Beihang University
School of General Engineering

Beijing, Nov. 29th, 2019
Sebastian Wandelt, Pengfei Yi, Junyu Zhao

Computer Science and Programming Lab Class 11

Task 1. ***Binary Search Trees*** *(20 minutes)*

Construct a binary search tree (BST) using a similar structure we applied in last lab class. The class should at least contain the following functions:

1. A member function which finds a given element in a binary search tree. If the element is not contained, your function should return *None*.
2. A 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.
3. A function that checks whether a tree is perfectly balanced, i.e., the length of the shortest path in the tree plus one is larger or equal to the length of the longest path in the tree.

Task 2. ***Hashing – linear chaining*** *(20 minutes)*

Implement the hashing class with linear chaining in Python. There should be constructor, insert function, search function and delete function. (1) __init__(self,size)
(2) insert(self,value)
(3) search(self,value)
(4) delete(self,value)

Task 3. ***Hashing – linear probing*** *(25 minutes)*

Implement the hashing class with linear probing in Python. There should be constructor, insert function, search function and delete function. Think carefully about how to properly implement the delete function.
(1) __init__(self,size)
(2) insert(self,value)
(3) search(self,value)
(4) delete(self,value)

Task 4. ***Red-black Trees - Basic*** *(30 minutes)*

Please extend the tree class from the previous task to make it able to model red-black trees (RBTs).

1. For each node in the tree, add a color attribute associated with it. Because in this case we are dealing with RBTs, the color attribute should be either red or black.

2. Implement a function which checks whether or not a given tree instance is a red-black tree. The function should verify five rules (introduced in Page 9 in the lecture slide):
   (a) Every vertex is colored red or black.
   (b) The root vertex is a black vertex.
   (c) A NIL child is a black vertex.
   (d) The child of a red vertex must be a black vertex.
   (e) For all vertices v, all paths from v to its NIL descendants have the same number of black vertices.

Note that this task is only a beginning of the implementation of RBTs. Because of time constraints, we do not implement additional functions for RBTs. It is a nice homework to implement these on your own.