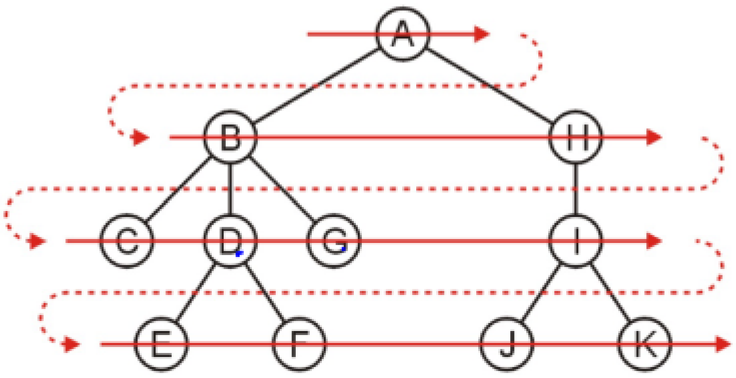
Lab Questions

1. Tree Traversal

As discussed in class, there are various ways to traverse a tree, i.e., visit all nodes of a tree (preorder, postorder, inorder, breadth-first, and depth-first).

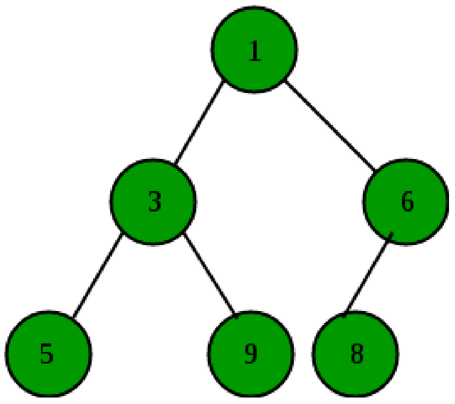


Recall that, in a breadth-first traversal, we visit all the nodes at the same depth before moving to the next depth. While this may be clear conceptually, let's try and code it in C++. You can try and submit your code here. (For those trying to submit on that link, take note of the return data type of the required function).

PS: What is the time complexity of your code?
PPS: Once you are clear with BFS, you should try out DFS(depth-first search)

2. Sum of SubTrees

So why did we learn traversal? Let's try to apply it to a problem.



In this problem, you need to calculate the sum of every subtree in a given binary tree. The sum of a subtree is defined as the sum of values of both its left and right subtree along with its own value.

PS: What is the time complexity?
PPS: Which traversal algorithm did you use? Does one of them fit better here?

Follow up - Can you return the subtree with the largest sum? You should try and submit your code here.

3. Binary Search Trees - Find Kth Smallest Element

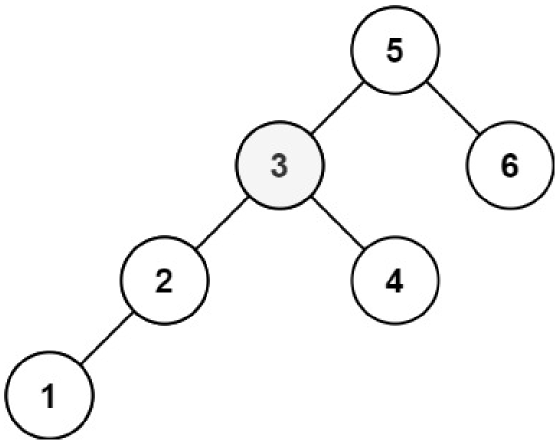
Let's recall what a binary search tree is.

A binary tree such that -

a. Keys stored at nodes in the left subtree of every node v are less than or equal to k (the key at "this" node)
b. Keys stored at nodes in the right subtree of every node v are greater than k

You may be able to appreciate the special properties of BST already. (If not, try doing an inorder traversal. What do you observe?)

Let's solve a different question now. Given a binary search tree and an integer k, you must find the kth smallest element in this BST.



For example, with k=3, the 3rd smallest element in this given BST is 3. (You should verify by hand that this is a BST). Try and submit here.