**MODULE - 3**

**Syllabus: -**

**Trees:** Basic terminologies, forest, tree representation (using array, using linked list). Binary trees - binary tree traversal (pre-, in-, post- order), threaded binary tree (left, right, full) - non-recursive traversal algorithms using threaded binary tree, expression tree. Binary search tree- operations (creation, insertion, deletion, searching). Height balanced binary tree – AVL tree (insertion, deletion with examples only). B- Trees – operations (insertion, deletion with examples only).

**Graphs:** Graph definitions and Basic concepts (directed/undirected graph, weighted/un-weighted edges, sub-graph, degree, cut vertex/articulation point, complete graph, simple path, simple cycle). Graph representations/storage implementations – adjacency matrix, adjacency list, Graph traversal and connectivity – Depth-first search (DFS), Breadth-first search (BFS) – concepts of edges used in DFS and BFS (tree-edge, back-edge, cross-edge, forward-edge), applications.

**Basic Terminologies of Tree**

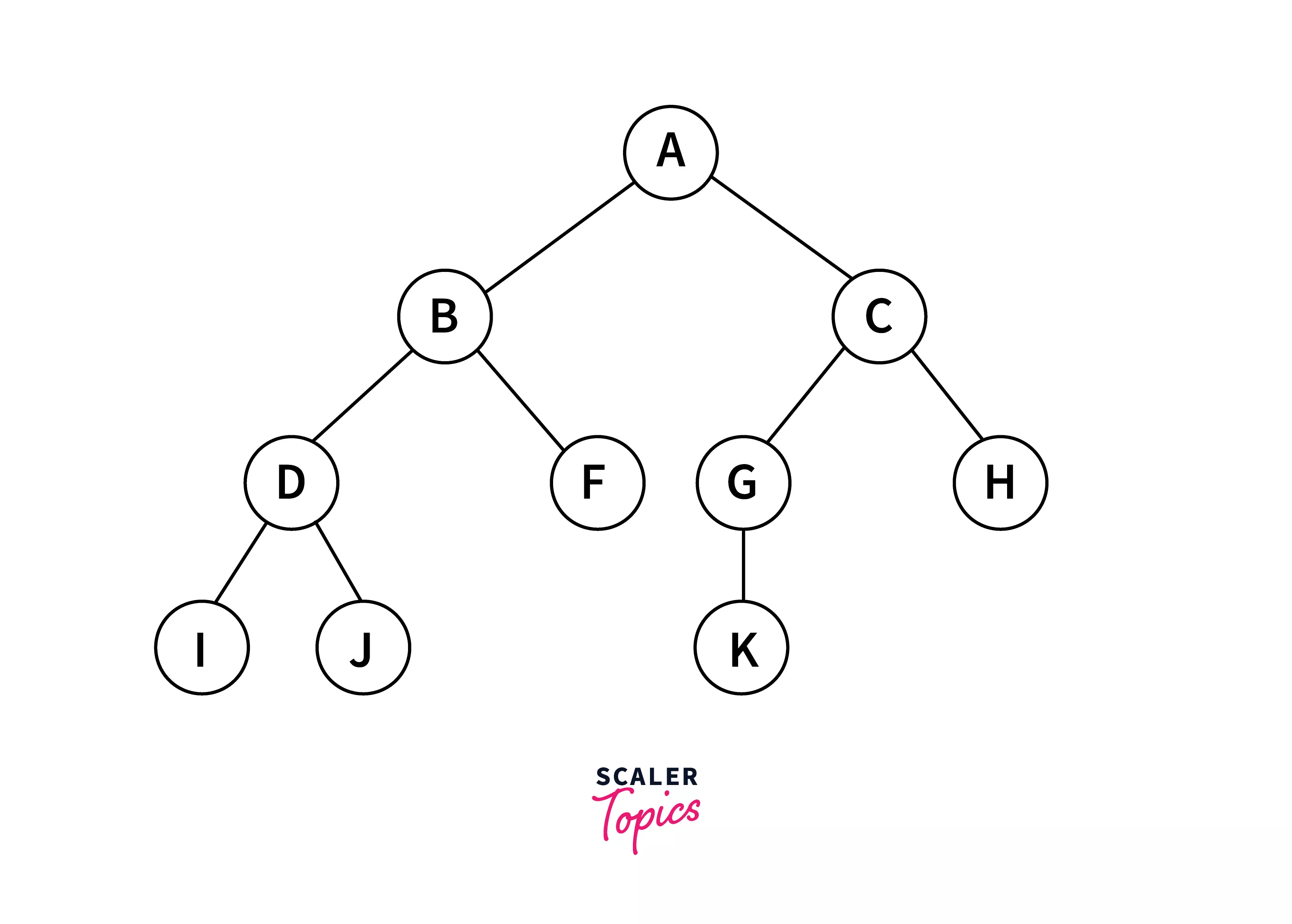


1. **Parent Node:** The node which is a predecessor of a node is called the parent node of that node.**{B}** is the parent node of **{D, E}**.
2. **Child Node:** The node which is the immediate successor of a node is called the child node of that node. Examples: **{D, E}** are the child nodes of **{B}.**
3. **Root Node:** The topmost node of a tree or the node which does not have any parent node is called the root node. {A**}** is the root node of the tree. A non-empty tree must contain exactly one root node and exactly one path from the root to all other nodes of the tree.
4. **Leaf Node or External Node:** The nodes which do not have any child nodes are called leaf nodes. **{K, L, M, N, O, P, G}** are the leaf nodes of the tree.
5. **Ancestor of a Node:** Any predecessor nodes on the path of the root to that node are called Ancestors of that node.**{A,B}** are the ancestor nodes of the node**{E}**
6. **Descendant:** Any successor node on the path from the leaf node to that node. **{E,I}**are the descendants of the node **{B}.**
7. **Sibling:** Children of the same parent node are called siblings.**{D,E}** are called siblings.
8. **Level of a node:** The count of edges on the path from the root node to that node. The root node has level **0**.
9. **Internal node:** A node with at least one child is called Internal Node.
10. **Neighbour of a Node:** Parent or child nodes of that node are called neighbors of that node.
11. **Subtree**: Any node of the tree along with its descendant.

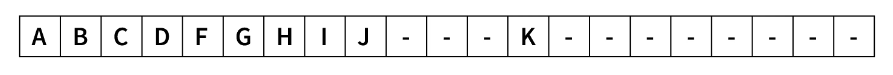
**Array representation of Binary trees**

The binary tree can be represented using an array of size 2n+1 if the depth of the binary tree is n. If the parent element is at the index p, Then the left child will be stored in the index (2*p*)+1, and the right child will be stored in the index (2*p*)+2.

Example: -

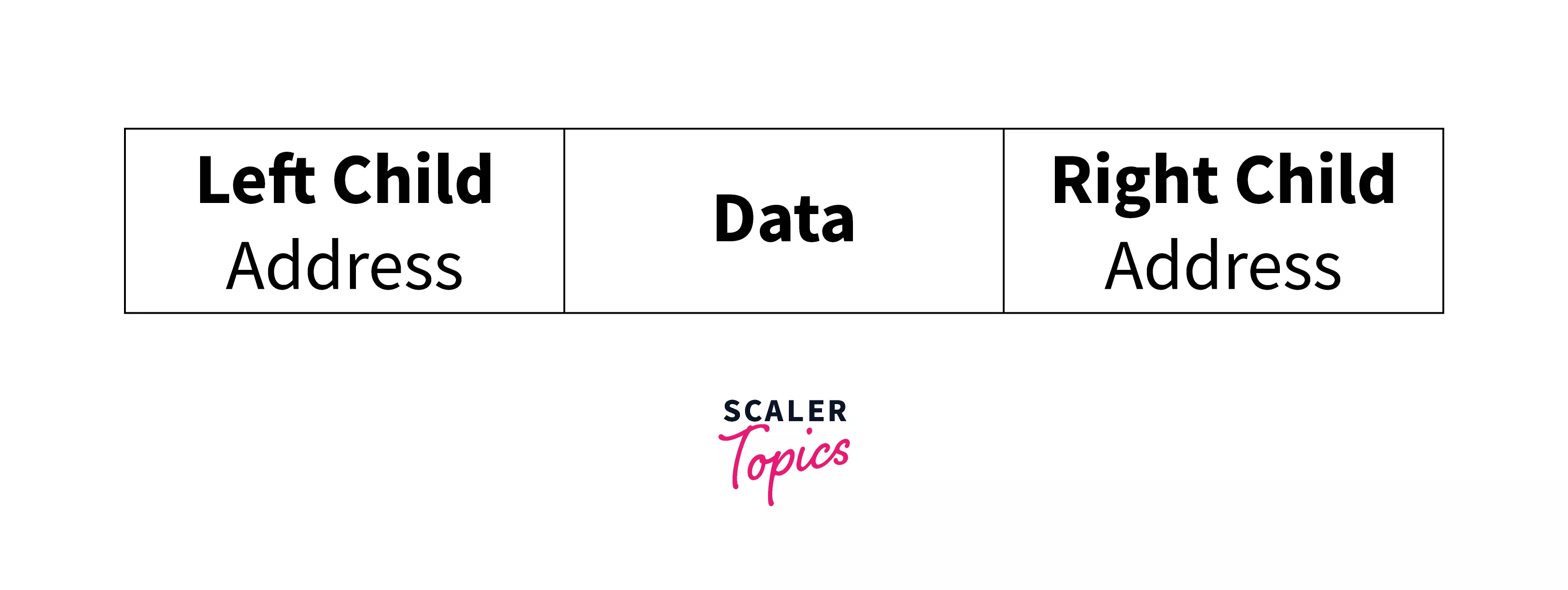


As in the above binary tree, A was the root node, so that it will be stored in the 0th index. The left child of A will be stored in the 2(0)+1 equal to the 1st location. So, B is stored in index 1. And similarly, the right child of A will be stored in the 2(0)+2 index. For every node, the left and right child will be stored accordingly.

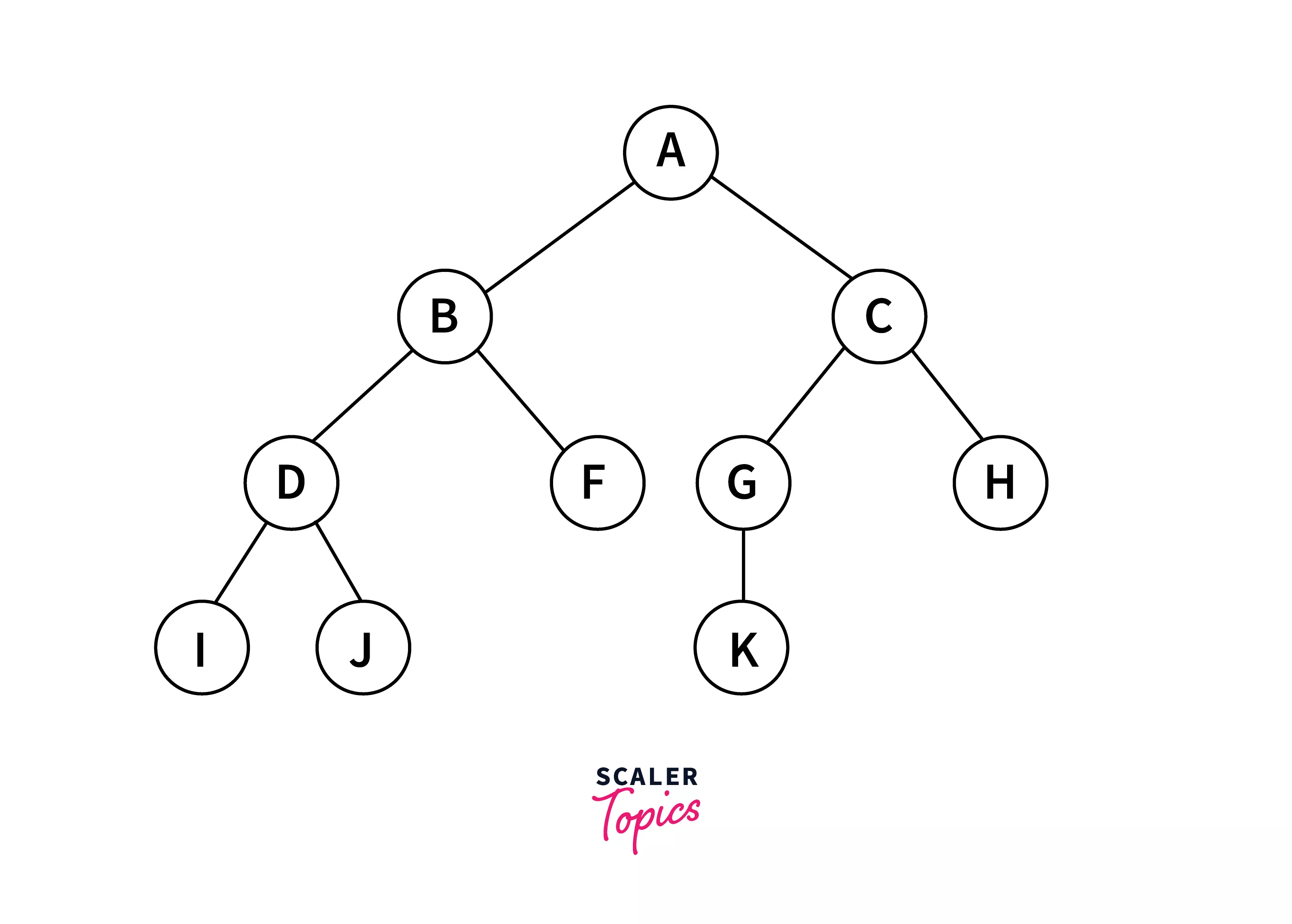


**Linked list representation of Binary trees**

For the linked list representation, we will use the doubly linked list, which has two pointers so that we can point to the left and right children of a binary tree node. NULL is given to the pointer as the address when no child is connected.



Example: -



****The Linked list representation of the above binary tree is :****

