what we are seeing here is note that this leaf is at depth 4 correct and if we add up the probabilities of all nodes where this leaf is appearing. So, this particular root node and this other 3 nodes you will notice that contribution of this leaf is appearing at these 4 nodes.

So, essentially the contribution of this leaf to the overall sum of their probabilities will be 4 times probability of occurrence of this leaf. So, this is what I am seeing the sum of probabilities of the nodes is equal to sum of product of each leaf probability. And it is depth. So, you can see that each leaf will appear in the sum equal to the depth of it is leaf. So, then if we do the sum of probabilities of these all the nodes including the root node, what we will get is basically we will get average depth of the leaves in the tree.

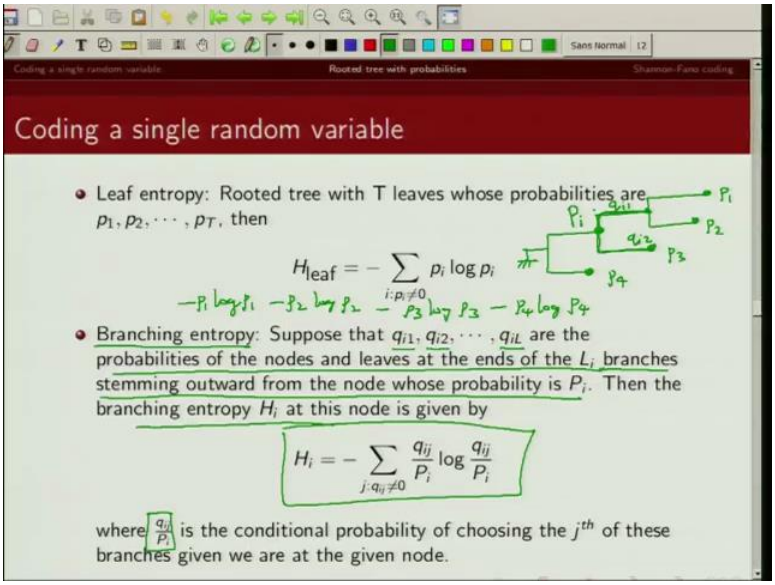
(Refer Slide Time: 10:06)



Take this example now, what is the sum of probabilities of the nodes including the root node this is probability is 1 and what is this probability the sum of probabilities of these 2 nodes. So, this probability is 0.7. So, sum of probability is of all the nodes including the root node is 1.7. Now what does path length lemma says the path length lemma, says that this is the average depth of the leaves or where the leaves there are are 2 leaves at depth 1 and there are 2 leaves at depth 2. Now what is the average depth of the leaves this is 1 into 0.1 plus 1 into 0.2 plus 2 into 0.3 plus 2 into 0.4. This is equal to 0.10 .3, 0.9 and this is 1.7 this is precisely the number which is given by path length lemma and you can see this particular node appears only once in the summation and which is the depth of the leaf this particular node appears only once in the summation this particular node. Appears twice it appears in this a probability of this node and it is also appearing in the probability of this node.

Similarly, this particular node it appears in the sum of probabilities of this particular node as well as root node and which is precisely the depth of this leaf. So, you can see from the path length lemma by summing up the probabilities of all the nodes including the root node you can find out the average depth of leaf.

(Refer Slide Time: 12:21)

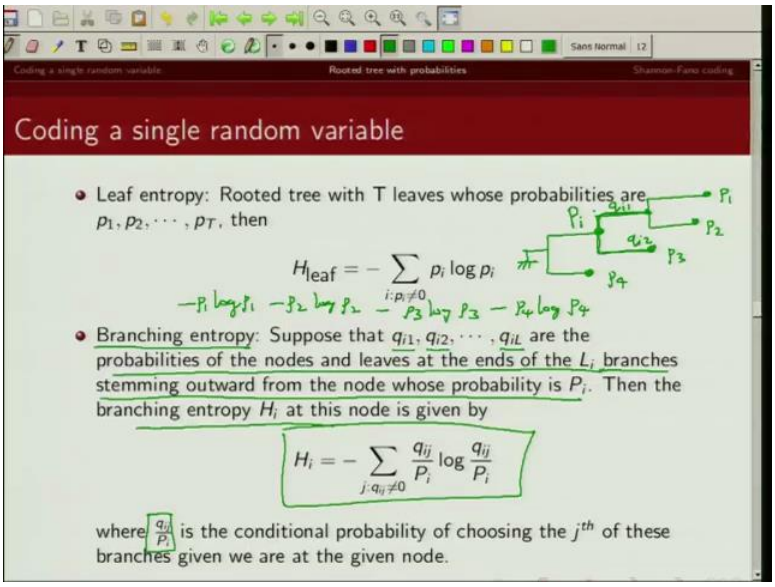


Now, let us define a rooted tree with t leaves whose probabilities are p1, p2, p3, pt then, we define leaf entropy as minus P i log P i. Sum over all is. So, this is how we define leaf entropy similarly we can define what is known as branching entropy. So, suppose q qi1, qi2, qil are the probabilities of the nodes and leaves at the end of li branches that stemming outward from the node. Whose probability is pi then, the branching entropy is given by this expression where qij by pi is the conditional probability of choosing the jth of these branches given that, you are at a particular node whose probability is pi. So, graphically let us have a look. So, let us consider a binary tree. So, these are the leaves. So, let say the probability is p1, p2, p3, p4 then this leaf probability is given by minus p1 log of p1 minus p2 log of p2 minus p3 log of p3 and minus p4 log of p4 that will be my leaf probability.

Now, if you want to compute the branch probability. Let us just take let say this node and let us say that probability of this node is capital pi. Now there are 2 branches leaving from this node, let us say the probability of proving this branches this probability is, this probability is qi1 and this probability is qi2 then qi1 by pp1 is the probability of choosing. This branch given I am at this particular node and qi2 by P i is the probability of choosing this branch given I am at this node whose probability is P i. Then the branching probability is basically given by minus qi1 by P i log of qi1 by P i minus qi2

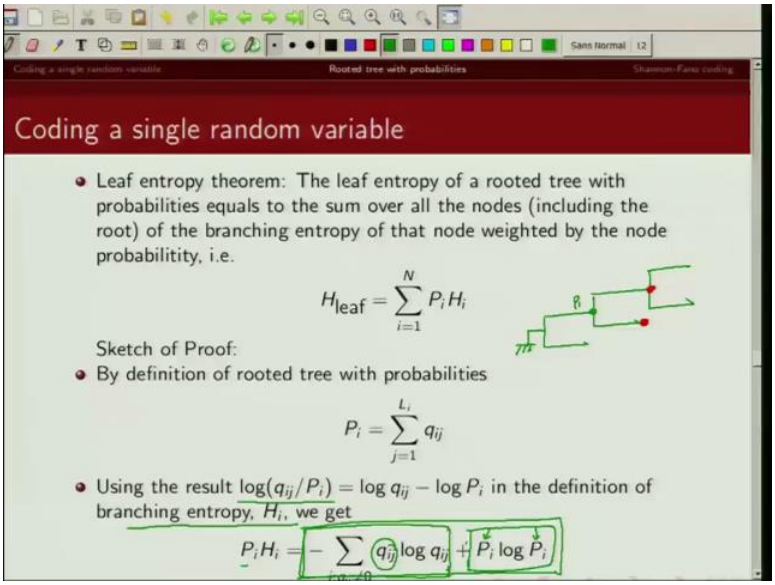
by P i log of qi2 by P i. So, that is my branching entropy. So, I define 2 terms 1 is leaf entropy which is the entropy of the leaves and other is branching entropy. So, this is computed at each node whose probability of occurrence is P i and this probability of choosing any of these l branches which is stemming from this node whose probability is P i.

(Refer Slide Time: 15:49)



So, let us take this example in this case the leaf probability probabilities are 0.1, 0.2 0.3 and 0.4. So, the leaf entropy in this case is given by 1.846 bits. Similarly we can compute the branching entropy we can compute the branching entropy here at this node or we can compute the branching entropy at this particular node. If you compute the branching entropy of this particular node, which I am denoting as h of 1 then, what are the branching probabilities of selecting each of this branch probability of selecting this branch is 0.1 by 1 this is 0.2 by 1 and this is 0.7 by 1. So, the branching entropy for this particular node is given by this expression similarly, you can compute the branching entropy at this node we can denoting by h 2. Now the probability of selecting this branch is 0.3 by 0.7 and similarly a probability of choosing this is 0.4 by 0.7 then my branching entropy is given by this expression.

(Refer Slide Time: 17:34)



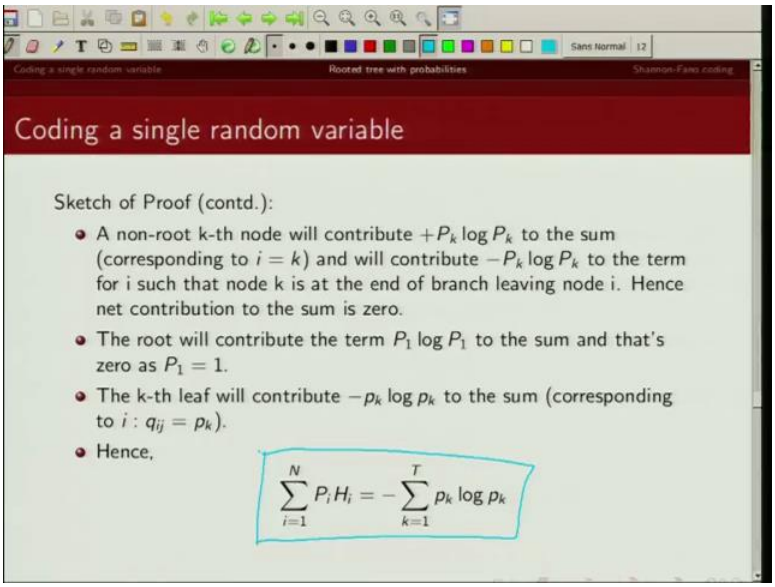
Now, once we have defined this leaf entropy and branching entropy. Let us prove a theorem if we call as leaf entropy theorem. So, the leaf entropy theorem says that the leaf entropy of a rooted tree with probability is equal to sum over all nodes including the root node of the branching entropy of that node weighted by the node probability. So, this leaf entropy theorem relates leaf entropy to the branching entropy and it says the leaf entropy is nothing, but branching entropy at node I multiplied by the probability of occurrence of that node I and this summation over all the nodes including the root node that is my leaf entropy theorem. Let us try to prove this result. So, by definition we know that sum of probabilities at a particular node. So, the probability of the node is nothing, but sum of probabilities of it is leaves or nodes at depth 1 from that particular node.

So, if you are, let us say interested in finding the probability of this node this is nothing, but probability of this node plus probability of this node. So, we have l I branches you have this relation. So, that is what I am saying that this follows from the definition of rooted tree probability that probability of a node in this rooted tree is nothing, but probability of nodes and leaves at depth 1 from that particular node. Now let us look at this from this term log of q i j by P i. So, in the branching entropy if, you recall the definition of branching entropy you can see here you have this term of the form log of q i j by P i.

So, if you look at this term log of q i j by P i this is this will be log of q i j minus log of q i. Now from the definition of branching entropy if I multiply this by P i, what I get is P i H i is given by this expression. Now what is this pi, P i is the probability of that particular node and what are what are this q i js. q i js are the probabilities of nodes and leaves stemming from that particular node at depth 1. So, if you have a tree like this and let say I am looking at this particular node then P i is the occurrence of probability of this node where qi1 and qi2 are these 2.

So, these in the red are my probability of these nodes is my q i js, where as in green what I have is my probability of P i. So, note that in this P i H i product there are 2 terms which are coming in 1 contribution due to probability of this node which is the positive contribution second contribution is coming due to the nodes and leaves at depth 1 which is a negative contribution. So, let us look at these terms further.

(Refer Slide Time: 21:27)



So, if you have a non root kth node it will contribute a sum corresponding to P k log P k to the sum which I just now mentioned. So, if you have a non kth root then, we want contribution from due to the probability at the node which is P k log P k and the same node will contribute minus P k log P k to the term. Because if you consider non root node, let us say it if you consider this node is a non root node right. So, if I do this

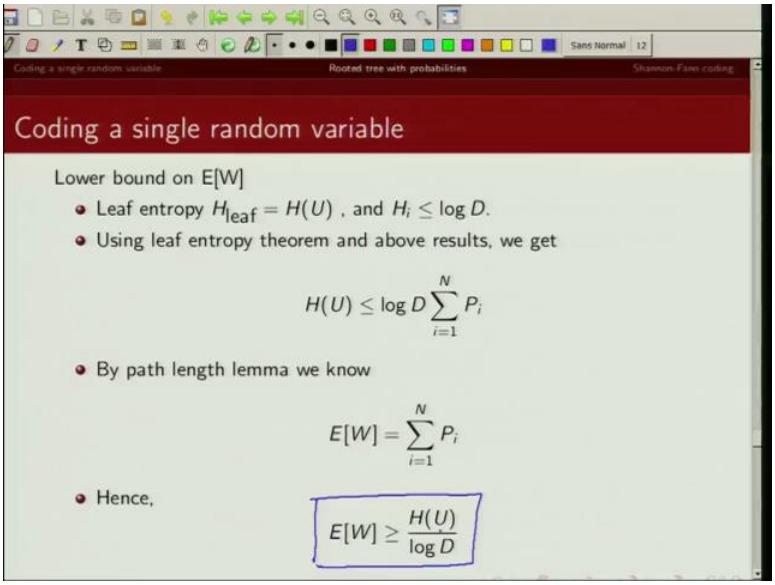
summation P i H i at this particular node the contribution of this P i would be this positive here.

How about the same node is for this particular node for this particular root node this is another node at depth 1. So, this P i is actually q i j for this particular node right, if I compute P i H i for this particular node this P i is giving me a positive contribution which is P i log P i is giving me a contribution which is P i log P i and the same node is acting as q i j for this particular node. So, then this will give me minus P k log P k contribution. So, each of the nodes which is not a root node will give a P k log P k contribution. When this summation is done over that particular node and it will give me a minus pk log pk contribution corresponding to summation, which is taken at 1 node before at particular node. So, the net contribution of any kth non root node will be 0.

What about root. So, if you look at root what is the contribution of the root. Now this root will contribute P i log P i where, P i is 1 to the sum. So, that contribution is also 0. So, the contribution of a root node is also 0. So, what we have shown is to this summation P i H i summation over all nodes including root node contribution of the root node is nil contribution of a non 0. A non root node is also net contribution of a non root node is also 0. So, then what is the only contribution left only contribution, left is contribution of the leaves. So, if you have trees of the form this and if you are looking at, let us say sum of probabilities at this particular node then since these are leaves they will contribute minus q i j log q i j term corresponding the probabilities of these particular nodes. So, the net contribution to this summation P i H i sum over all the nodes including the root node the total contribution will be just the contribution of the leaves and this is nothing, but the leaf entropy that we have defined.

Hence we have proved that this summation P i, H i sum over all the nodes including the root node is nothing, but given by leaf entropy. So, this is the important result because it we are going to make use of this result to compute bounds on the size of lower bound and the block to variable length coding.

(Refer Slide Time: 25:53)

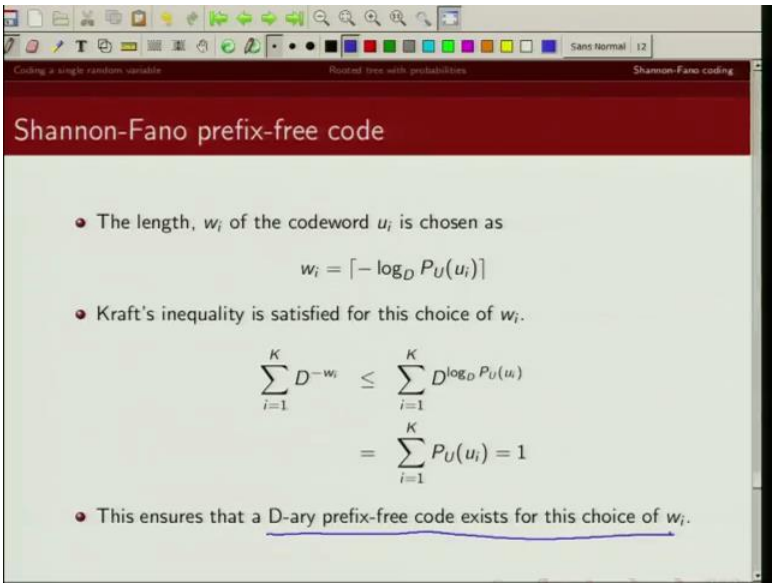


So, now, what is leaf entropy if we are assigning each of our code words to these leaves the uncertainty in the source u is nothing, but leaf entropy and what is branch entropy since there are d branches stemming from the each node. So, we know from the property of discrete random variable that the entropy cannot be more than log of number of possibilities and here the number of possibilities is d. So, the branching entropy is upper bounded by log of d and this equality happens. When each of the branches are likely to happen. So, then invoking the leaf entropy theorem we, can write that because leaf entropy theorem says that leaf entropy is equal to summation of P i H i where summation is taken over all the nodes including the root nodes. So, this is our leaf entropy right and branching entropy we upper bound by log of d. So, then this summation P i H i because we are upper bounding this by log d we can take it out. So, what is left is summation of P i over all the nodes including the root nodes.

Now, what is this quantity from the path length lemma we know this the expected depth of the leaves and what are leaves are my code word. So, this is the expected code word length average code word length this follows from the path length lemma. If I plug this 1 in here what I get here is a lower bound on average code word length. So, the average code word length should be at least equal to uncertainty in u divided by log of d. So, you can see using this leaf entropy theorem we have arrived at a lower bound on average

code word length for this block to variable length coding scheme now finally, let us talk about some practical prefix free coding techniques.

(Refer Slide Time: 28:40)



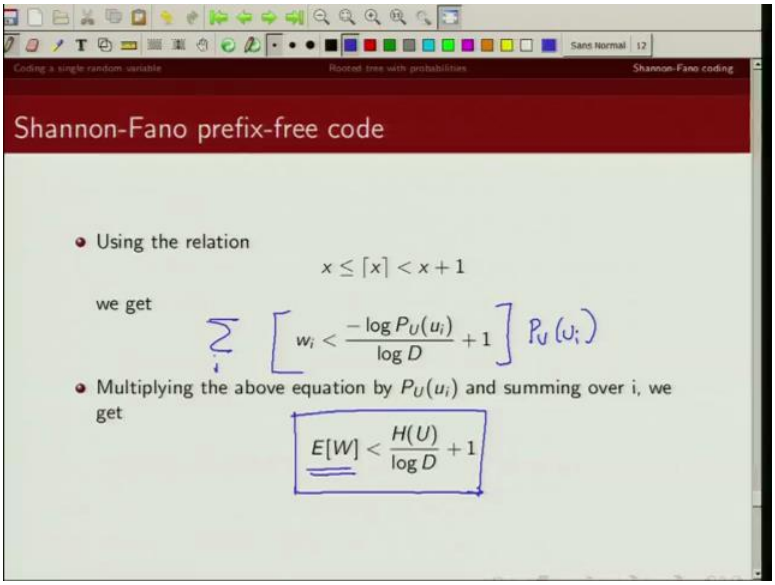
So, we will start with Shannon Fano prefix free code. So, in Shannon Fano prefix free code the code word lengths are given by this. So, it is minus log of d of probability of occurrence of u I say in other words the code word length is inversely proportional to the probability of occurrence. So, if a code word is occurring more frequently it will be a signed a smaller code word length if the particular occurrence of u is happening with less frequently it will be assigned a larger code word length.

Now, to verify whether a prefix free code exist which has code word length given by wi what do we need to do first we need to first check whether Kraft's inequality is satisfied. So, this is what we are going to do we are going to check that if you choose w I in this particular way will craft inequality be satisfied. So, what does Kraft's inequality says, so if you it is a clearly random variable then d raise to power minus w I sum over all these k should be less than equal to one. So, let us see whether Shannon-Fano code satisfies the condition for prefix free code. So, this we can write as now note that this is a seal function. So, this minus log of P i u i might be some real numbers, we round it off to the nearest larger number. So, here we are doing a minus of w I which is integer bigger,

bigger I mean the smallest integer basically which is a bigger than this quantity. So, if we if we remove this seal function. So, we are raising it by smaller quantity and this is minus of that. So, then this will be greater than equal to this.

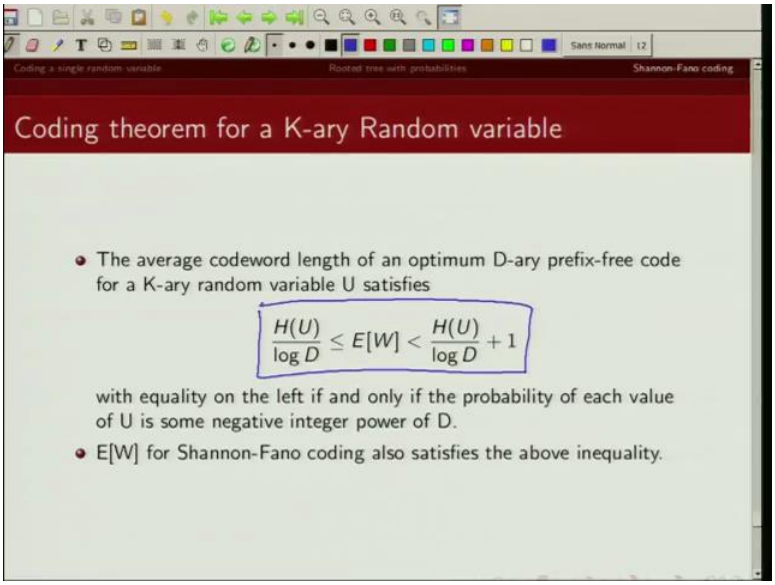
Now, this can be written as summation over I equal to K of P u I and this is nothing, but equal to one. So, what we have shown is if we choose our code word way length in this particular fashion then Kraft's inequality will be satisfied hence this will be a prefix free code. So, this ensures that Shannon Fano code is a prefix free code for that choice of w I that we have chosen.

(Refer Slide Time: 31:41)



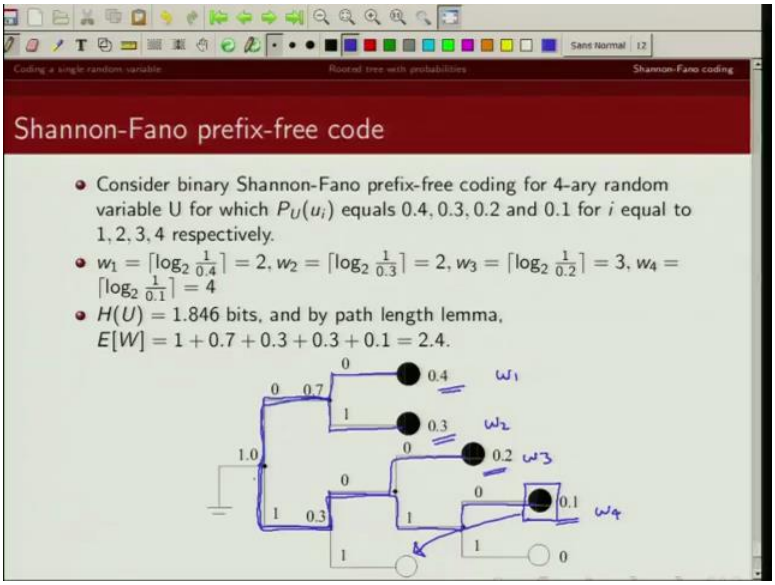
Now, we know that a seal function can be upper bounded and lower bounded like this. So, x is less than equal to seal of x is less than x plus 1 and w I is, given by this expression. So, we plug that in we can write from this expression that w I is less than minus log of P u I by log d plus 1 and if we multiply both sides by. So, if you multiply both sides by P u u I and sum over all is sum this over all I then this would w I P u i, sum over all is that would be expected code word length and minus P u I log P u I sum over all I that is the entropy function log of d and summation of P u I is sum over all is would give me 1.

(Refer Slide Time: 33:00)



So, what I will get is a simple upper bound on expected code word length So, average code word length of an optimal D-ary prefix free code satisfies this relationship. So, expected code word length thus lower bounded by this quantity and it can be upper bounded by this quantity and of course, this equality of the left hand side will only happen if the probability of each of these values of u is some negative power of d and in that case the Kraft's inequality will be satisfied with equality now Shannon Fano code also satisfy this particular bound.

(Refer Slide Time: 33:46)



Now let us take a simple example of Shannon Fano coding. So, we have a fourary random variable of random variable which takes for different values u1, u2, u 3, u 4 and their respective probability is given here. So, u1 occurs at probability 0.4, u2 happens at particular 0.3 u3 happens with probability 0.2 and u4 happens with probability 0.1.

Now, according to Shannon Fano coding the code word length is given by log of 1 by P i and seal function of that. So, the code word length corresponding to I equal to 1 is then, given by this. Similarly we can find out the code word lengths for other 3u i. So, what we have is now we know the code word length of our prefix free codes. Now if we know the code word length we know from the algorithm that we talked about earlier how to map these code words of length w1, w2, w k into a prefix free code and the way we do it is we consider we draw a full D-ary tree of length equal to maximum of w1, w2, w3, w4 and this case that is 4 and then we start we arrange these code word lengths in the ascending order.

So, we will we will basically arranged them like this and then we start with the first code word proven that free at depth equal to from the root equal to w 1 if we do the same thing for w2, w3, w4 and that is how we get our we are able to map each of these code words of length w1, w2, w3, w4 into a D-ary tree and that is how we can get a prefix free

code.

So, you can see here I have already proved. So, this is my tree of depth maximum 4 lengths 4 and I have already proved it and you can see. So, this is corresponding to u I code word corresponding to u1 this is code word corresponding to u2 of length w2 this is of length w1 this is of code word corresponding to u3 of length w3 and this is code word corresponding to u4 of length w4. Can you can just again we arrived at this by first creating a full D-ary tree of depth 4 and then we start proving the tree at w 1 equal to two. So, you can see the depth here is two. So, what was coming out from this node we had proved it similarly w 2 was two.

So, we considered a load at depth 2 and then we proved the tree beyond that, we followed the same process for w3 which is at depth 3 and similarly for w4 which is at depth 4. Now you can see clearly this is not optimal coding. Why? For example, this particular code word which is depth 4 we could have easily brought in here which is of depth 2 which would have reduced our expected code word length. So, with this we will conclude this discussion on bounds on optimal code word length and Shannon Fano coding in the next lecture we will talk about optimal prefix free coding which is known as (Refer Time: 38:10). So, we first talk about what are the conditions that need to be satisfied for the code to be optimal.

Thank you.