Assignment 4
Priority Queues and Medians

Release Date	Due Date
February 27, 2020	**March 19, 2020**

Objectives

- Understanding the mechanisms of priority queues in depth
- Creating and manipulating heaps based on linked lists and arrays
- Using priority queues to solve practical problems
- Get experience with implementing tree-based data structures and their traversals

Problem Specification

Suppose we are interested in dynamically maintaining a set S of integers, which is initially empty, while supporting the following three main operations:

1. *add(v)*: Adds value v to set S.

2. *getMedian():* Returns the current median value of the set. For a set with even cardinality, we define the median as the average of the two most central values. For a set with odd cardinality, median is the middle element when considering data in a sorted manner. (In other words, a median is the element in the data set which separates the higher half of the data sample from the lower half; OR half the elements are smaller and half the elements are larger than the median).

3. *deleteMedian():* deletes upper median of the set for **even cardinality**, otherwise deletes median if it's an **odd cardinality**.

We will store each element of the set in one of the two priority queues:

- A min-oriented priority queue, Q+, of all elements greater than or equal to the current median value.

- A max-oriented priority queue, Q−, of all elements less than the current median value.

Steps to follow:

a. Explain how to perform the operation getMedian() in O(1) time given such a representation.

b. Explain how to perform the operation S.add(v) in O(log n) time, where n is the current cardinality of the set, while maintaining such a representation.

c. Explain how to perform the operation S.deleteMedian() in O(log n) time, where n is the current cardinality of the set, while maintaining such a representation.

d. Design and analyze efficient algorithms to support these three operations when S is stored using an array instead of two priority-queues. Compare the theoretical time complexities of the two solutions. You do not need to implement this solution to find median of a set.

e. Implement Q+ using size-balanced **Linked-List based** heaps.

f. Implement Q- using **array-based** heaps.

g. Test your implementation of the above "median-finding" data structure using interactive prompts from console, as shown in the Sample Inputs and Outputs below.

h. A large sequence of add()/getMedian()/deleteMedian() is given in an input file **hw4input.txt**, measure the average time for add(), getMedian() and deleteMedian() operations from this sequence. Do these timings concur with the theoretical time complexities? If not, can you explain why there is discrepancy? *For this and the remaining, turn-off the interactive prompts from the console and execute the operations listed in the hw4input.txt file.*

i. Use the above data structure to implement *sortUsingMedians()* which returns a sorted array of length n, where n is the current cardinality of the set S as maintained above. Measure the time for sortUsingMedians() – does it concur with O(n log n)?

j. (**10 points extra credit**): Using the same data values as in **hw4input.txt** file, use merge-sort to return a sorted array of length n. Compare this timing with sortUsingMedians()? What do you observe? Compare these two timings with merge-sort's theoretical time complexity O(n log n). What do you observe?

Sample Inputs and Outputs

Console output 1:

Assuming that the set already has some values inserted by the user, and the set currently contains these values. S = {5, 15, 1}. Q- contains {1}, and Q+ contains {5, 15}

```
>A – add?
>C – check median?
>D – delete median?
>E – exit?
```

```
> A
> ok, specify the value v to be inserted into S
> 3
> done, value 3 is inserted in Q-,
Inorder traversal of Q- is: 1, 3
Inorder traversal of Q+ is: 5, 15
```

Console output 2:

Assuming that it’s the same set after adding 3, i.e., S = {5, 15, 1, 3}, with Q- = {3, 1} and Q+ = {5, 15}

```
>A – add?
>C – check median?
>D – delete median?
>E – exit?
>C
>the median is 4        // because its (3+5)/2
Inorder traversal of Q- is: 1, 3
Inorder traversal of Q+ is: 5, 15

>A – add?
>C – check median?
>D – delete median?
>E – exit?
>D
>lower median 3 is deleted from Q-      //even cardinality deletes lower median
Inorder traversal of Q- is: 1
Inorder traversal of Q+ is: 5, 15

>A – add?
>C – check median?
>D – delete median?
>E – exit?
>D
>median 5 is deleted from Q+.           //odd cardinality deletes median
Inorder traversal of Q- is: 1
Inorder traversal of Q+ is: 15

>A – add?
>C – check median?
>D – delete median?
>E – exit?
>E
```

>exit program.

Hw4input.txt file sample

a 10, a 12, a 3, a 5, a 7, g, d, g, a 2, a 20, d, g, d, d, d, g, g, d, d, d

Here “a 10” represents add(10), d represents deleteMedian() and g represents getMedian() operation.

Design Requirements

Code Documentation

For this assignment, you must include documentation for your code. This include how to compile and run your program. Also, you **must** give outputs of your program using console prompts for the input sequence: a 10, a 12, a 5, a 7, g, d, g, a 2, a 20, d, g, d, d, d, g, g, d, d, d. If you don’t provide outputs for adding, deleting, or returning the median of the data structures along with inorder traversals of Q- and Q+ as shown in sample inputs and outputs, it means your program does not execute properly.

You must also include documentation for your code as generated by JavaDoc (and similar tools if using some other language). 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 Conventions and Programming 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 files, classes, variables, method parameters and methods. Read the material linked from our class web-pages (in case you can’t recall programming styles and conventions from your CS1 and CS2 courses).

Testing

Make sure you test your application with several different values capturing different cases, to make sure it works. That is test your program not just on the inputs we specified but other representative test data that covers your program’s logic as well.

Assignment Submission

- Generate a .zip file that contains all your files, including:
    - Signed Plagiarism Declaration
    - Source code files
    - Include any input or output files
    - Documentation of your code – e.g. using Javadoc if using Java
    - A brief report (in a pdf file) on your observations of comparing theoretical vs empirically observed time complexities. Note this report will include (a) a brief description of problem statement(s), (b) algorithms descriptions (if these are standard, commonly known algorithms, then just mention their names along with customization to your specific solution(s), otherwise give the pseudo-code of your algorithms, (c) theoretically derived complexities of the algorithms used in your code, (d) table(s) of the observed time complexities, and (e) plots comparing theoretical vs. empirical along with your observations (e.g. do theoretical agree with your implementation, why? Why not?).
    - Beginning of the first file (i.e., readme file) should clearly identify you, the class, submission date, and the main goals of the homework / programming assignment. Also **add credit to others if you had to resort to looking up the solution elsewhere**. Finally add one of the sentences: " I give permission to the instructor to share my solution(s) with the class." or " I do NOT give permission to the instructor to share my solution(s) with the class."
- Don't forget to follow the naming convention specified for submitting assignments