Beihang University
School of General Engineering

Beijing, December 8st, 2020
Lin Wei and Yida Ding

Computer Science and Programming
Lab Class 11

Note: Please use the *Node* class introduced in the homework to solve the following tasks.

Task 1 *Binary Search Tree – Search an element (15 minutes)*

Implement a member function which searchs for a given node in a binary search tree. If the node is in the tree, your function should return *True*; if the node is not in the tree, your function should return *False*.

Task 2 *Binary Search Tree – Identify a BST (15 minutes)*

Implement a member function that checks whether a tree is a binary search tree, i.e. a tree such that all left-descendants are smaller than a node's key and all right-descendants are larger than the key of a node.

Task 3 *Binary Search Tree – Obtain the leaves (15 minutes)*

Implement a member function which return all the leaves of the BST, i.e. those nodes without children. For the example visualized in Figure 1, the function should return [1,3,6,8,10].

Task 4 *Binary Search Tree – Obtain the path length (15 minutes)*

Implement a member function which returns the length of each path of the BST. For the example visualized in Figure 1, the function should return [4,4,4,3,3].

Task 5 *Binary Search Tree – Obtain each path (15 minutes)*

Implement a member function which returns each path of the BST in a nested list. For the example visualized in Figure 1, the function should return
[[1,2,4,7],[3,2,4,7],[6,5,4,7],[8,9,7],[10,9,7]].

Once you have finished the above tasks

Task 6 *Binary Search Tree – Construct a BST from list*

Implement a function which constructs a binary search tree from a given list *L*. That is, the binary search tree should contain all the elements in the list and therefore a user could use the binary search tree to accelerate the search process.

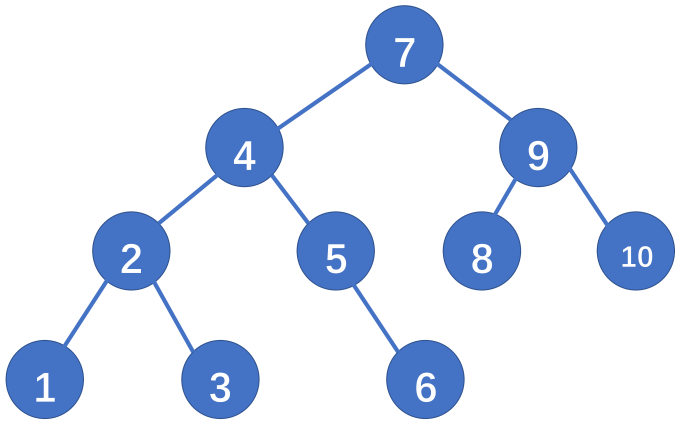

Figure 1: An example for BST