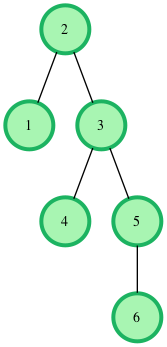
**Algorithm for finding Lowest Common Ancestor of two nodes in a Binary Search Tree**

**What is Lowest Common Ancestor of two nodes?**

Let T be a rooted tree. The lowest common ancestor between two nodes n1 and n2 is defined as the lowest node in T that has both n1 and n2 as descendants (where we allow a node to be a descendant of itself).

The LCA of n1 and n2 in T is the shared ancestor of n1 and n2 that is located farthest from the root. Computation of lowest common ancestors may be useful, for instance, as part of a procedure for determining the distance between pairs of nodes in a tree: the distance from n1 to n2 can be computed as the distance from the root to n1, plus the distance from the root to n2, minus twice the distance from the root to their lowest common ancestor.



If we are given a BST where every node has **parent pointer**, then LCA can be easily determined by traversing up using parent pointer and printing the first intersecting node.

We can solve this problem using BST properties. We can **recursively traverse** the BST from root. The main idea of the solution is, while traversing from top to bottom, the first node n we encounter with value between n1 and n2, i.e., n1 < n < n2 or same as one of the n1 or n2, is LCA of n1 and n2 (assuming that n1 < n2). So just recursively traverse the BST in, if node’s value is greater than both n1 and n2 then our LCA lies in left side of the node, if it’s is smaller than both n1 and n2, then LCA lies on right side. Otherwise root is LCA (assuming that both n1 and n2 are present in BST)

Time complexity of above solution is O(h) where h is height of tree.