Contents

AVL Trees

Description

Purpose

This project is designed to help you develop a better understanding of binary search trees and AVL trees. It requires you to manipulate trees in various ways, and to understand the different cases requiring re-balancing a tree.

Challenge

In short Our goal is improve the second implementation of AVL tree and to understand it better. You will be asked to write additional methods, develop new examples, and comment your code.

In more details We want to implement a more pedagogical version of AVL trees, where operations such as re-balancing are easier to observe step-by-step.

- Start by downloading the existing implementation,
- Add your name in a delimited comment at the top of `Program.cs`,
- Observe how there is currently some illustration as to how `RotateleftChild` and `DoubleleftChild` operate, using the `public` methods `Rotateleft` and `Doubleleft`, when trees are unbalanced after insertion.

Your goal is to edit and expand the solution as follows:

- Inside `Program.cs`, illustrate similarly with `Rotateright` and `Doubleright` from `IBtree` how `RotaterightChild` and `DoublerightChild` operate. Create a tree by inserting values, note (in the comments) why it becomes un-balanced, and how it is possible to re-balance it using one of the aforementioned method. Create another example to illustrate the other method.

- Inside `Program.cs`, create a `BSTree` tree object that is "overall" balanced, but that has sub-tree(s) with a balance greater than or equal to 2 or less than or equal to -2.
- Create an "Improved" AVL tree class called `IAVLTree` that inherits from `AVLTree`, and contains a `Depth` method that computes the depth of a value: given a value of type `T`, the method should return the depth of the node containing this value, or `-1` if this value is not in the tree. Remember that

  > The depth of a node is the number of edges from the node to the tree's root node.

- Inside `Program.cs`, write a snippet of code that
  - Create an `IAVLTree` containing `int`s,
  - Insert 10 random values between 1 and 49 inside of it,
  - Ask the user to enter a number,
  - Displays the depth of the number in the tree.

Pay attention to details:

- Your program should catch possible exceptions.
- **Do not modify any file other than `Program.cs`, do not create any file other than `IAVLTree.cs`**. If you *really* need to edit some other file, *please indicate it very clearly at the beginning of `Program.cs`*.
- **Do not load any additional libraries**, in particular, **do not use C# native lists or LINQ**.

Bonuses

Bonus points will be given if:

- (easy) Illustrate how
  - `RotaterightChild`,
  - `RotateleftChild`,
  - `DoublerightChild` or
  - `DoubleleftChild`

  operate after a tree becomes unbalanced after a **deletion** (the examples above had trees unbalanced following an **insertion**).
- (medium) Override the `Insert` from `AVLTree` in your `IAVLTree` class so that it uses `SubtreeBalance` (like `Delete` do). Write good test cases to make sure your method behaves as expected.

Submission

Please, follow our guideline on project submission. In particular, make sure you write your name and the date in a delimited comment at the beginning of your file.