Programming assignment 2

Getting started

All your submissions should be implemented in Java. If you do not have a Java development environment, we recommend that you either use the Visual Studio Code Java extensions or Jetbrains IntelliJ. If you use IntelliJ, do not forget to register as a student, so you get access to the full versions. If you run Linux, macOS, or WSL, and want more control over your JDK, we recommend SDKMAN!

Note that you must solve all of the programming problems in Java. Kotlin or Scala are also ok, but you might not get as much support. You are allowed to use Python (or any other suitable tool) to analyse the results, plot graphs, and so on.

Problems

Problem 1

Implement a method that reverses a single-linked list in $O(n)$ in place. You cannot use any additional memory other than the required pointers to traverse the list. You must implement a single-linked list to demonstrate that your method works.

Problem 2

Implement a generic deque (double-ended queue). In a deque you can add and remove from either end so that it can work either as a stack or a queue. Your implementation should at least support the following operations: size, isEmpty, addFirst, addLast, removeFirst, and removeLast. You should also implement an iterator for your deque.

Include reasonable error handling, e.g., throw exceptions when trying to remove something from an empty list.

Problem 3

Implement a randomized queue where each dequeue operation returns a random element from the queue. Your implementation should support the following operations: size, isEmpty, enqueue, dequeue. Note that dequeue should pick one of the inserted elements randomly, remove it and return it. You should also implement an iterator for your randomized queue. Note that the iterator should iterate over the elements in random order.

Problem 4

Write a program that, given a starting directory/folder, finds all the files and folders of that directory recursively and represents them using a tree. Start by implementing a left-most child, right sibling tree ADT. You can decide the API, but it should at least support finding a node by path/name and adding a child to an existing node. Then use the methods in `java.io.File` to find/list the files contained in the provided folder.

Problem 5

Implement a binary search tree that supports at least the following operations: height, size, add, remove, and contains. You should also implement in-, pre-, and post-order iterators. Once you are convinced that your operations work as expected, add an operation to remove the kth largest value in the tree, where k is a parameter.

For example, if you have a tree that contains the values 1 to 10 and we want to remove the third largest value (k=3), then eight would be removed.

Your implementation should throw an exception if there is no kth largest value.

Problem 6 (optional, for higher grade)

Huffman coding is commonly used for lossless data compression. Write a program that reads a text file, computes the frequency for each character, and creates a Huffman tree. You do not need to produce the compressed output, but your tree should provide a method to get the Huffman code for a character.

Problem 7 (optional, for higher grade)

Implement one of the "balanced" trees discussed (AVL, splay, or red-black) and compare it to a binary search tree without effort to balance it. You should devise a reasonable way to create realistic average case trees (e.g., combinations of inserts and deletes) and then compare in terms of operation cost/time, height, etc.

In addition to the code and README-file, provide a report where you describe and reflect upon your experiment and findings.

Submission guidelines

Submit your solutions as a single zip-file via Moodle no later than 17:00 on October 7, 2022 (cutoff 24:00 October 9). This is an individual assignment. Your submission should contain well-structured and organized Java code for the problems with a README.txt (or .md) file that describes how to compile and run the Java programs.