Reichman University

Data Structures & Algorithms – Problem set 3

Coding Assignment

Due: 14.12.2021

Honor code:

1. Do not copy the answers from any source.
2. Don't share solutions.

Submission guidelines:

- Use Python 3.7 or later.
- Submit a zip file named as your id number (e.g 123456789.zip).
- Inside the zip your adjusted ex3.py.
- No other files should be in the zip.

In this exercise we're going to "get our hands dirty" and implement a data structure – AVL tree.

You are given a python file called ex3.py.

Inside the file you have 2 main classes – BST and AVL. In addition, the file includes the classes Node and AVLNode representing a BST node and AVL node, respectively.

You need to implement all of the following empty methods according to the algorithms we learned in class.

An explanation of the desired functionality of each method is supplied here as well as in the python file as a documentation on each method.

BST

- ***Insert(key, value)*** *:* Inserts a new (key,value) pair to the BST. In case key already exists in the BST update the node's value.
  The key is of type int.
  Return value: None.
- ***Delete(key):*** Remove the node associated with key from the BST. If key not in BST don't do anything.
  The key is of type int.
  Return value: OK if deleted successfully or NO_ITEM if key not in the BST.

Reichman University

(OK and NO_ITEM are defined in the beginning of ex3.py, and their values are set to 0 and -1, respectively).

- ***create_BST_from_sorted_arr(arr):*** Creates a balanced BST from a sorted array of keys according to the algorithm from class. The values of each key should be None.
  The input is a sorted array (Type: Python list).
  Return value: an object of type BST representing the balanced BST.
  Note: this is a static method of the class (see explanation below).
- ***inorder_traversal():*** Returns an array (Python list) of keys sorted according to the inorder traversal of the BST (self).
- ***postorder_traversal():*** Returns an array (Python list) of keys sorted according to the postorder traversal of the BST (self).
- ***preorder_traversal():*** Returns an array (Python list) of keys sorted according to the preorder traversal of the BST (self).

AVL:

- ***Insert(key, value)*** *:* Inserts a new (key,value) pair to the BST. In case key already exists in the BST update the node's value.
  Make sure to keep the tree balanced!
  The key is of type int.
  Return value: None.
- ***Delete(key):*** Remove the node associated with key from the BST. If key not in BST don't do anything.
  Make sure to keep the tree balanced!
  The key is of type int.
  Return value: OK if deleted successfully or NO_ITEM if key not in the BST.

As an inspiration we provided you with the implementation of ***find(key).*** Also, we supplied a visualization of the tress which can be accessed by calling ***print*** with a BST/AVL object.

Here is an example:

```
>>> tree = BST()
>>> tree.insert(10,"value for 10")
>>> tree.insert(20,"Hi")
```

```
>>> print(tree) #calls __repr__ of class BST
-------------------------------------
 10
! 20
>>> tree.insert(30,"Hello")
>>> print(tree)
-------------------------------------
   10
 !   20
! ! ! 30
>>> tree = AVL()
>>> tree.insert(10, "value for 10")
>>> tree.insert(20, "Hi")
>>> print(tree)
-------------------------------------
 10
! 20
>>> tree.insert(30, "Hello")
>>> print(tree)
-------------------------------------
 20
10 30
```

You may add additional methods but **you may not change the API**, i.e. do not change class name, existing fields, the signatures of the methods and the methods we supplied for you.

In case you change the API you would fail all of the automatic tests and the assignment.

You are encouraged to test your implementation and check if it satisfies the desired functionality. You may share tests among you ☺

Notice that we've implemented the AVL class as extension of the BST class so the insert and delete methods of the AVL override the BST methods. In case you need a reminder of inheritance in python use this nice tutorial:
https://www.w3schools.com/python/python_inheritance.asp

Reichman University

Note that some of the methods are implemented as static methods of a class, which means that they do not require a class instance creation. Therefore, they do not accept a parameter "self" as other representing the instance itself (the object).

These methods are called by the class. For examples:

arr = [1,2,3,4,5]

BST.create_BST_from_sorted_arr(arr)

More information can be found here: https://www.geeksforgeeks.org/class-method-vs-static-method-python/

Good Luck!