**Merge Sort**

**MergeSort(arr[], l, r)**

If r > l

**1.** Find the middle point to divide the array into two halves:

middle m = (l+r)/2

**2.** Call mergeSort for first half:

Call mergeSort(arr, l, m)

**3.** Call mergeSort for second half:

Call mergeSort(arr, m+1, r)

**4.** Merge the two halves sorted in step 2 and 3:

Call merge(arr, l, m, r)

**Code**

**mergeSort(A,lb,ub)**

**{**

**if(lb < ub)**

**{**

**mid = (lb + ub)/2**

**mergeSort(A,lb,mid)**

**mergeSort(A,mid+1,ub)**

**merge(A,lb,mid,ub)**

**}**

**}**

**merge(A,l,m,u)**

**{**

**i = lb**

**j = mid + 1**

**k = lb**

**takenew - new**

**while( i <=mid && j <=u)**

**{**

**if(a[i] <= a[j])**

**{**

**b[k] = a[i]**

**i++;**

**k++;**

**}**

**else**

**{**

**b[k] = a[j];**

**j++;**

**k++;**

**}**

**}**

**if(i > mid)**

**{**

**while(j < u)**

**{**

**b[k] = a[j]**

**j++;**

**k++;**

**}**

**}**

**if(j > u)**

**{**

**while( i < mid)**

**{**

**b[k] = a[i];**

**k++;**

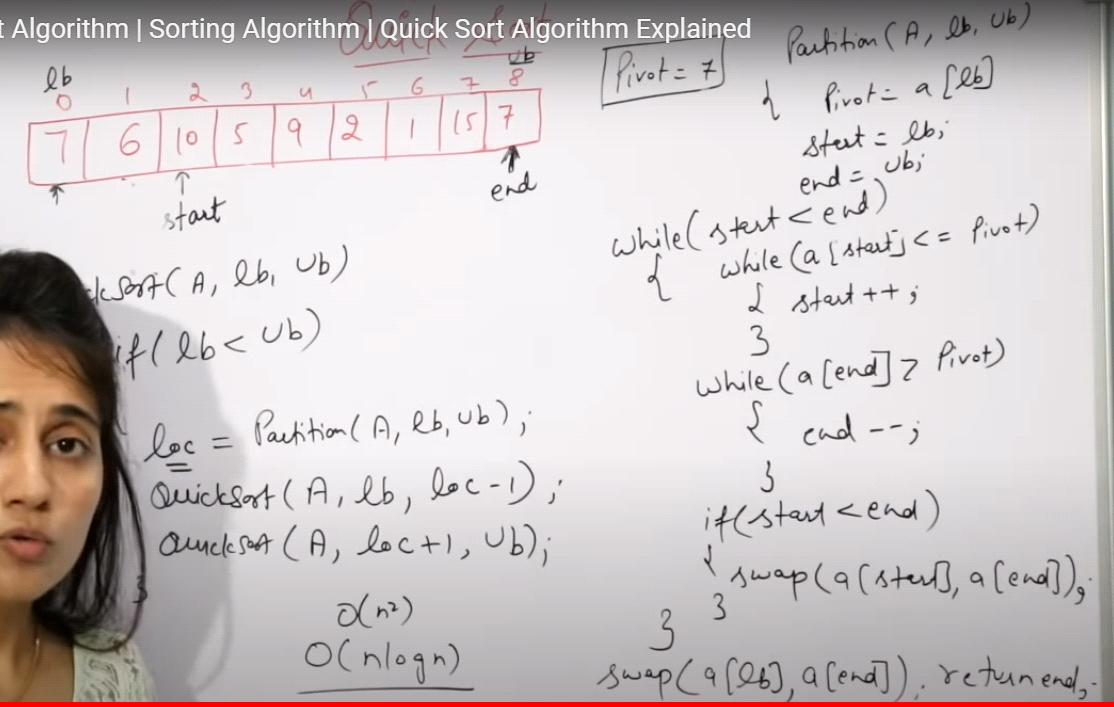
**i++:**

**}**

**}**

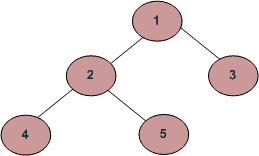
**}**

**Quick Sort**



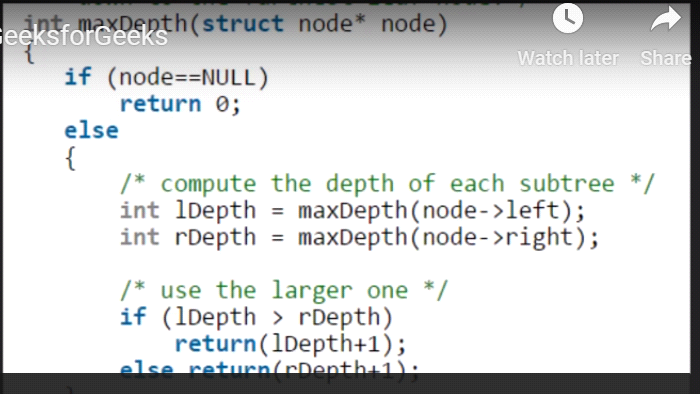
* **Binary Tree Properties**
* ***The maximum number of nodes at level ‘0’ of a binary tree is 2l***.
* ***Maximum number of nodes in a binary tree of height ‘h’ is 2h – 1***.
* **Types of Binary Tree**
* **Full Binary Tree** A Binary Tree is full if every node has 0 or 2 children. Following are examples of a full binary tree. We can also say a full binary tree is a binary tree in which all nodes except leaves have two children.
* **Complete Binary Tree:** A Binary Tree is complete Binary Tree if all levels are completely filled except possibly the last level and the last level has all keys as left as possible
* **Perfect Binary Tree** A Binary tree is Perfect Binary Tree in which all internal nodes have two children and all leaves are at the same level.
* **A degenerate (or pathological) tree** A Tree where every internal node has one child. Such trees are performance-wise same as linked list.

Binary Tree Traversals



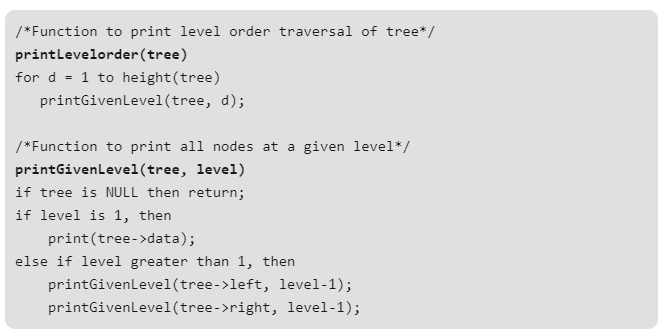
Depth First Traversals:  
(a) Inorder (Left, Root, Right) : 4 2 5 1 3  
(b) Preorder (Root, Left, Right) : 1 2 4 5 3  
(c) Postorder (Left, Right, Root) : 4 5 2 3 1

[**https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/**](https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/)

**Find Height of Tree**

[**https://www.geeksforgeeks.org/write-a-c-program-to-find-the-maximum-depth-or-height-of-a-tree/**](https://www.geeksforgeeks.org/write-a-c-program-to-find-the-maximum-depth-or-height-of-a-tree/)

Level Order Tree Traversal



<https://www.geeksforgeeks.org/level-order-tree-traversal/>

**Construct Binary Tree from Preorder and Inorder traversal**

<https://www.youtube.com/watch?v=PoBGyrIWisE>

**Construct a Binary Search Tree(BST) from given Postorder traversal**

<https://www.youtube.com/watch?v=RttBwUVWfV8>