Data Structure and Algorithms Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 27
Understanding Tree Data Structures

Let us start one new topic it is another data structure tree we have learned many data structures, but tree compared to other data structure unique in several respects. So, we will learn about tree structures in this module. This module includes 5 more video lectures we will cover all the concept in tree data structures and also we will see the programming with tree and then the JCF utility for tree in any program implementation.

So, today we will cover about basic concept about that tree data structure. So, we try to understand what exactly that tree and why these tree structure and then how these tree structures and so basic. So, it is a basically basic concept about tree particularly those are not from the CSE discipline for them this needs to be studied a little bit in depth. And for those other CSE students, they can learn many more new things related to tree also. So, this, this video lecture basically plan to introduce that tree structure at the very beginners level.

(Refer Slide Time: 02:01)

So, first we will introduce the concept of tree. And as because it is a new data structure, so many terminologies are there. So, we try to first get acquainted with those are the basic terminologies

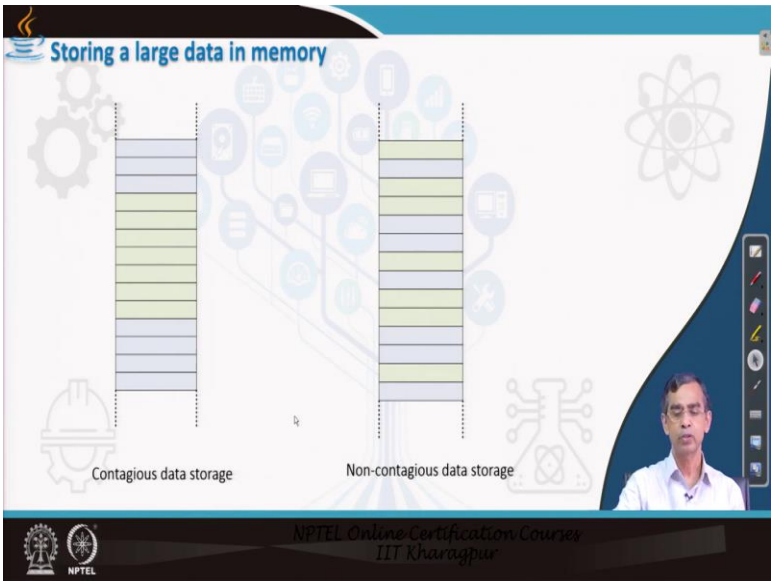
related to these tree structures. And then finally, every data structures should have its own characteristics.

So, tree also no except some, we will see exactly what are the different properties that tree should have. Mainly we will discuss about a special type of tree only which is used in theory and in Computer Science is basically all about a tree and if it is tree then our only it is binary tree is very surprising thing that no other tree's concept is there, except the binary tree. Now, there are many variations of the binary tree also will be covered in this course.

But anyway, a basic introduction to binary tree will be covered first, and then we will try to understand how a tree can be stored in computer memory. Because we know how a stack can be stored using an Array or LinkedList like Queue also can be Array or LinkedList. How Array can be stored in memory? That we have studied, how LinkedList can be stored in memory that also we have studied.

Now, here also we have to study about how a binary tree can be stored. So, this is the representation of binary tree. So, this is our topics for today.

(Refer Slide Time: 03:32)



Let us start first give me an introduction, the introduction to the tree data structures. Now, first of all, why the tree data structure come into the picture? Now, so basic objective of any data structures is basically to store the elements and in such a way that it can be accessed as fast as possible and it should be also that efficient memory can be utilized.

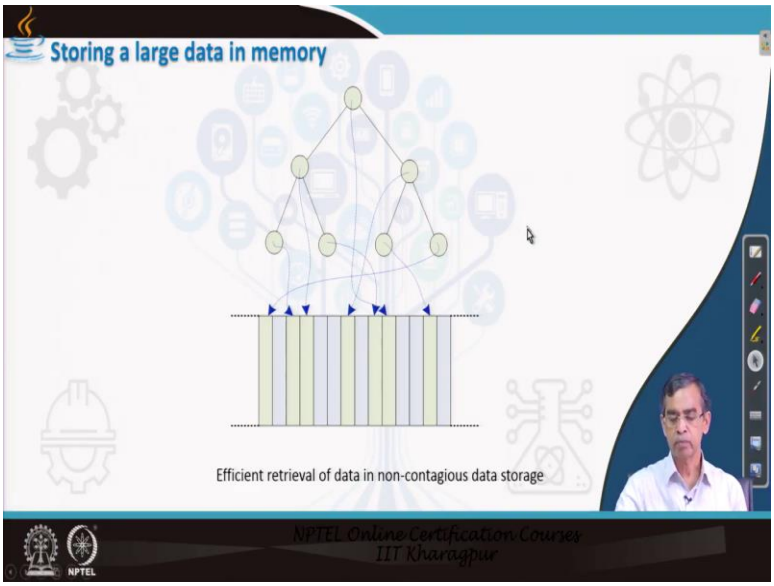
So, there are 2 aspects mainly matters. So, for the data structures and its utility is concerned mainly that faster access and second thing is that better memory utilization. Now, from the memory utilization point of view Array absolutely very good because it does not take any other extra information other than the elements to be stored there unlike in case of LinkedList or other data structure where we have to store pure information rather link like this 1.

But sometimes you know, so this extra storage is at the cost of extra speed actually. So, there is a speed versus memory take up. So, Array is good, so far memory utilization is concerned but operations are not so, faster because insertion and deletion is really costly appears. On the other hand, memory utilization if you see sometimes say suppose your memory available that is there in computer memory I am talking about, but not at 1 place.

For example, you want to store an Array of 1000 elements and for this 1000 element it required 800 bytes. Now, in your memory 800 bytes free space is available, but not at 1 location that means as a single block. So, in that case the memory is available I mean free space 800 bytes quite available, but as it is not in a contagious location, so, you will not be able to allocate a memory for the Array.

So, the Array operation will fail. So, if it is a contagious data structure, then fine, Array you can and then if it supports fine, but if it is a non-contagious then Array may not be good 1 and usually the memory stored the data in a non-contiguous manner rather better. So, for small data that you want to maintain Array is fine, but for larger data that is, that needs to be maintained then definitely Array should not be, you have to give a second thought to Array to represent your data that is what. And the data usually in the real life application is very large it is called Big Data like so, tree Array is really not good.

(Refer Slide Time: 06:30)

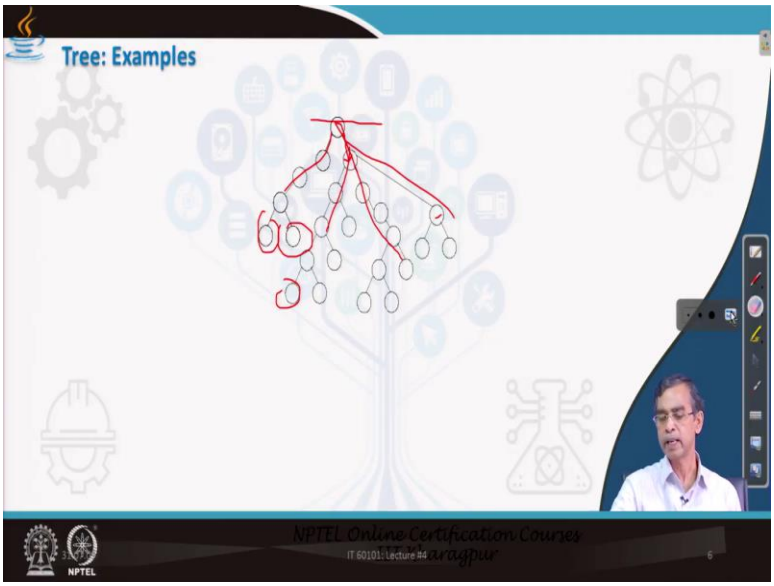


So, tree is the solution. So, what is the idea about the tree is that; if the elements are not stored in a contagious location. So, let it be in a whatever it is memory is available, but if you do that, then we have to have the ordering information to be maintained. We have learned about how LinkedList structure is used to maintain that which elements after which elements like.

So, ordering is imposed there. But if you see that is in a linear fashion. So, in that sense LinkedList is called a linear data structure. But there is another concept is called nonlinear ordering that is sometimes very fast then the LinkedList operation. Because in case of LinkedList if you want to search for a particular element, then you have to start from the header and traverse the list 1 by 1 and that is why the accessing operation in LinkedList other than the first element actually it is very costly.

But here the tree is a basically a solution where the fast access can be possible. We can make faster access than any other data structure like Array or LinkedList structure. Now, we will see exactly how it is possible using tree and that is the strength the tree or what is called a plus point of the tree is that it can store elements in a non-contagious manner and that to the be retrieved in a very fast very in the fastest way possible actually and let us see this is the idea or motivation rationale behind this new structure.

(Refer Slide Time: 08:17)



Now, let us see first how a tree look like actually. As a name to implies that a tree basically is a real-life tree a tree has a root and then it has many branches and then finally it has leaf and then fruits, flower and everything is there. So, actually a tree based on the same concept, it has a root, it has a branch, it has leaves, it has fruits and flowers like this 1.

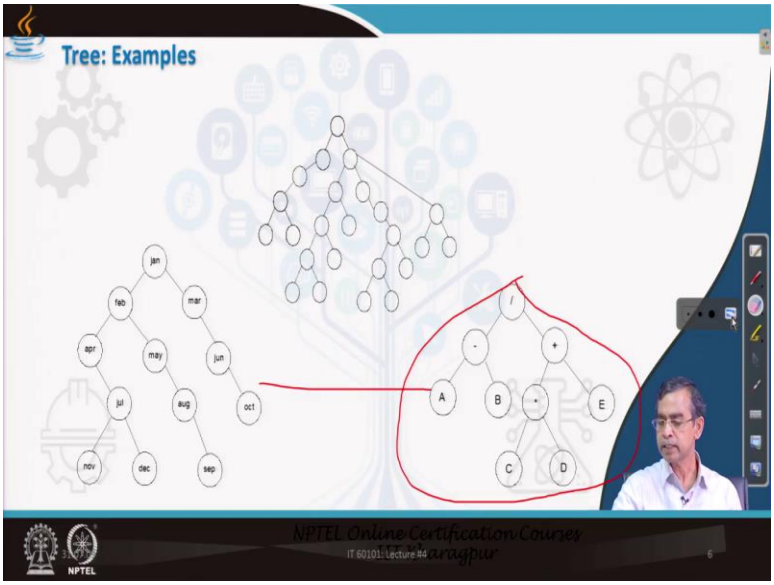
Now, this is typically a physical structure of I mean if you have a tree look like so this is basically root the starting point and these are the different branch actually you can say these are the different branch and these are the leaf that is our last node actually here the end because after that, there is no more branching it is there and where is the fruit and flowers?

So, actually each node here they are not dummy, they basically store the data. So, all elements which is stored their data we can tell they are basically fruit and flower in this tree. So, actually we have to store fruits and flowers in the form of a tree structure and then what is that good point is that starting from the root finding a particular fruit or flower is very fast actually it is not

necessary to traverse the entire tree, we can have the only in which branch this fruit and flower exist we can reach to there.

So, this way we can go to that particular target elements at a very fastest rate actually. So, this is a concept about that tree and actually physically, a theoretical tree or tree of Computer Science looks like this as it is on here, and this is a general tree. But there is a special tree we will be discussing in this lecture called binary tree. This is not a binary tree of course, I will see exactly what is our concept of binary tree actually it is there.

(Refer Slide Time: 10:13)



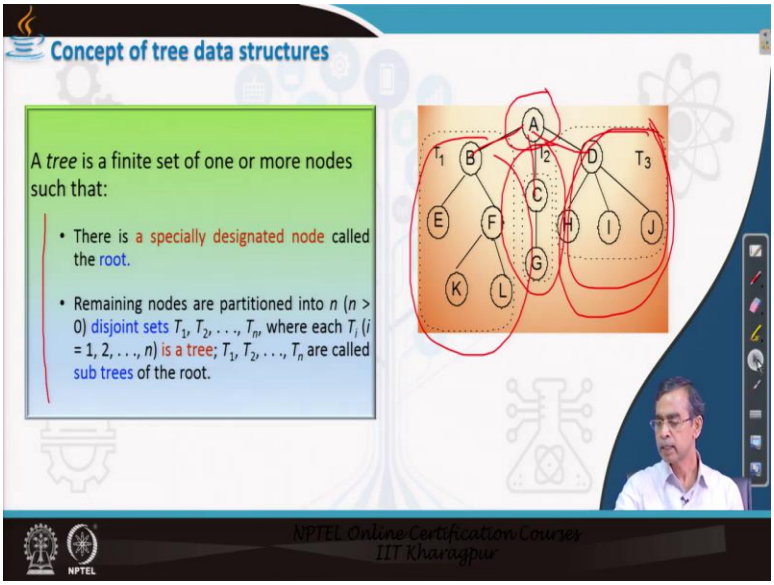
Yes, this is an another example. And you can see this tree stores information and as you see, here, we have stored the different months of a year. These are different things. And for these different months we can store the different other elements and everything, whatever it is there. And this is a particular example, you can find the difference between the 2. This, although it does not look like a black box is blank.

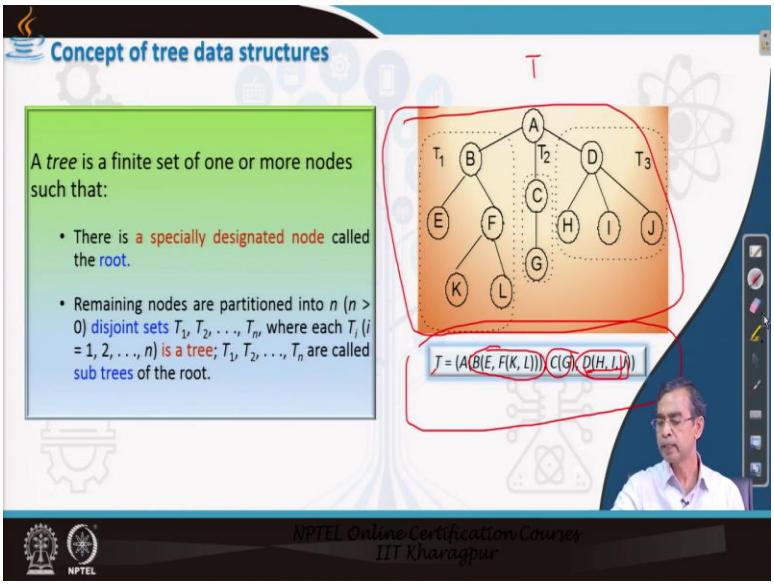
But it is not the blank, the tree is loaded with some elements in it. And another difference is that here every node has at most 2 branches for example here, but here are actually some nodes has more than 2 branches. Some nodes have only 1 branch or whatever it is there. Anyway, so this is a particular difference and in this context this has a special what is called characteristics that is called a binary tree.

It has at the max, maximum 2 branches. Anyway, otherwise these are the same concept actually. So, this is the idea about tree. And here is another example of tree and you can see the difference between these tree and these tree is obviously clearly visible, because this tree store some string or names like, but here these tree stores something else. It basically store operator and operand or we can say in other words is that each store and arithmetic expression.

We have also studied how an arithmetic expression can be stored in the form of an Array, how an equation expression polynomial can be stored in the form of a LinkedList, but here is an example, this is a binary tree how an arithmetic expression can be stored in the form of a tree. So, tree has many applications, but we can say what one the thing is that it can store any, any information, any type of information rather as you wish and also it can be stored in a noncontiguous memory location and the tree structure is such that any element can be accessed in a fastest memory. So, this is the idea about the tree structures.

(Refer Slide Time: 12:31)





Now, let us learn few many more things about the tree. And so, this is a concept I will discuss about the basic idea about how the tree can be stored in a different way, but it is a concept actually the theoretical definition. This is a theoretical definition as I said about how, how formally a tree can be defined. So, this says that a tree is decided by root node every tree should have only one root node that is the important characteristics, all these definitions to make the things more stringent for application point of view, because every should be defined in a very proper manner actually.

That is why it makes sense actually. So, that is a proper definition or formal definition is equal, we are discussing a formal definition here. Now, this definition says that every tree should be designated by a special node called a root node for example, this is a root node and then each node whether this root node or any other node will be characterized with a number of sub trees. For example, this root node has sub tree called T1 is another sub tree, and this is a sub tree.

Actually, this tree has 3 branches, and each branch leads to sub tree. And if you come to each sub tree, each sub tree again is tree. So, it is called a recursive definition or it is a null. This is a termination condition for any recursive definition and this makes sense that how it will look like it can be as large as possible because it can have many branches and many sub trees for each nodes and it can be as large as possible.

So, this basically tell you that tree can store in infinitely, infinite any number of elements into it and root is basically is the starting point to access any elements you need. Because there are so

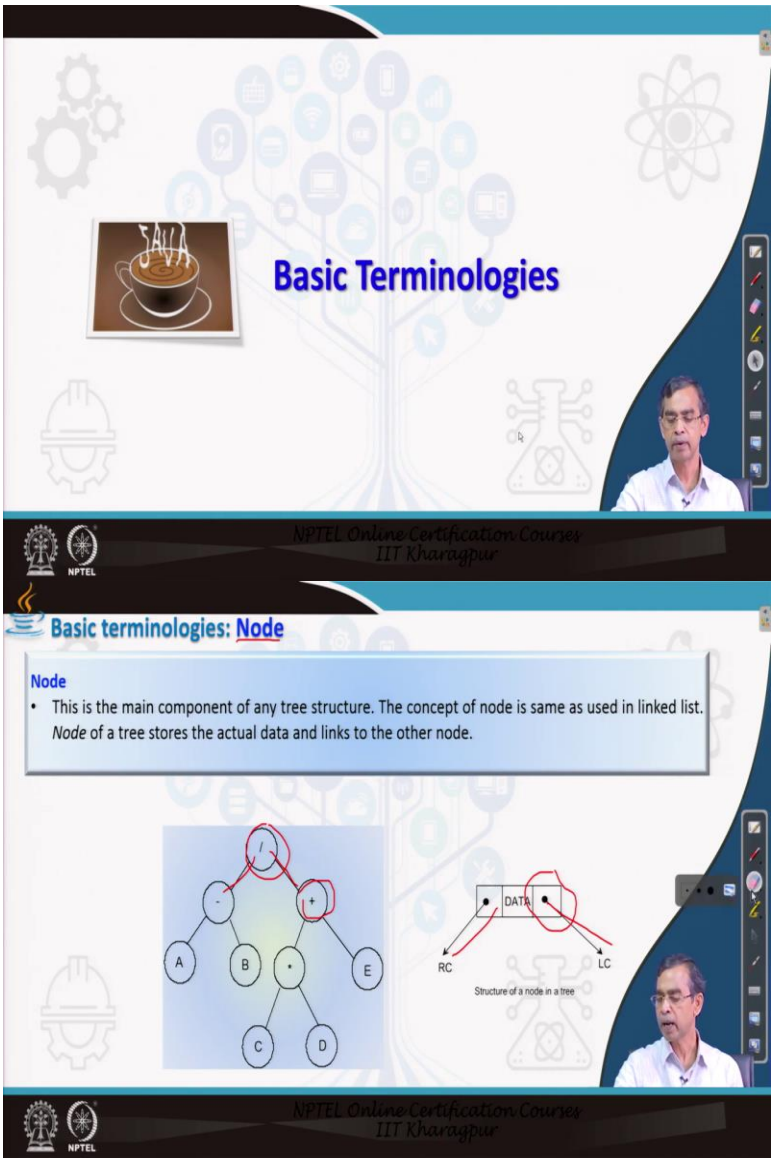
many nodes are there so, many elements are there, but you have to start from somewhere. So, root is the point and this basically the concept about the which this is a concept that the Computer Scientists have decided about to define a tree and then follow this tree in many theory in many application and some programs.

So, let us learn about few basic terminology related to tree we have to have the basic concept about and a tree, this is a very old concept, I just want to see about how a tree look like this can be stored in the form of an expression actually, because the expression is a basically string. So, here this is suppose tree this denotes that tree T and we can denote and denote this T as an expression.

The idea is that here if you see A then he it will store the sub tree by the parentheses. So, A has a sub tree 1 the sub tree this is the one parentheses another sub tree and this is a sub tree. Now, if you come to this again another tree, this has the sub tree for example D has that these elements, it does not have any other elements. So, parentheses so it is from the parentheses within the parentheses innermost parentheses and again inner innermost parentheses these basically lay down you to store all the elements all the nodes in a tree in an expression.

But it is a very pretty old concept and no more use but is only has the important that, how a tree can be stored in a compact to a in the form of a string? Only and Java is good for handling string. So, many of you can think about that storing your tree in this type also. Although it is it is obsolete, but still many, many application like natural language processing or missing information retrieval and web page, web content storing and everything this kind of concept is used.

(Refer Slide Time: 16:51)



So, now let us have few more definitions or terminologies related to our Queue structure. First is our concept of node we have discussed. Now, here as you see each tree is basically the collection of nodes. So, for example, here is node. Now, here if you see how a node can be defined because node is node. Now, how this node can be defined, I am telling about define means, whenever you have to write a program, how a node can be declared and how a node can be defined.

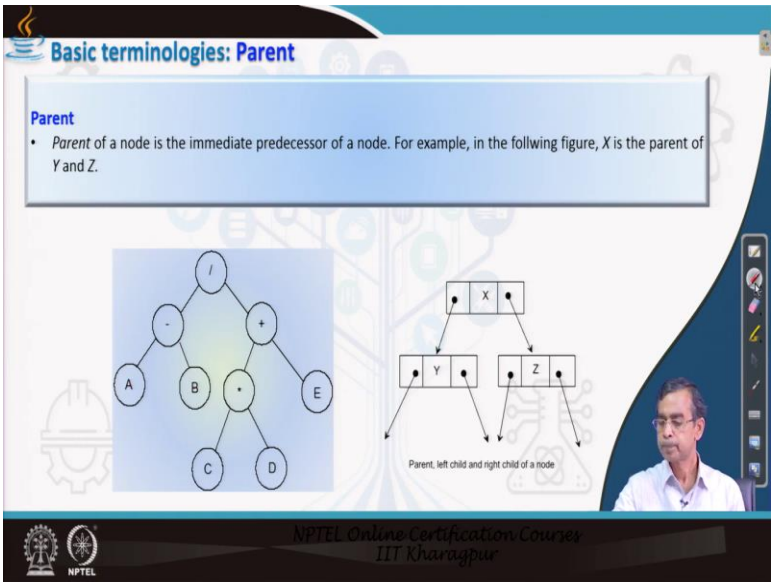
So, a node in a for a tree I am telling about a specialty is a binary it our discussion will be limited because binary is ultimately that is basically everybody use it as a tree has does not have any

implication or any significance. So, any node of a binary tree is characterized to it, 3 element, 3 information rather. Now again, I you can recall for a LinkedList, it is characterized again, it is a node of a singly LinkedList or that it is characterized with 2 information, data and link.

In case of doubly LinkedList it is characterized with 3 information data, then right link, left link, but in case of tree just like a doubly LinkedList, it is characterized with 3 information, the data, it is just like fruit and flower where actual value to be stored and it has 2 link field. This link field is basically 2 point the left subtree and right subtree. So, this particular link field is basically 2 point that left subtree when is the pointer to the another branch and this field stored that pointer to the right.

So, this is called the left child and right child here. For example, this is a 1 node it has left child this 1 so it basically store the address of the left child and address of the right child and every node therefore store the address of their 2 children, left child and right child. And this way the entire tree can be stored the entire binary tree can be stored and that is the beauty of this. Only simple node structure is very useful to store any I mean data of any capacity any size. So, this is about idea about the node information.

(Refer Slide Time: 19:35)

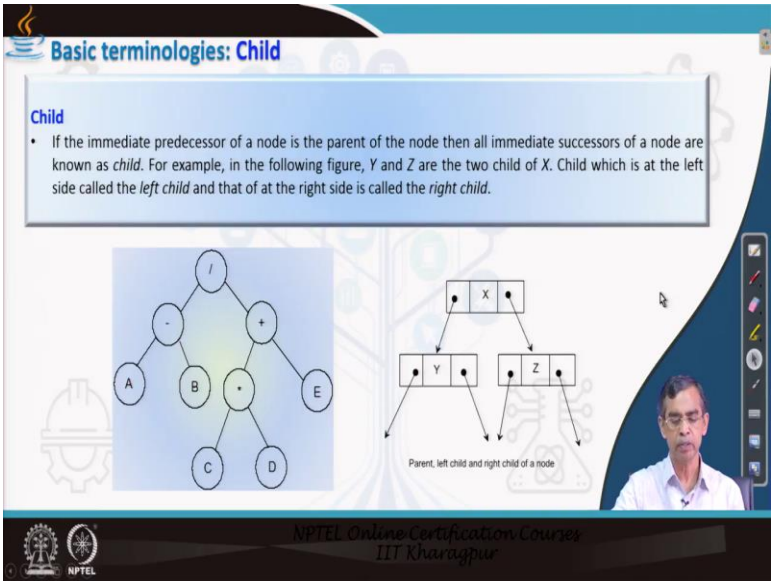


Now, let us discuss about another as a parent. Now, every node is characterized to it 2 things their parent and its children. Now, here in this case, this is a 1 node which does not have any

parent This is a special node, root, root does not have any parent otherwise every node has their parent. So, these nodes it has parent and this node it has parent.

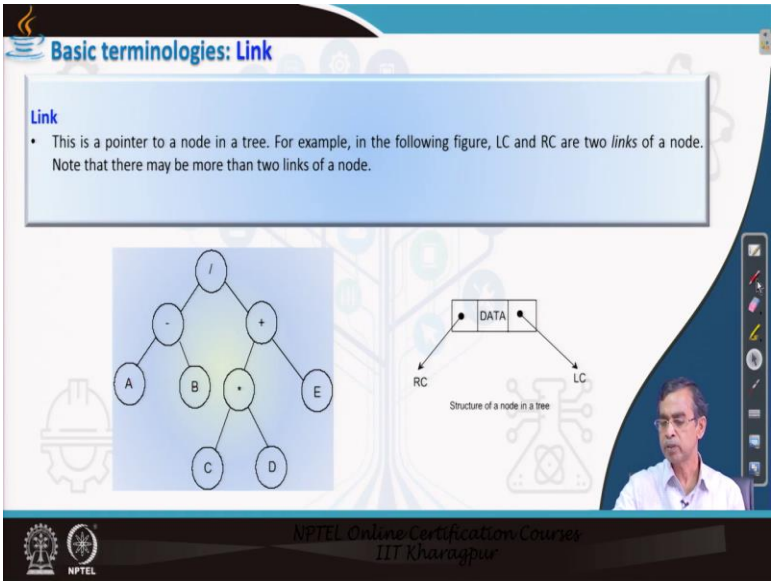
So, 2 nodes may have the same parent, 1 or more node have the same parent, like. And there are some nodes which does not have any children. For example, this node does not have any children and these are the nodes. Now, so there are some nodes which does not have any children they are called terminal node or leaf node. For example, A, B, C, D, E in this example is the leaf node, like.

(Refer Slide Time: 20:33)



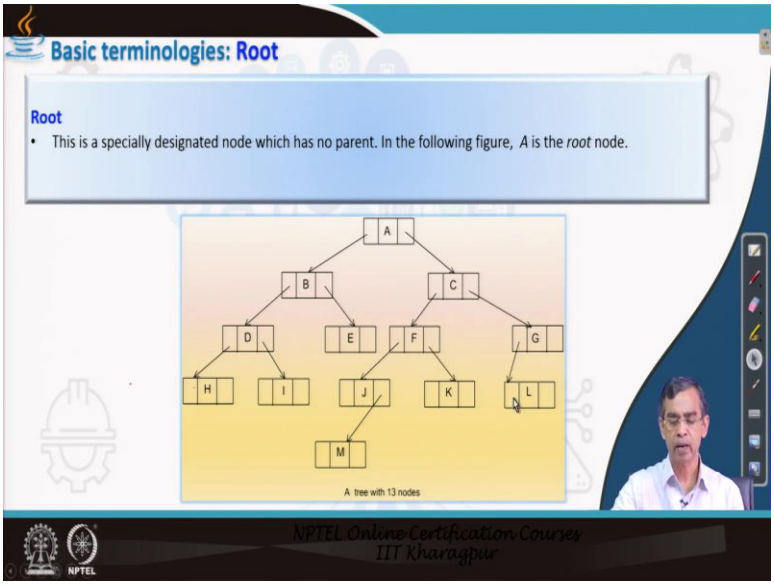
Now, children we have discussed about node, parent and everything left child, right child also we have discussed about.

(Refer Slide Time: 20:40)



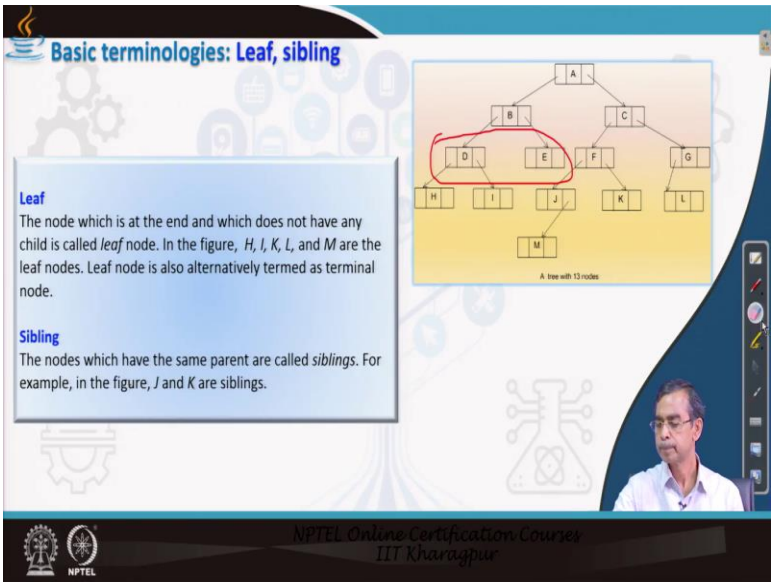
Now, there is another link. Now, we have already discussed that for every node every node rather stores 2 links. They are basically pointer or addressed to their left child and the right child, the link field can be null. For example, for all leaf node, the 2 link field stored null values. If a node can have only 1 children, if it is left children they need has null value there and right children link value is null otherwise. So, it is a concept about link.

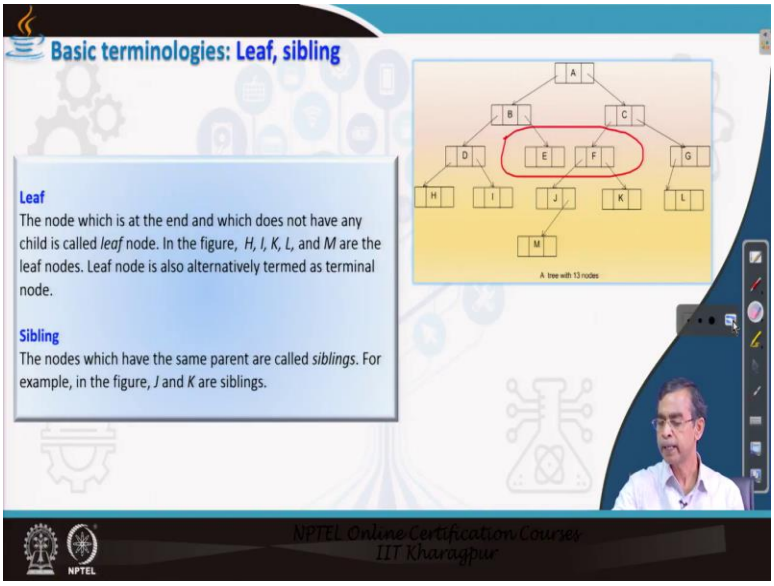
(Refer Slide Time: 21:17)



Then root we have discussed about a special node that node is there and root does not have any parent. And this is an example about storing the different elements into a tree, 30 nodes are stored in different order it is a repeating order of course you can find.

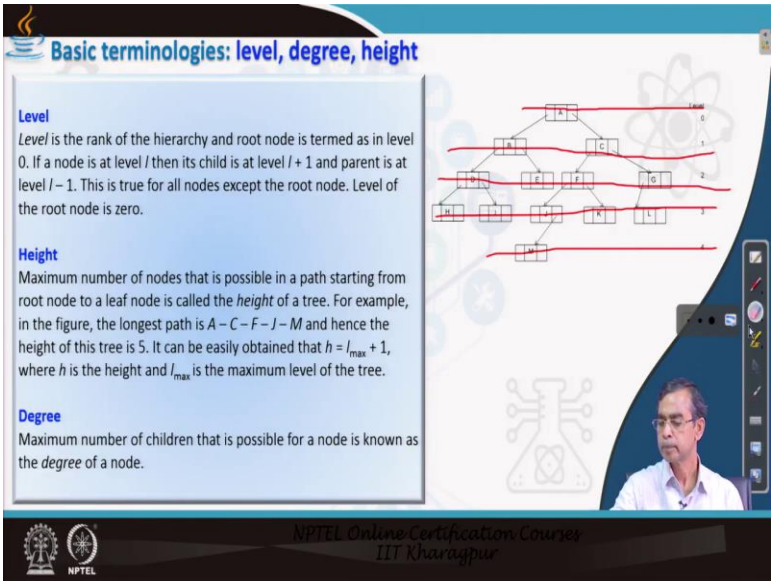
(Refer Slide Time: 21:36)





And then the sibling, sibling basically, the 2 childrens of the same parents is called a sibling of each other. Now, in this example for example, D and E are the sibling. However, however, here in this example, E and F they are not sibling because their parents are different. So, now this is a concept of sibling that is there in the tree.

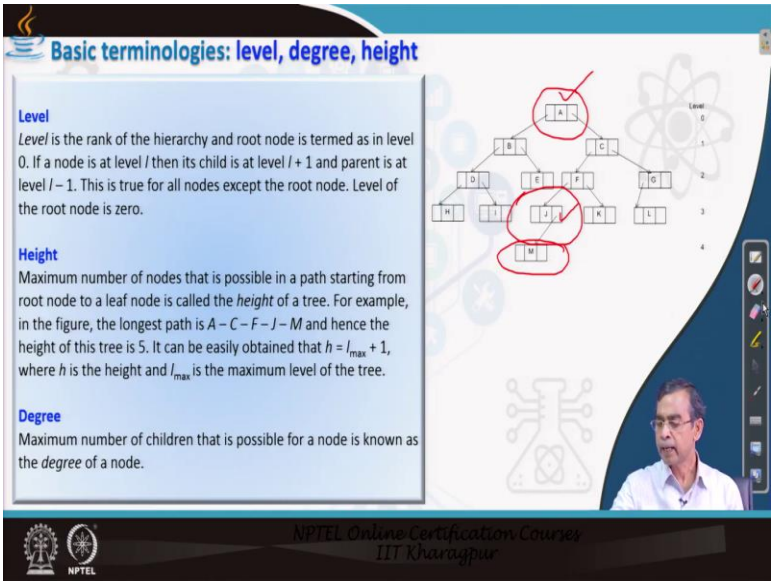
(Refer Slide Time: 22:26)

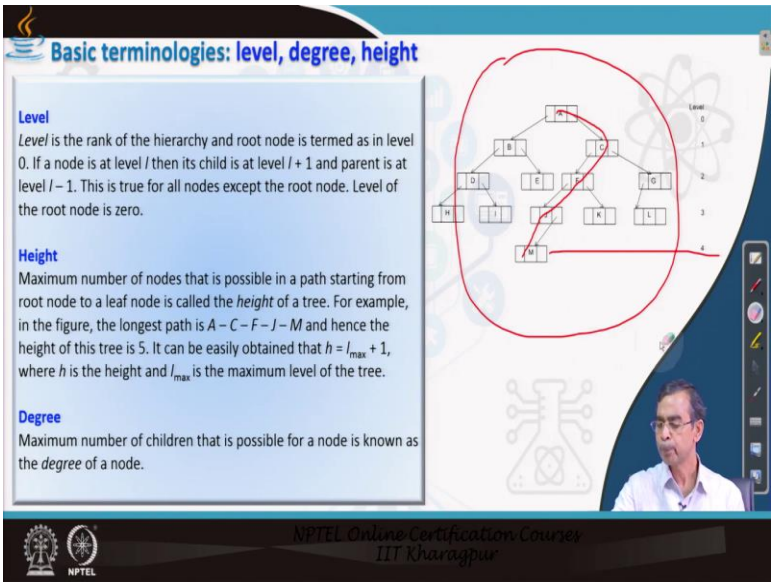


And then there are a few more definitions which are very important regarding level, degree and heights. So, the root, the node is in a level called the 0 level that then all childrens of the root node is called next level, call 1. Then if a node is an level I then its childrens are at level I plus 1. So, these are the all nodes in second level, the third level and this is the nodes in the fourth level.

So, this basically you see the every nodes, they are in a particular level. So, for the binary tree each concerned and the leveling starts from 0 to any value. 0 indicates that the nodes at that root level. Now, here degree every nodes has number of link or sub trees actually, so number of sub trees that a node has it is called degree of this node.

(Refer Slide Time: 23:46)





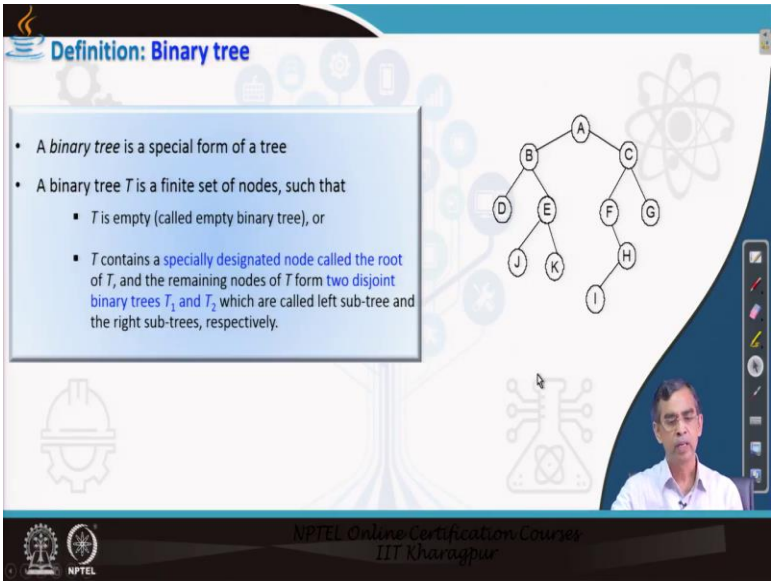
Now, in this example, in this example, here this is a node, what is the degree of this node? It has 2 children. So, degree of this node is 2. On the other hand, if you come here, what is the degree of this node? It has only 1 children so, the degree this node is 1. But if you cannot do this on node, what is the degree? It is degree 0 because it does not have any children. So, all nodes in a binary tree have degree in between 0 and 2 both inclusive.

So, there should not be any tree in a binary, any node in a binary tree having degree more than 2 or less than 0. So, this is the concept of degree and now, let us come to the discussion about height. Now, height is basically defined as the number of nodes to be traverse in a longest path

starting from the root node. Now, here if we see these binary tree this is 1 the longest path there may be more than 2 paths have the longest path also that mean same path which are large.

Now, how many nodes you visit? 1, 2, 3, 4, 5. So, what we can say that, for this binary tree, for this binary tree, the height is 5 and 1 interesting correlation as you can see here in this example also it can reveals that height is exactly the highest level possible for a binary tree plus 1. For example, highest level is basically 4 and then height is basically 5. So, this is just right there is a proof also always it is true that height of a binary tree is the highest level plus 1.

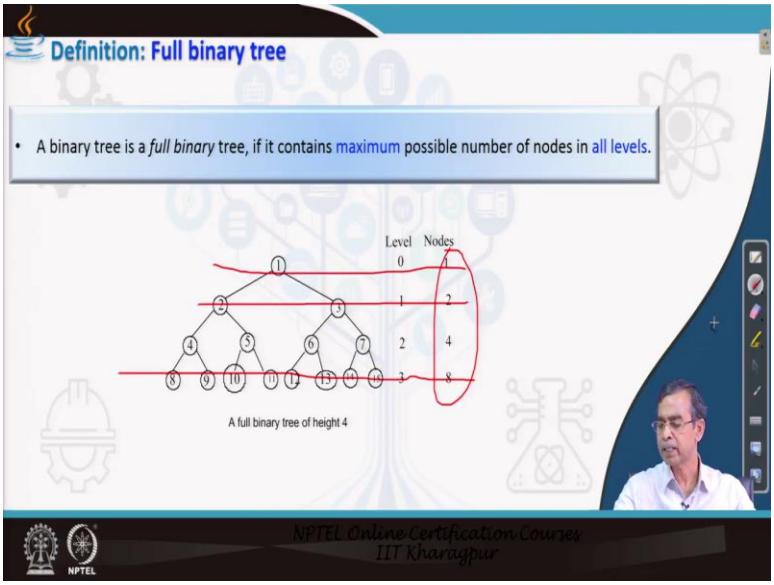
(Refer Slide Time: 25:52)



Then, so binary tree is a special tree it has only 2 sub trees that we have discussed. And all sub trees are different and unique and 1 more important point that you can note is that whether node can store duplicate values or not any node can store any values. So, in that sense you can store but actually from the application point of view a node should not store the duplicate values, all not should be unique.

It because it does not have otherwise any meaningful application of it actually. Anyway so, any node, any. Logically it should not, physically you can store any elements in it actually, but it is an homogeneous data structure like Array, LinkedList and Stack, Queue whatever it is there. If a tree is defined, then it can store in a particular type of things it is there. So, this is the idea about the concept.

(Refer Slide Time: 26:53)

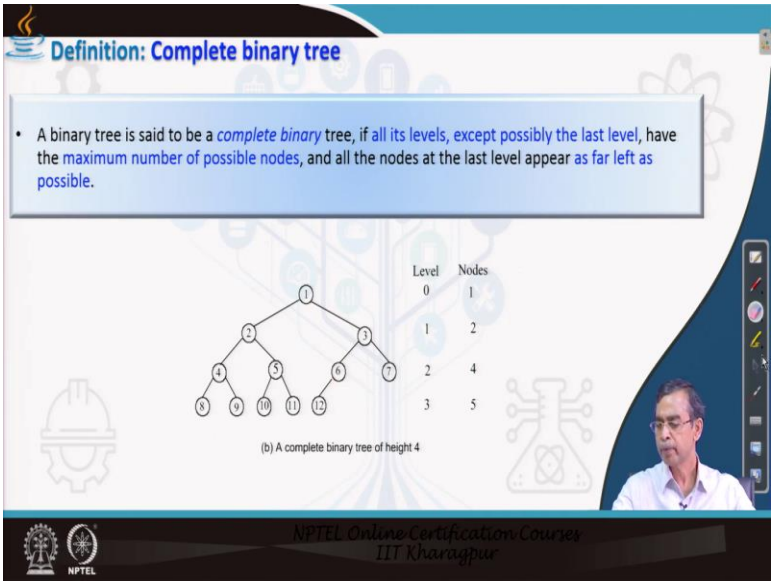


And there are few variations so far the binary is concerned 1 complete is called a full binary tree. A binary tree is called the full if all its levels stored the maximum number that is possible. Now, I am telling you the maximum number that a binary tree at level I is possible is 2 to the power I, where I can change from value 0 to any numbers. Now, for example, the level 0 that is a root contents 2 to the power 0 only 1. So, next level 2 to the power 1, 2 elements, so and so on, as you can see here.

So, level 0 number of nodes 1 and these are number of nodes 2 and the last level which is level 3, and the number of nodes is 8, so 2 to the power 3. So, you see the number of nodes at any level, I is 2 to the power I that level. So, these are peculiar things it is there and so a tree a binary tree is called full binary tree if it contents maximum possible numbers in each level, maximum possible number in Ith level is 2 to the power I that is all.

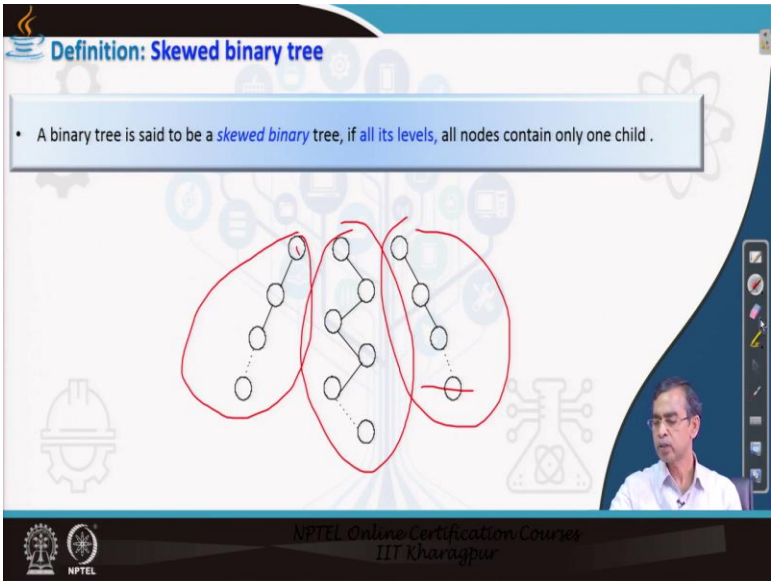
So, this is a peculiar, this is another particular characteristics or property that a binary tree have if it is satisfied then such a binary tree is called full binary tree.

(Refer Slide Time: 28:24)



Now, similarly full binary tree there is a complete binary tree. Complete binary tree just like a full binary tree but only difference is that, except the last level all levels are having the maximum number of nodes possible in that level. So, in that sense a full binary tree is a complete binary tree, but reverse is not true. a complete binary tree is not a full binary tree. So, this is the 2 different what is called that nomenclature or name for a binary is known in the.

(Refer Slide Time: 28:59)



There are another few I mean type of binary tree is there, it is called a skewed binary tree. As you say as you have already learned about that a binary it can have 0 children, 1 or 2 children,

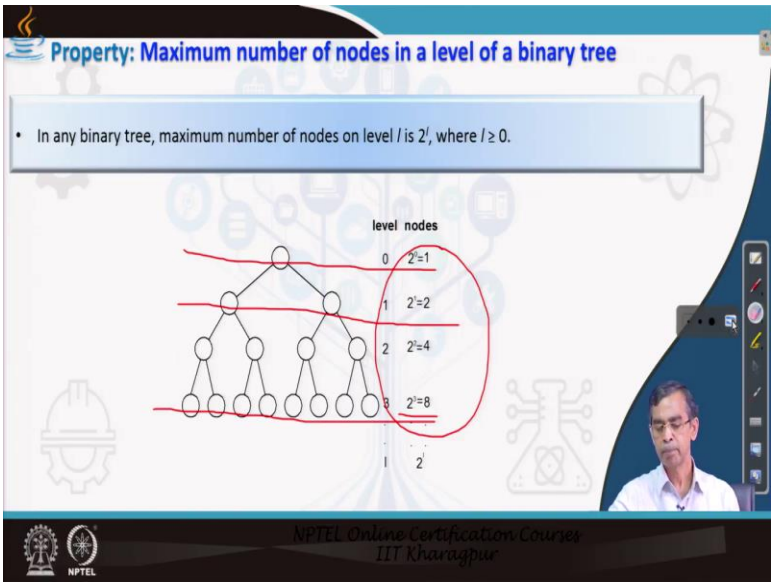
but if a particular binary tree has always all nodes content only 1 children and then such a binary tree is called the skewed binary tree.

So, there is a typically skewed binary here you can see in this binary tree, all nodes has their only the right children but they do not have any left children. Here, the left children, all nodes have except the leaf node of course. Now, here alternatively 1 children has left and then right and so on. But here all 3 trees that we have demonstrated here, all node except the last node or terminal node or leaf node has maximum 1 children of it. So, such a tree called skewed binary tree.

(Refer Slide Time: 30:05)

Now, let us first discuss about properties of binary tree, we will discuss few important properties of binary tree and these properties are very important, because it has many theoretical questions particularly those are the students interested for GATE or competitive examination. Many questions asked from this concept properties of binary tree. Binary tree owns many properties, very interesting properties.

(Refer Slide Time: 30:37)

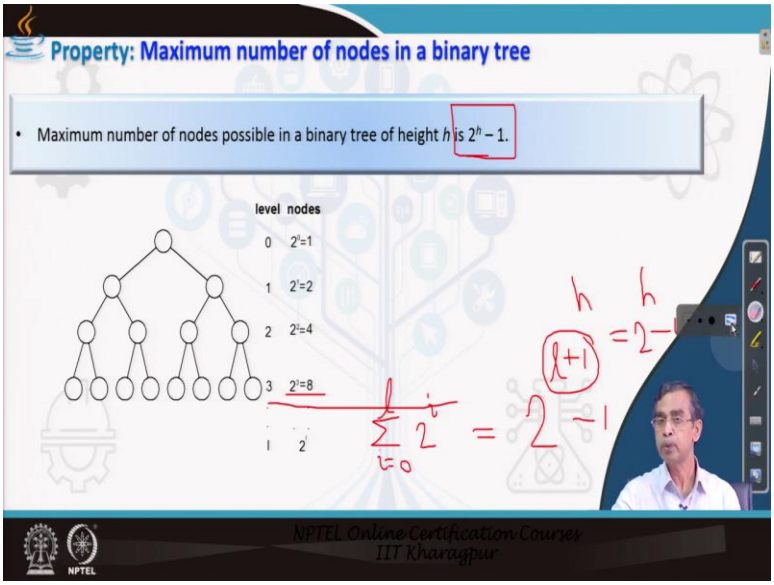


So, let us learn about the different properties that a binary tree possess and then the first property; this is basically regarding the maximum number of nodes in a level. We have already learned or discussed about it, that there is there is a basically theorem we can say very basically every properties can be expressed in terms of theorem it is always true actually, that is why their property or their theorem it is called.

So, this property says that in any binary tree the maximum number of nodes in level l is 2 to the power l higher the level can varies from 0 to any numbers. So, this is clearly visible from here that, ok, this is the level 0 to 2 to the power 0 that mean number of nodes in that level to level 1, 2 to the power 1, 2, and this is basically 3, level 3 number of nodes is this 1, so this way. So, total number of nodes that also can be counted by summing up all the number of nodes in the level it is there.

So, for a full binary tree, it is basically sum up all the nodes in each level actually it is there, anyway so, those things is a separate issue there. But this property is very useful to count the total number of nodes that is there in a particular binary tree. So, this is the first property.

(Refer Slide Time: 32:02)



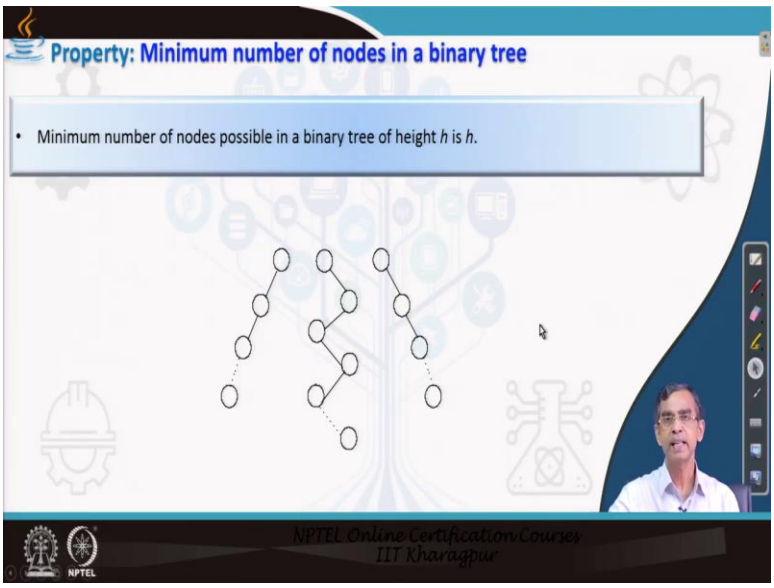
The second property that is again interesting property is that this property is related to what is the maximum number of nodes that is possible in a binary tree. Now, obviously if a binary tree is a full binary tree, then it can contents maximum number of elements because all levels are occupied with all elements in it, all nodes in it there and so, maximum number of nodes that is possible in a binary tree if its height is 8, then this formula evaluates 2 to the part 8 minus 1. Now, that can be clearly understandable from here. So, it is basically summation.

So, I equals to 0 to l, l is the maximum number of level in this case l is 3 and 2 to the power I. So, this basically expression can lead to 2 to the power l plus 1 minus 1 and you know l plus 1 is basically height. So, that is it this formula stands as 2 to the power h minus 1. So, this is basically the formula for the maximum number of nodes that is possible for a binary tree. So, it is possible when binary tree is full binary tree.

Now, if I asked you the question, what is the minimum number of nodes that is possible for a binary tree? Then that question can be answered again. So, minimum number of nodes possible for a binary tree if all levels contents minimum number of nodes. Now, all level content minimum number of nodes possible only if it is skewed binary tree. So, in case of skewed binary key if its height is h that means the maximum number of nodes that is possible there is h is there.

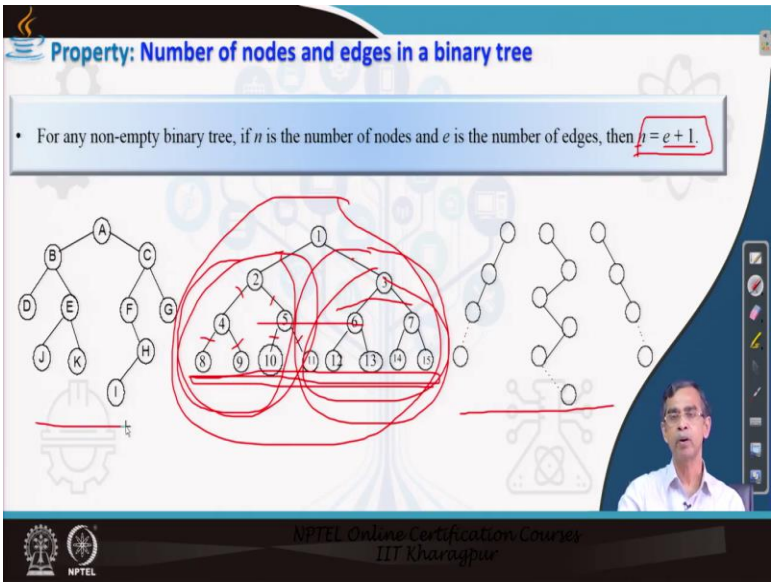
So, if n number of nodes are there in a binary tree its height is h or its height is n if it is a skewed binary tree actually that you can verify again. Now, again let us continue 2 more properties there.

(Refer Slide Time: 34:32)



And ok this is the property that we have just now told about the minimum number of nodes that is possible in a binary tree, it is possible for a skewed binary tree and if n number of nodes are there, there is a maximum, minimum number of nodes, basically is the height, so in this case, the height is basically the number of nodes it is there.

(Refer Slide Time: 34:52)

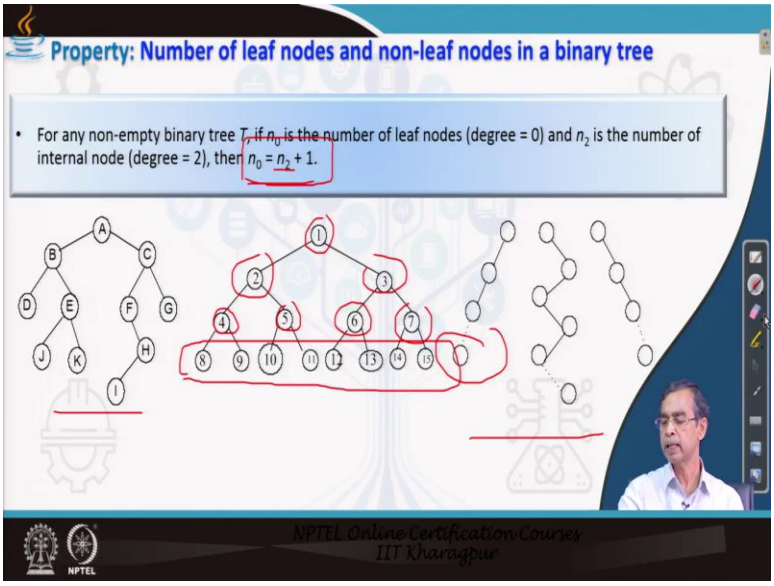


Now, there is another property is a number of nodes and edges in a binary tree. Now, there is a very interesting property that it holds that number of nodes is equals to total number of edges plus 1 it is always there. Now, for example, in this binary tree if you consider total number of

nodes is 15 and total number of links or edges are there. Now, these are the last leaf nodes, they do not have any edges there or links there, but all others now here if you see in this particular tree, this has these are the 1, 2, 3, 4, 5, 6, 7.

So, this part has 7 and this also part has a 7. So, 7 plus 7 is 14, and then this 1 is a 1. So, total actually here 15 total number of nodes 15 and number of edges is basically 7. Here is a 7 here and in this party 7 edges are there. So, this formula always valid in the total number of nodes is equals to 1 extra than that number of edges it is there. Now, this formula is also true for this as well as true for this you can verify of your own. So, this is an interesting property.

(Refer Slide Time: 36:25)

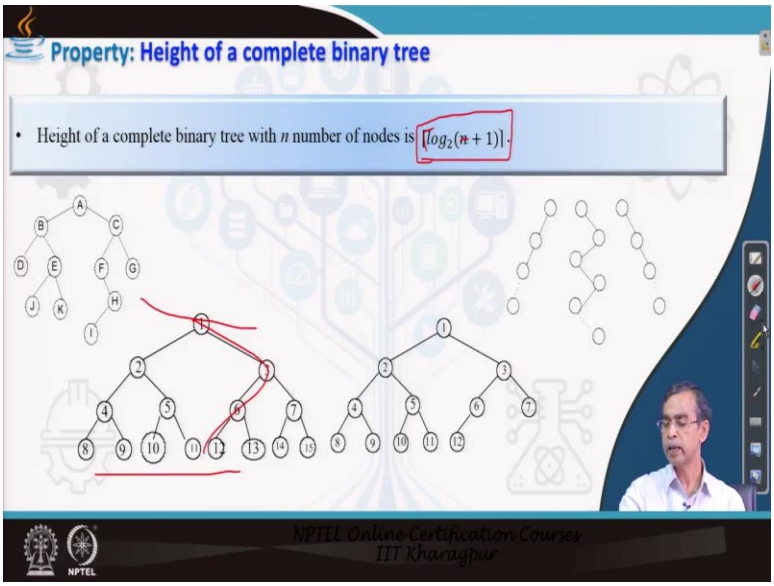


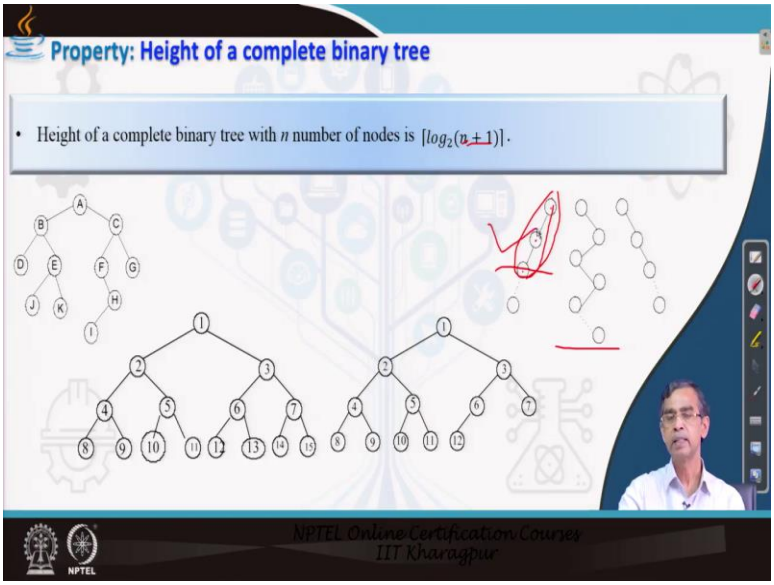
Now, let us see the other property and here is a property related that is a very interesting property and valid for any binary tree. This says that n 0 equals n2 plus 1 what is the n0? n0 is number of terminal node or leaf node and n2 is a number of nodes having degree 2 that means the nodes which has both children. It is always 2 now let us come to here.

Now, in this example as you can see total number of (())(36:55) nodes that we see, it is 8 in this node because 8 nodes are there, n0 is 8 in this example. Now, let us see how many nodes having degree 2. So, this node 1 this node 2, 3, 4, 5, 6 and 7. So, 7 nodes are having degree 2, so n2 equals to 7 and n0 is 8. So, this formula valid. Now, here also you can see n0 is 1 and n2 is 0, so that is also valid for all skewed and here also you can verify of your own that the formula is also true for any tree.

Now, these are the few trees I have given you can draw any binary tree and you can verify that this formula holds good for any type of tree. So, this is another interesting property that the binary tree follows.

(Refer Slide Time: 37:55)



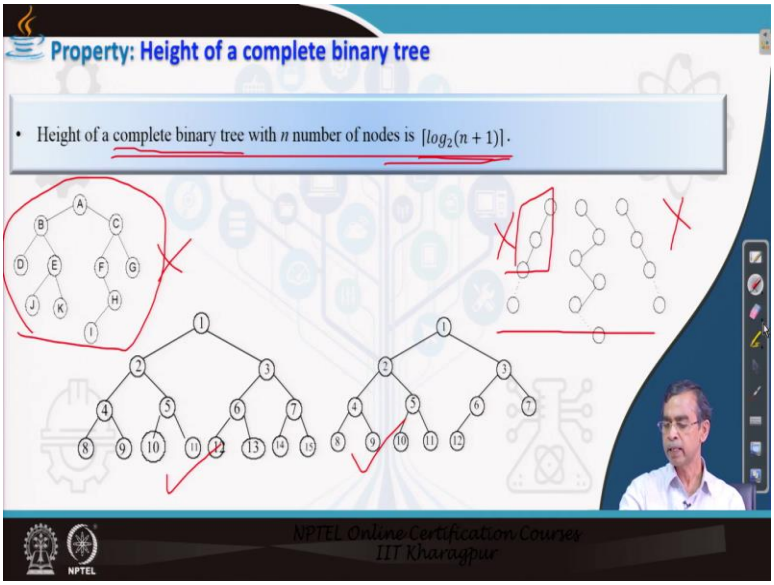


Now, height of a complete binary tree now there is a formula this formula again you can check it log2 n plus 1 is the formula where it is basically it is the ceiling. So, n plus 1 its ceiling value it is there. Now in this example for example, this is a complete binary tree of course, the number of nodes is n, so here are 15, so 15 plus 1 it is 16 and log 2 16 is 4, so height of this binary it is basically 4. So, this is ok, fine, true for this tree. Now, for this tree again we can verify, for this

tree again we can verify, for this tree again. Now, number of nodes is basically for example here, 3, so 1, 2, 3, 3 nodes after this 1, so here, so n is 3, so it is 3, and then 1, 3 plus 1 is a 4.

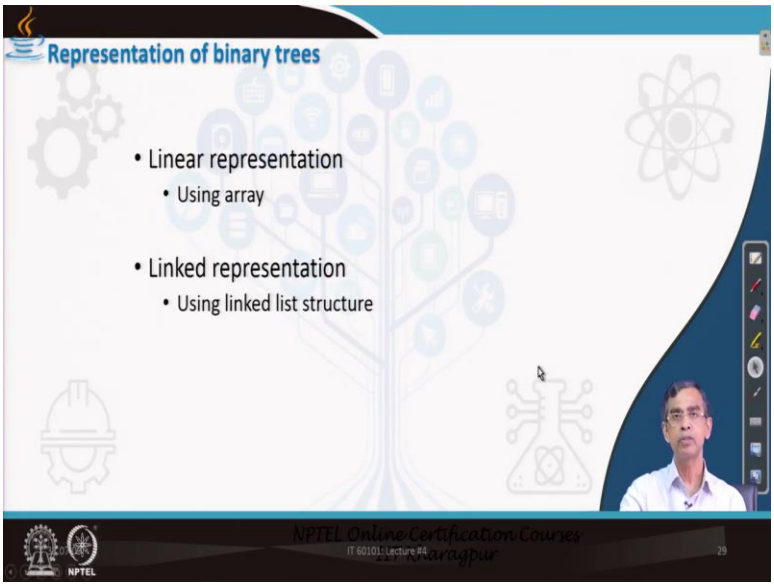
Now, log 2 this basically log 2 or 2 that basically are log 2 2 is 1, so height is 2 in this case, height is, floor. It is a 3, right. So, 3 plus 1 is a 4, log 2, 4 is basically ceiling of 4 is 3. So, this way it is height is 3. So, you can verify this formula for this tree as well as, and for any tree value, it valid actually. I am sorry, this formula is not valid. This formula is only valid for a complete binary tree as this is not a complete binary tree.

(Refer Slide Time: 39:39)



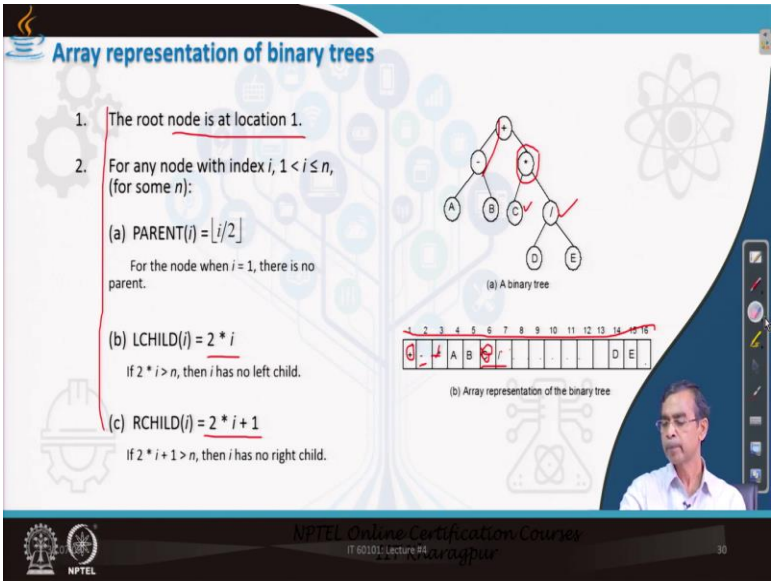
So, this formula is not valid. So, that is, that is why we could not find the height is for this and this is also not a complete binary it you can see this formula you cannot valid. So, for this binary tree this formula is not valid. But this is a complete binary at this formula is valid and this is also a complete binary it so this formula is valid. So, this formula is valid only for complete binary tree. So, this is another interesting property, that you can have it.

(Refer Slide Time: 40:17)



Now, there is another property. We will just discus about how a binary tree can be stored in a memory, there is a 2 representation, linear representation that means an Array can be considered to store a binary tree. Let us have a quick idea about how a binary tree can be stored.

(Refer Slide Time: 40:26)



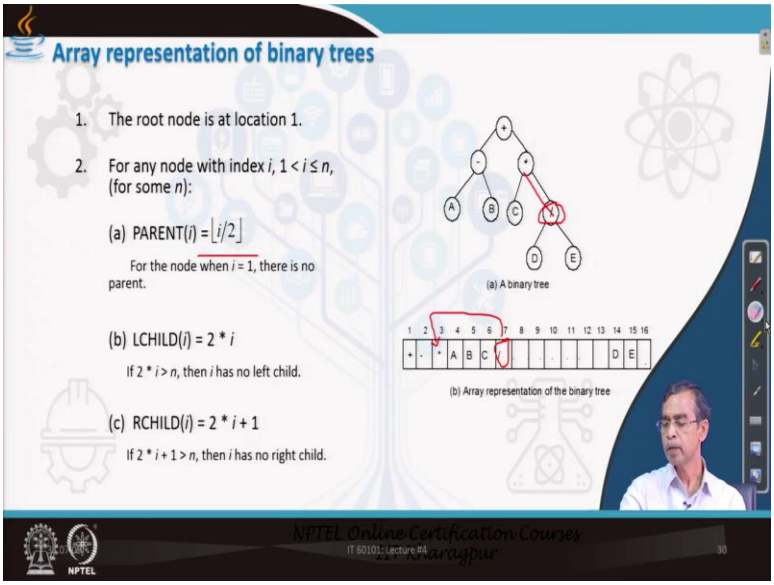
Now, in this tree, this is a tree there is a rule, these are the rule actually that you can follow, so that the link can be automatically mentioned without storing them actually. So, this is very good idea about that without any maintaining the link information but link can be ensured. What is idea? The idea is that, say index start from 1, 2 and in this example.

So, root node is at location 1, so always it is there, root should be stored in the first location. And then any node whose rood is at 8 location, its left child will be stored in 2 star I location, and its right child will be stored in 2 star I plus 1. For example, this is the star, star is stored in the location 3.

Then its right child C will be stored in, its left child will be stored in 6, it is a 6 and right child will be stored in the 7. So, here is a, this is a right child and this is a left child whatever it is stored there. Anyway, so, this is the idea it is there and so, root node is 1, so it this is a right child, you can say right child is stored here in 2 star.

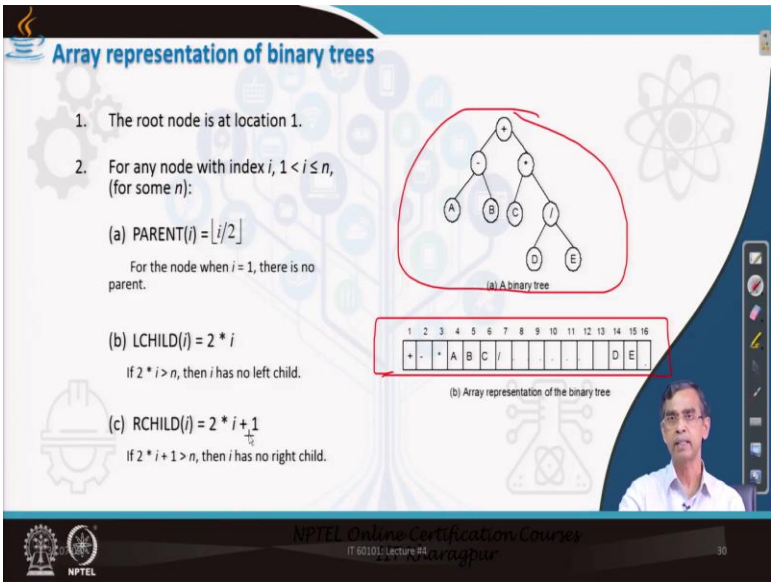
Right child, this is a 2 star, right child is this 1 and left child, this is a left child. Left child is 2 star I, so this 1, 2 star I and its right child is there; right child is star, it is 2 star I plus 1. So, this formula that you can, you can verify.

(Refer Slide Time: 42:28)



And another important property that you can check it, if any node it is in the i th location, then its parent can be tagged at this formula, it is basically 4 of i divided by 2. For example, this is stored in seventh and i divided by 2, 7 by 2 means 3.5 and 4 value is 3 so, it root is in the third location. So, it is a root location as you see, i third location is this 1.

(Refer Slide Time: 43:07)

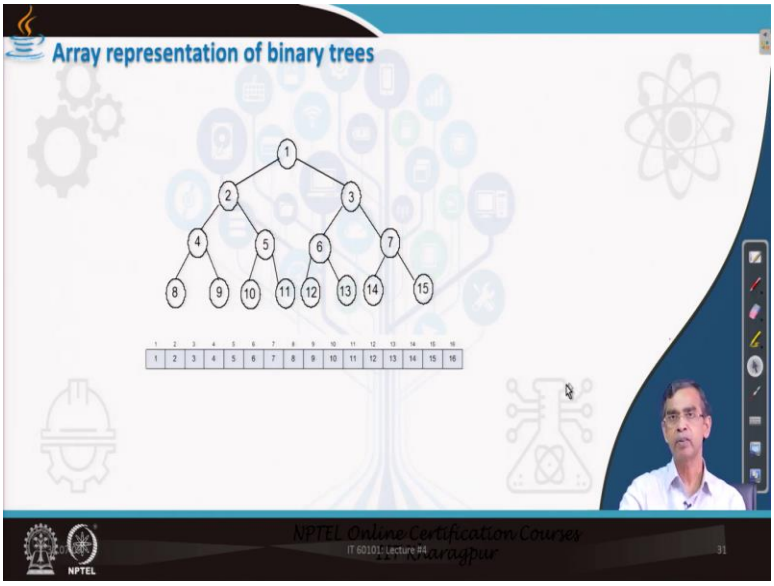


Now, following this mathematical formula, following this mathematical formula you can check for any node in this tree can be stored this way and this is an entire Array that is required to store

this 1. Now, this is very important 1 observation, here you see automatically the tree can be stored into Array but without storing any link information.

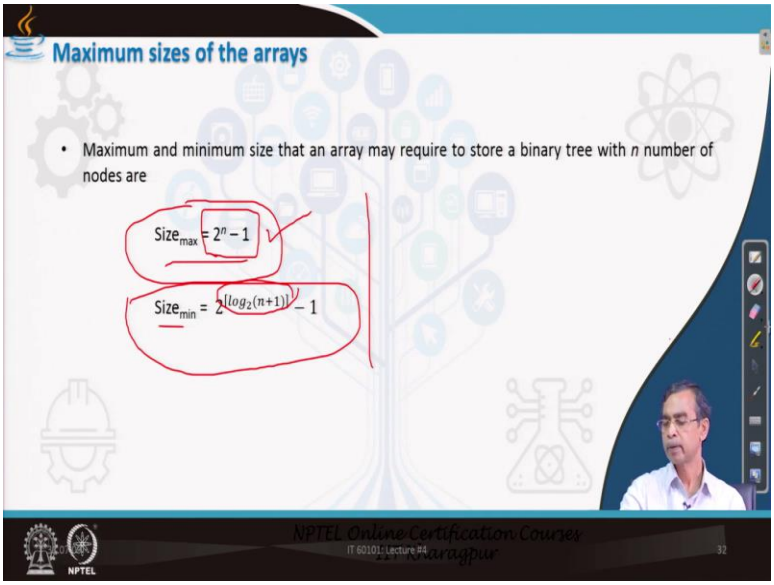
That mean what is a link or link information mean left child link and right child link or address of the right child node or left child node it is there. Now, again that formula considering this formula consider 1 to 10 and index starts from 1, but if its the index starts from 0, this formula needs little bit modified by the accordingly so, 2 star i this is fine, but here 2 star i plus 1 instead 2 star i minus 1 and accordingly, so that formula you can verify again. If index start from 0 and index start from 1 the formula will be little bit changed only this formula will be changed like this one.

(Refer Slide Time: 44:16)



So, this is about Array representation and here is an example about another binary tree and its Array representation; again you can check the formula that we have followed to do that there and I should advise you to repeat the same, same but for starting with 0 and this 1 and then accordingly you can verify it.

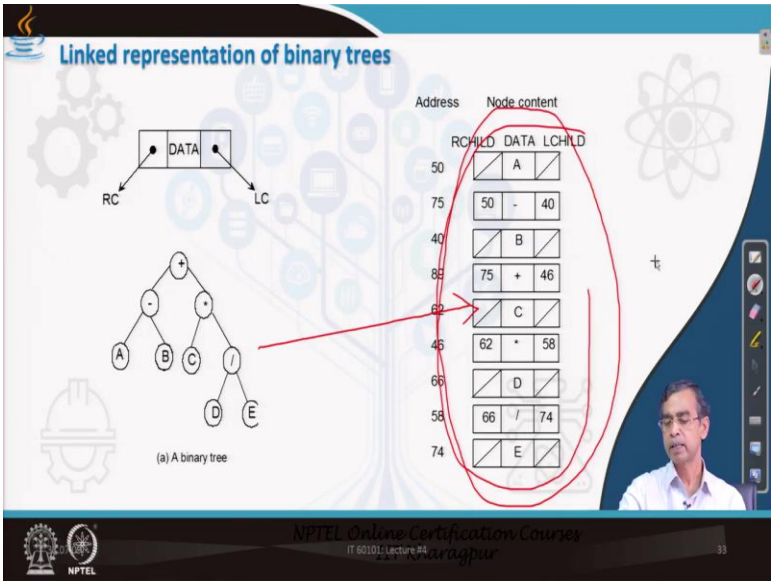
(Refer Slide Time: 44:39)



Now, here is a 2 more properties that is very interesting property again, it says that what is the size of the Array that is required to store a binary tree and this formula says that the maximum size that is required to store a binary tree is equals to 2 to the power n minus 1 and this is the case when binary tree is skewed. On the other hand, minimum size that is required to store a binary tree it is basically when the binary tree is complete and this formula says that minimum size, which is equals to 2 to the power, this is basically the height formula; 2 to the power height minus 1, log 2 n plus 1 this 1.

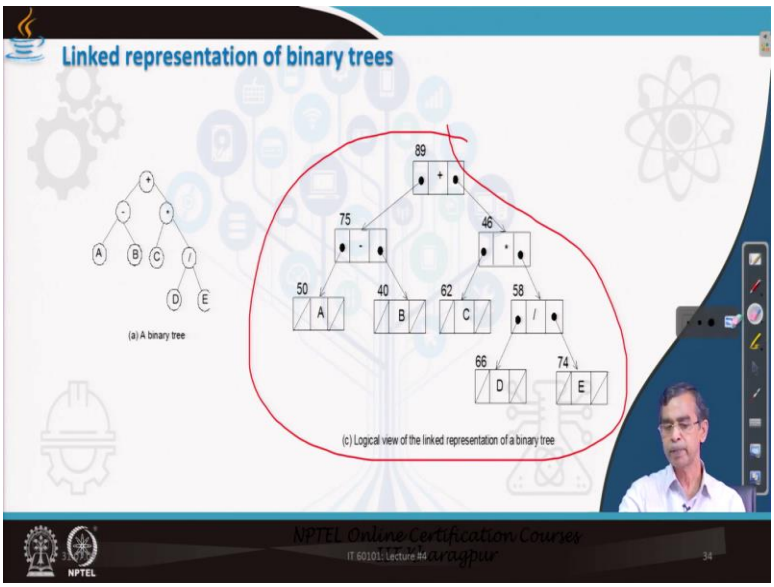
So, this is basically size that is required to store a, size of the Array that is required to store a binary tree. Now, this is maximum and that is a one advantage, there is advantage that for a skewed tree if you want to store it require huge Array to store it. Although many location of the Array remains vacant actually.

(Refer Slide Time: 45:41)



Now, so, these are the minimum and maximum size. Now, let us come to the link representation of the binary search tree, it is basically you have to store the pointers there. Now, here is the idea about different nodes, how the address can be stored, you can check that this tree and this memory mapping also can be readily about it, its nodes are there and then pointers it is there. So, this is basically memory representation and this you can see all the elements or nodes are stored non contagious fashion. But in case of Array the elements are stored in a contagious location.

(Refer Slide Time: 46:19)



So, this is a memory representation, this are another example of link representation actually it takes place in memory actually. This is basically the logical view of a binary tree when it is stored and physical view that we have shown in the previous slide actually. So, this is a binary tree representation using memory, I mean using linked information and then Array also we have discussed about it and there are many more things that is discussed you can find in this book, it is very nice it contains lot of discussion, nicely and I should advise you to go through if you want to learn many things about the tree structure, you can find chapter 7 there. Thank you, in the next lecture we will discuss about the different operations that we can apply on the binary tree, Thank you, Thank you very much.