American University of Armenia, CSE
CS 121 Data Structures A, B, C
Fall 2019

Homework Assignment 5

Due Date: Thursday, November 7 by 23:55 electronically on moodle

Any submissions containing cheating and/or plagiarism, by university policy, will be reported and will result in the grade F for the entire course.

Please solve the programming tasks either in Java or C++, following good coding practices (details posted in moodle).

You should submit full tested programs for all questions.

1. **(23 points)** Write a class `ArrayBinaryTree` that extends the `AbstractBinaryTree` class using an array of fixed capacity as the underlying data structure. Note that the `AbstractBinaryTree` class in turn extends the `AbstractTree` class and implements the `BinaryTree` interface. Your class should support all of the following functionality:

   (a) two constructors that create an empty tree: a no-arg constructor that sets the default capacity of the array, and another constructor that receives the array capacity as an argument;

   (b) two methods for determining the **height** and **depth** of a given position and a method for determining the **height** of the tree;

   (c) functionality for traversing the elements of the tree, i.e. an `iterator()` method;

   (d) functionality for traversing the positions of the tree in preorder, postorder, inorder and breadth-first order traversals, i.e. `preorder()`, `postorder()`, `inorder()`, `breadthfirst()` methods, all of which return an iterable collection of the positions of the tree;

   (e) functionality for traversing the positions of the tree, i.e. a `positions()` method implementing preorder traversal;

   (f) methods `addRoot(`e`)`, `addLeft(`p, e`)`, `addRight(`p, e`)`, `remove(`p`)` similar to the corresponding methods for the `LinkedBinaryTree` class.

   Think carefully where you should add each of these methods; you may need to modify any of the `AbstractTree`, `AbstractBinaryTree` and `ArrayBinaryTree` classes.

2. **(20 points)** Write a generic class that implements the stack ADT using only a priority queue and one additional integer instance variable.

3. **(15 points)** Write a `heapsort` method that implements **in-place** heapsort for a given array of entries with integer keys.

4. **(12 points)** Write a method that, given an array of entries and a comparator for the key-type, checks if the array represents a heap.

5. **(15 points)** Write a method that takes an array *arr* of n entries and a comparator for the key-type and produces an array of k largest entries in *arr*, based on their keys. The execution time of your algorithm should be $O(n + k \log n)$ and it should use a max-heap.

6. **(15 points)** Extend the `LinkedBinaryTree` class with an **iterative** `inorderAfter` method that, given a position p in the tree, returns the position q that follows p in an inorder traversal of the tree.