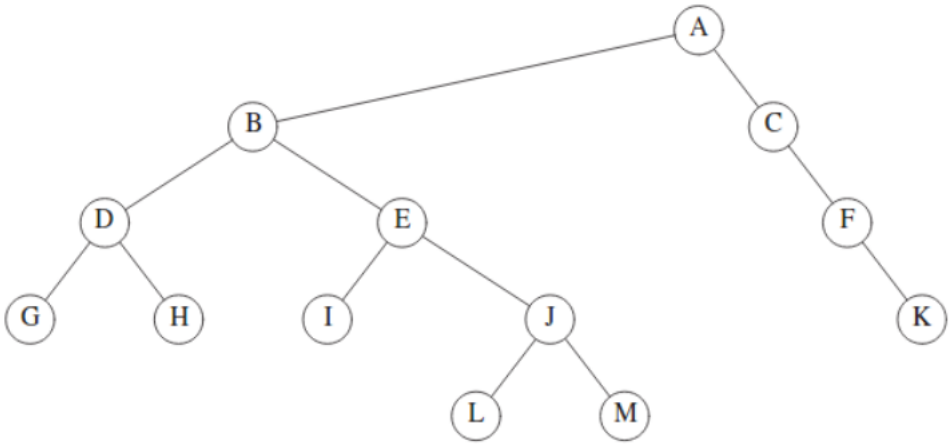
Assignment-4 (Due: November 8, 2018 (THU) – Before the tutorial)

UE101: Algorithms and Programming

Anand Mishra

Instructions:
(1) This assignment contains **5 problems**. Each problem carries 20 points.
(3) We expect **academic honesty** in this course. If someone found copying from other student, both the students will get zero point.
(4) Submission is **due on** November 8, 2018 (THU) – 11:00 AM, i.e. before the tutorial. **No late submission allowed.** Please submit handwritten answers in stapled A-4 sheets, and **do not forget** to write your serial number clearly in the first page of your submission.
(5) **Until specified left subtree means left subtree with respect to root, and similarly right sub tree is defined.**

Problem 1.



For the above tree answer following questions?
(1.1) How many nodes are there in the left subtree?
(1.2) List out the leaf nodes.
(1.3) Which node is a sibling of J?
(1.4) What is the height of this tree?
(1.5) What is level of node I?
(1.6) Is B an ancestor of M?
(1.7) Count the number of nodes having exactly one child?
(1.8) Are C and I siblings?
(1.9) Which node is parent of F?
(1.10) Compute the ratio between number of leaves and total number of nodes.

Problem 2. Create a Binary Search Tree (BST) by inserting names in the following order:
Ankit, Anand, Swati, Adhrit, Satwik, Bharat, Yashwant, Yashaswi, Pankaj, Narendra. Note that (i) order in names are alphabetically defined based on first character. For example: $Anand < Bharat$ since A comes before B in the English alphabet, and (ii) if first character is same in both words then define order based on second character, and if the second character is same then define order based on the third character and so on. For example $An\mathbf{a}nd < An\mathbf{k}it$, since **k** comes after **a** in the English alphabet.
(2.1) Show BST after insertion of each name in the order.
(2.2) In the above BST, list out leaf nodes.
(2.3) What is the height of this tree.

(2.4) How many string comparisons are needed to search word "Bharat" in above BST, and how many string comparisons are needed to say that word "Zhang" does not exist in above BST. (Hint: saying $Ankit > Anand$ requires one string comparison. Although we need three character comparison for this, but we will not be counting character comparisons for this problem.)
(2.5) Delete the root node of above BST, reconstruct it, and show the reconstructed BST.

Problem 3.

A **complete binary tree** is a binary tree in which every level, except possibly the last, is completely filled, i.e., level i contains exactly 2^i nodes except for the last level, and all nodes are as left as possible in the last level.

A **full binary tree** is a binary tree where each node has 0 or 2 children.

A **perfect binary tree** is binary tree where level i contains exactly 2^i nodes.

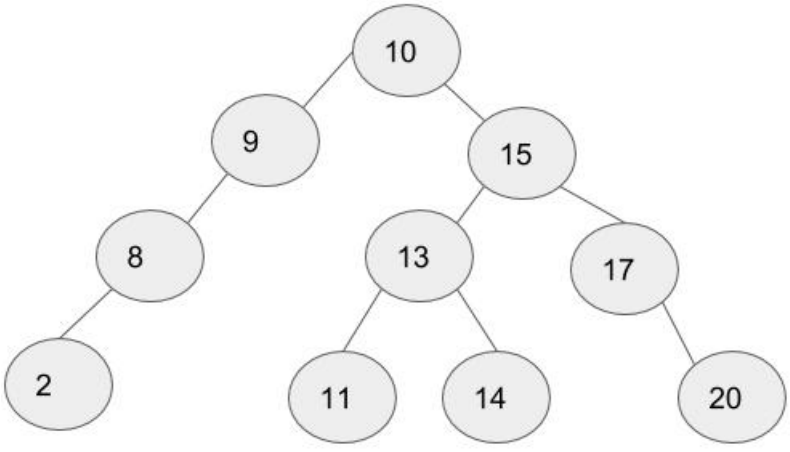
(3.1) If there are n nodes in a complete binary tree then compute its height.

(3.2) If there are 120 nodes in a complete binary tree, where will you find the 120th node: in left subtree or right subtree. Note: (i) Here left and right subtrees are with respect to root node. (ii) Nodes are numbered using level order traversal, i.e., root node is 1st node, left child of root is 2nd node, right child of root is 3rd node and so on.

(3.3) Draw two complete binary trees with 11 and 31 nodes respectively. Nodes can be shown empty.

(3.4) State True or False for the following statements. If you say statement is False, justify with a counterexample/argument. (i) All complete binary trees are perfect binary trees.
(ii) All perfect binary trees are full binary trees.
(iii) There does not exist a full binary tree with finite number of nodes such that its each node has exactly two children.
(iv) In a complete binary tree number of nodes in the left subtree is always greater than or equal to number of nodes in the right subtree.
(v) Number of leaf nodes in a full binary tree is always equal to number of internal nodes plus two.

Problem 4. We have covered following four ways to traverse trees in the course: **(4.1)** preorder **(4.2)** postorder, **(4.3)** inorder, and **(4.4)** level order. Print elements by traversing following tree using above ways.



Problem 5. Answer following questions, and justify with appropriate argument or counterexample.

(5.1) In a complete binary tree there are 20 leaf nodes, in which of the following traversal will the last twenty keys correspond to leaf nodes. Justify your answer.:
(a) Preorder (b) Postorder (c) Inorder (d) Level order

(5.2) State True or False for the following statements. If you say statement is False, justify with a counterexample/argument. :
(a) In a BST the last entry of the inorder traversal is always the largest key.
(b) In a BST the first entry of the postorder traversal is always the smallest key.
(c) In a perfect binary tree ratio between number of all the leaf nodes and number of all the nodes is always greater than 0.5.
(d) The time complexity to insert a sequence of n sorted keys in BST is $O(nlog_2(n))$.
(e) In any BST, searching a key can not take more than $log_2(n)$ time, where n is the number of nodes in the BST.

(5.3) Draw all possible BSTs which store following keys: 1, 2, 3. How many such unique BSTs are there:
(a) 1 (b) 6 (c) 5 (d) None of above

(5.4) In context of BST, state True or False for the following statements. If you say statement is False, justify with a counterexample/argument.
(a) A successor of any node can not have two children.
(b) Minimum key can be found in the right subtree.
(c) If a BST has an empty right subtree then the root contains the maximum key.
(d) If successor of the root node exists, then it has to be in the right subtree.
(e) There does not exist a node in a BST such that the sum of the keys corresponding to it's left child and right child is lesser than the key corresponding to it. (Hint: the keys need not be positive)

(5.5) Draw a BST by inserting following sequence of keys: 10 -10 20 -20 30 -30 40 -40. The sum of the keys corresponding to leaf nodes of tree will be equal to:
(a) 0 (b) -10 (c) -50 (c) None of above

========== END DOCUMENT ==========