Assignment 3
Priority Queues

Release Time	Due Date
February 22, 2017	**March 16, 2017**

Objectives

- Experience different ways to implement heaps
- Practice developing high-performance solutions.
- Practice tree-traversals.
- Analyze the advantages and disadvantages of explicit and implicit representations of trees

Problem Specification

In class we discussed various ways to implement priority queues. Two prominent ones are using heaps in their explicit or implicit representations. We will practice implementing these and compare their advantages and disadvantages. In particular, implement (i) a min-heap using a size-balanced binary tree SBT in its explicit representation, and (b) a min-heap using an array-based complete binary tree CBT.

Write a JAVA application to solve the following problem,

1) Given are a sequence of heap-operations namely, "findmin", "deletemin", and "insert <key, value>" where key is an integer and value is also an integer (in practice value would be a reference / pointer to the record but for this homework assignment this simplification will suffice)

2) Execute each operation on SBT and CBT, i..e,, if the operation is deletemin, delete the record with the smallest key form SBT and CBT; if the operation is "insert <key, value>", then insert the record with the integer key along with its value in the min-heaps SBT and CBT; obviously if the operation is findmin, then simply return the appropriate record from each of CBT and SBT.

3) In order to empirically compare the computational requirements of the two implementations, we will randomly generate data and the sequence of operations (in practice these come from another application). Generation of data

    a. Operation – randomly generate an integer x in the range [1, 10000], if x **ε** [1, 2000], then the operation is insert, if x **ε** [2001, 4000], then the operation is deletemin, and if x **ε** [4001, 10000] then the operation is a findmin. Note that we can always vary the lower and upper bounds of these ranges to vary the probability of each operation.

    b. Key – a randomly generated integer in the range [1, 1000]

   c. Value – a randomly generated integer in the range [5000, 20000]

   d. Repeat a-c for the desired data sizes (i.e., number of operations or the length of the sequence of operations) in your experiment

4) Measure the average execution time for each of the operations in each experiment.

5) Measure the total time for the sequence of operations generated in an experiment.

6) Run at least 100 different experiments with sequence lengths 100, 200, 300, ….

7) Plot the measured times in items 4) and 5) above (so the x-axis would be the sequence lengths and y-axis would be measured times). Compare these. Which one is better CBT or SBT? For what sizes? Do the observed times agree with theoretical analysis?

8) Just so we can easily make sure your implementations are working correctly, print the before- and after-states of CBT and SBT when the heap-size reaches 30 for the first time. That is print CBT and SBT before and after each of the operations, insert, deletemin, and findmin, CBT is simply an array so just print the <key, value> pairs separated by commas as they are stored in your CBT. SBT uses explicit representation of binary trees with an additional "size" field at every node, so print SBT twice, once using a preorder traversal with node-data separated by commas (as in [<key1, value1>, size1], [<key2, value2>, size2], …) and then using an inorder traversal (*pause: why do we want SBT printed using different traversals?*). Thus, in total, there will be only six prints of CBT (once before and once after each heap-operation) and twelve prints of SBT (twice before and twice after each heap-operation) in this assignment. Exclude "printing" time from your measurements.

Design Requirements

Code Documentation

For this assignment, you must include documentation for your code as generated by JavaDoc. You should have JavaDoc comments for every class, constructor, and method. By default, JavaDoc should output html documentation to a subfolder within your project (/dist/javadoc). Make sure this folder is included when you zip your files for submission. You do not need to submit a hard copy of this documentation.

Hint: http://stackoverflow.com/questions/4468669/how-to-generate-javadoc-html-in-eclipse

Coding Standards

You must adhere to all conventions in the CS 3310 Java coding standard. This includes the use of white spaces for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods.

Testing

Make sure you test your application with several different values, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
  - Source code files
  - any input or output files
- Javadocs
- 1-2 page analysis report from item 7 of the problem specification
- Readme file – instructions on how to run your program