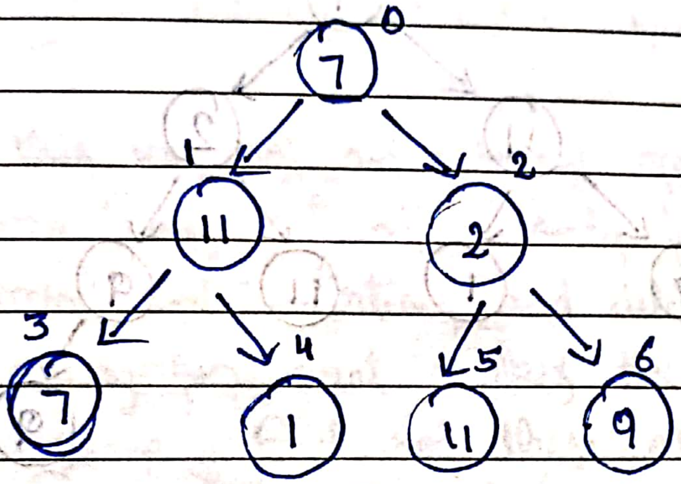
Representation of a Binary Tree:

→ Array representation of Binary trees:

Arrays are linear data structures and for arrays to function, their size must be specified before elements are inserted into them. And this counts as the biggest demerit of representing binary trees using arrays.

Suppose you declare an array of size 100, and after storing 100 nodes in it, you cannot even insert a single element further, regardless of all the spaces left in the memory. Another way you'd say is to copy the whole thing again to a new array of bigger size but that is not considered a good practice.

Suppose we have a binary tree with 7 nodes.



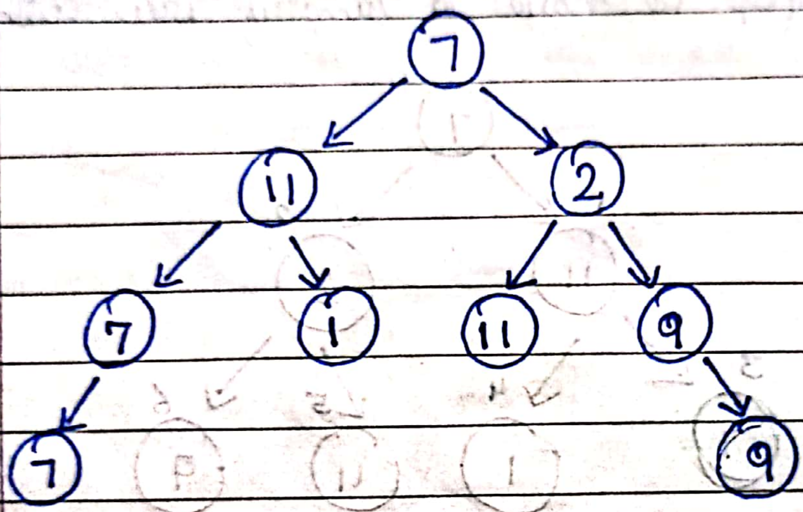
And there are actually a number of ways to represent these nodes via an array. But the most convenient one where we traverse each level starting from the root node and from left to right and mark them with the indices these nodes would belong to.

And now we can simply make an array of length 7 and store these elements at their corresponding indices.

0	1	2	3	4	5	6
7	11	2	7	1	11	9

→ Now, what if the cases where the binary is just not perfect! What if the last level has distributed leaves?

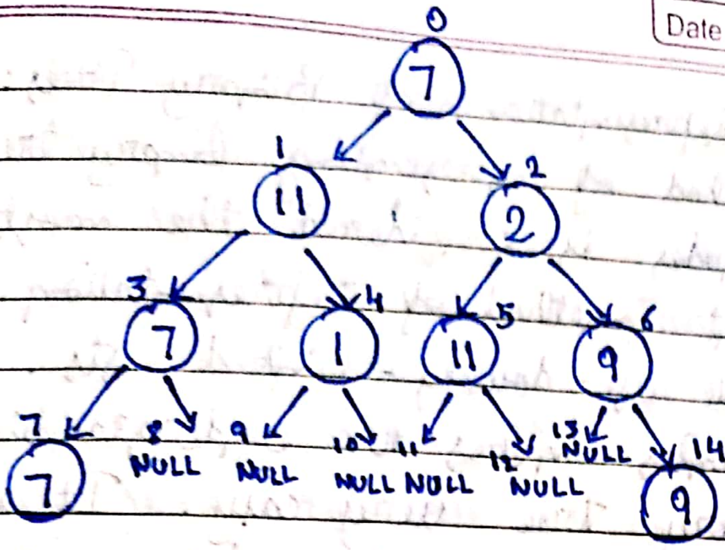
~~Ex~~ : Let's consider a binary tree with 9 nodes, and the last two nodes on the extremities of the last level.



Here while traversing we get stuck at the 8th index. We don't know if declaring the last node as the 8th index element makes it a general representation of the tree or not. So, we simply make the tree perfect ourselves. We first assume the remaining vacant places to be NULL.

And now we can easily ~~mat~~ ~~F~~ mark their indices from 0 to 14.

And the array representation of the tree looks something like this. It is an array of length 15.



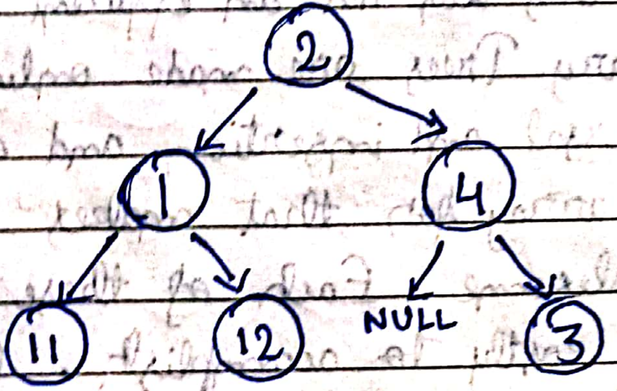
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
7	11	2	7	1	11	9	7							9

→ But this was not an efficient approach. Like Binary Trees are made only for efficient traversal and insertion and deletion and using an array for that makes the process troublesome. Each of these operations becomes quite costly to accomplish. And that size constraint was already for making things worse. So overall, we would say that the array representation of a binary is not a very good choice.

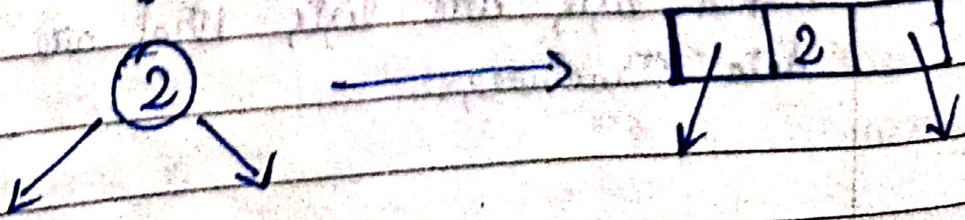
→ We have another method to represent binary trees called the linked representation of binary trees. Don't confuse this with linked list. And the reason why, is because linked lists are lists that are linear data structures.

→ Linked Representation of Binary Trees:

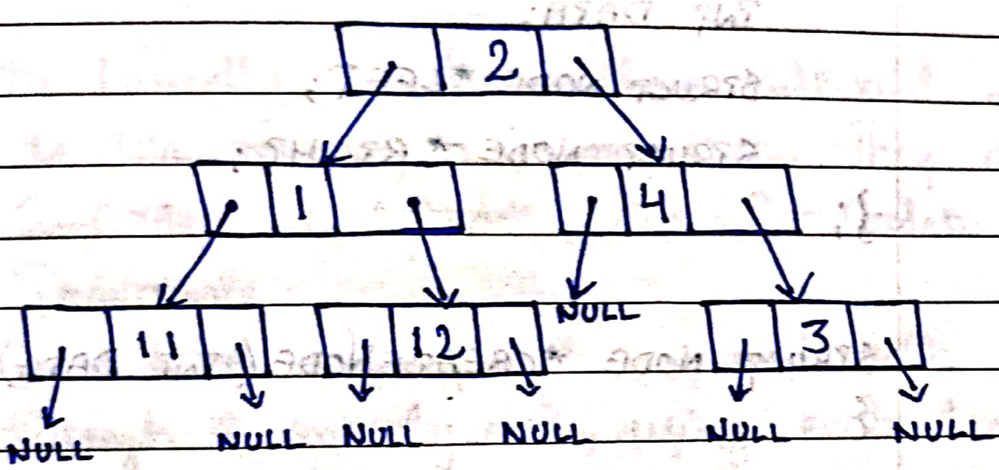
This method of representing binary trees using linked nodes is considered the most efficient method of representation. For this, we use doubly-linked lists. Using links makes the understanding of a binary tree very easy. It actually makes us visualize the tree even. Suppose, we have a binary tree of 3 levels.



A doubly linked list helped us traversing both to the left and the right. And using that we would create a similar node here, pointing both to the left and the right child node. Follow the below representation of a node here in the linked representation of a binary tree.

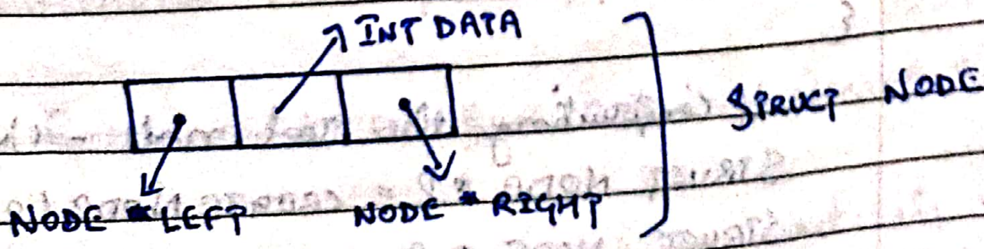


We can see how closely this representation resembles a real tree node, unlike the array representation where all the nodes succumbed to a 2D structure. And now we can easily transform the whole tree into its linked representation which is just how we imagined it would have looked in real life.



→ What are these nodes?

They are structures having three structure members, first a data element to store the data of the node, and then two structure points to hold the address of the child nodes, one for the left, and the other to the right.


INT DATA
STRUCT NODE
NODE *LEFT
NODE *RIGHT