CSCI 2235
Programming Project 03 - RBTree v. Skiplist

Assigned: November 04, 2019
Due: December 06, 2019 @ 23:00h

Purpose

- To explore the implementation of RedBlack Trees and SkipLists
- To compare the results of RedBlack Trees and SkipLists.

Problem Statement

As computer scientists we often have multiple competing solution approaches. To evaluate these approaches we need to compare these approaches across different contexts to evaluate which would be best for our given situation.

Assignment

0. Initialize a new Gradle project, using the gradle command at the command line, with the following characteristics:
    - Create a directory named `pp03` in your normal project directory
    - Open a command prompt/terminal and change directories to the newly created `pa04`
    - From within the `pa03` directory run the following command: `gradle init` and provide the following information
        - Type of project: 4 // Java application
        - Build script DSL: 1 // groovy
        - Test framework: 1 // junit
        - Project name: [enter] // default should be `pp03`
        - Source package: edu.isu.cs2235
    - This will create several files in your directory. Open the `build.gradle file` note the last line of the file says: mainClassName = `'edu.isu.cs2235.App'`
    - You will need to change this to the fully qualified name of the main class for your project, in previous projects this was Driver. You can keep app but you will need to modify it to complete the assignment. Also note that there is a test class called `AppTest` which if you delete App, you will need to delete AppTest as well.
1. Create a BinarySearchTree interface and extend your LinkedBinaryTree to create a LinkedBinarySearchTree.

- Implement a test suite to test and validate your implementation of the BinarySearchTree.

2. Further Extend the LinkedBinarySearchTree by creating a new RedBlackTree and implement the RedBlackTree algorithms.
   - Implement a test suite to test and validate your implementation of the RedBlackTree.
3. Implement a SkipList on top of your DoublyLinked List from PA02
   - Implement a test suite to test and validate your implementation of the SkipList
4. I have provided you with a two classes. The second class is the data type `Person` with a first and last name, height (cm), and The second is a Singleton utility class called Generator, which works as follows:
   - To call any method on the Generator, you need to first create an instance of the Generator class. Since this is a singleton, there can only be one instance at any time. But, you can acquire and instance using the following method call: `Generator.instance()` which returns the single known instance of `Generator`
     - To generate a random integer within some range you may use the `generateInt(min, max )` method which returns a random integer (uniformly distributed) of at least min and less than max.
     - To generate a random double within some range you may use the `generateDouble(min , max)` method which returns a random integer (uniformly distributed) of at least min and less than max.
     - To generate a random string with ranged length you may use the `generateString(min, max)` method which returns a random string of length at least min and less than max.
     - To generate a random person you may use the `generatePerson()` method.
5. Again we are going to evaluate the effectiveness of search, this time we are concerned with the effectiveness of the underlying data structure in providing for the capability of employing the binary search strategy within a linked structure. Thus, the comparison will compare for search times across a RedBlackTree and SkipList composed of the following sets of data (each a separate trial):
   - 250,000 random integers with a prespecified range (your choice).
   - 250,000 random doubles with a prespecfied range (your choice).
   - 250,000 random strings
   - 250,000 random people

For each of these data sets you are to construct a RedBlackTree and SkipList (with the exact same data inserted in the same order). Then, you will perform, per data set per data structure, 25,000, random searches. These searches should be conducted in the same order, with the same search data, per data structure (for comparison). You will collected the average time for each data structure, for each type of data to be used in comparison.

5. Once your data has been collected you will need to writeup the comparison into a report using the IEEE format. Save this file as a PDF into a folder named “writeup” in the root directory of your project.

Submission

Submit a zip file containing your project directory and your writeup in PDF format to moodle by the deadline.

Grading

7 Points Implementation of the RedBlackTree (including the BinarySearchTree) 7 Points Effective Testing of the RedBlackTree and BinarySearchTree 5 Points Implementation of the SkipList 6 Points Effective Testing of the SkipList 5 Points Implementation of the Data Collection main class 20 Points Well written writeup of the results and comparison.