http://www.cinstitute.co.in
https://youtube.com/tarunsir
+917742817777

Representation of binary tree

We can represent a binary tree in the following two ways:

1. **Array representation**
2. **Linked list representation**

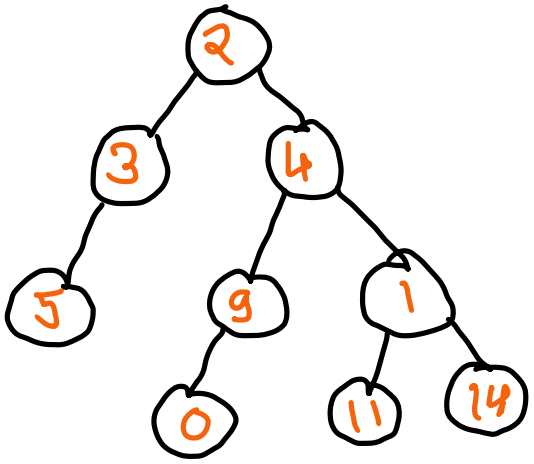
Array Representation

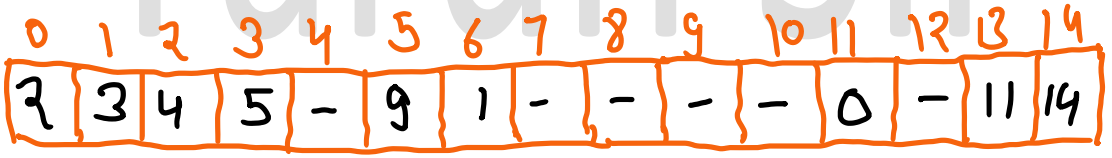
This is also called a static representation or implicit approach. In this approach, we use an array to store the actual tree. The memory, is allocated in advance to store the tree. The size of the tree is restricted because static allocation is used here. In this representation, the nodes are stored level by level. The root node of the tree is stored at the first position, at the 0 index in the array.

If a node is stored at the **N** index, then its left child is stored at the **2*N+1** index and its right child is stored at the **2*N+2** index.

To store all the nodes of a binary tree of height **H** we need an array of size 2^{H}**-1.** For example, if a tree has a height of 3 then we need an array of 2^{3}**-1,** size 8.

Consider the following tree and its array representation:

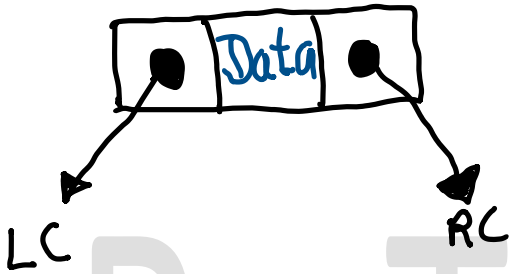




http://www.cinstitute.co.in
https://youtube.com/tarunsir
+917742817777

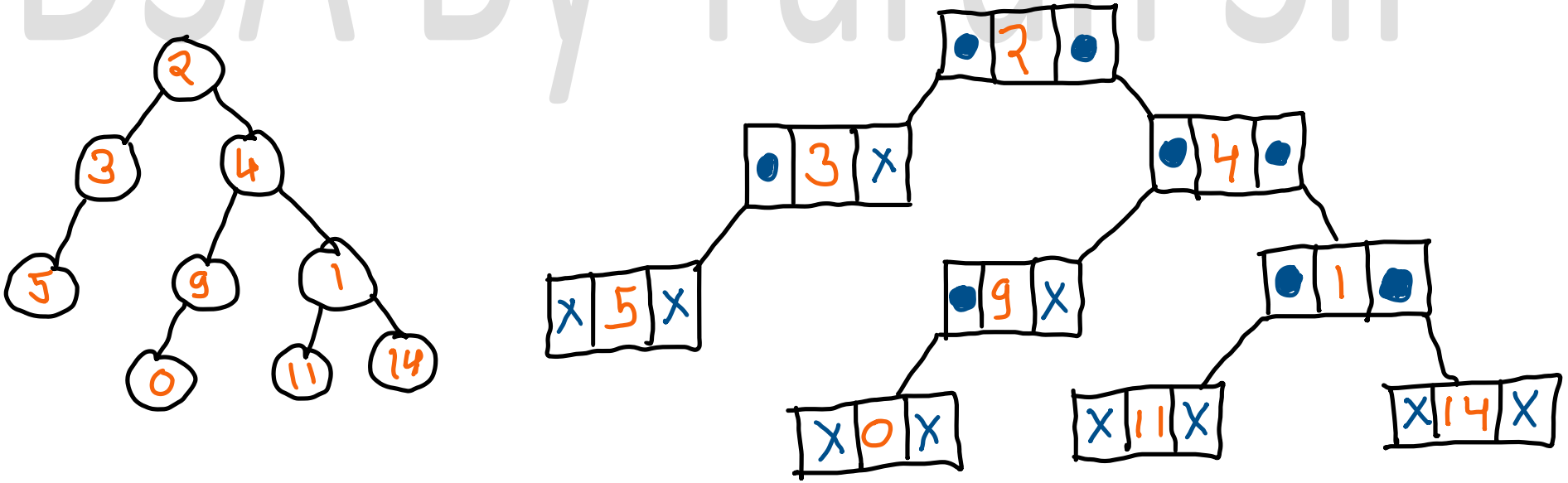
Linked representation of a Binary Tree

In this representation, we use nodes to store the binary tree in memory. The linked representation assumes the structure of a node to be as shown below:



LC and RC are two pointers while Data is the information content represented by the tree node. LC points to the address of the node which stores the Left Child and RC points to the address of the node which stores the Right Child of the node.

Consider the following tree and its Linked representation:



http://www.cinstitute.co.in
https://youtube.com/tarunsir
+917742817777

Array Representation	Linked Representation
Any node can be accessed from any other node by calculating the index and this is efficient from the execution point of view.	We can't access any node directly because there are no indexes in the Linked Representation.
Here only data is stored without any pointers so less memory is required.	The linked representation requires more memory as compared to the array representation. It requires extra memory to maintain pointers. Some pointers may have a null value but still, they require extra space.
In programming languages, where dynamic memory allocation is not supported (such as BASIC, FORTRAN), array representation is the only means to store the tree.	The linked representation can be used only in such languages which support dynamic memory allocation like C, C++, and Java.
A lot of memory can be wasted in array representation. Other than the full binary tree, the majority of the array entries may be empty.	We do not need entries for the nodes which do not exist.
It allows only static representation. It is no way possible to enhance the tree structure if the array size is limited.	It allows dynamic representation, so we can add or delete nodes whenever needed.
Inserting a new node into the tree, or deleting a node from the tree are inefficient in array representation because these require considerable data movement in the array.	Inserting or deleting a node is easier and consumes less time in linked representation as we do not shift data.