Department Computer Science and Software Engineering
Concordia University

COMP 352: Data Structures and Algorithms
Fall 2024 – Programming Assignment 3

Due date and time: Saturday Nov 30th, 2024 by 11:59 PM
The due date is sharp and strict. NO EXTENSION WILL BE ALLOWED BEYOND THIS TIME! DEMOS START ON SUNDAY, DECEMBER 1.

The Smarter PQ!

In class, we discussed the priority queue (PQ) ADT, which is implemented using min-heap. In a min-heap, the element of the heap with the smallest key is the root of the binary tree. On the other hand, a max-heap has as root the element with the biggest key, and the relationship between the keys of a node and its parent is reversed of that of a min-heap. We also discussed an array-based implementation of heaps (which are simply BSTs).

In this assignment, your task is to implement a new PQ, which we refer to as the **Smarter PQ (SPQ)** ADT using both min- and max-heap. SPQ is smarter than the standard PQ as being *adaptable* and *flexible***.**

The specifications of SPQ are as follows:

- The heap(s) must be implemented from scratch using an array that is dynamically extendable. You are not allowed to use any list (including Arraylist and linked lists), tree, vector, or heap implementation already available in Java or through a 3rd party online.
- You must not duplicate code for implementing min- and max-heaps (i.e. the same code must be used for constructing min or max heaps. Hence think of a *flexible* way to parameterize your code for either min- or max heap).

The priority queue is also *adaptable* meaning that any key or value of any entry of the priority queue can be modified, where the entry object is passed as a parameter. An entry can also be removed from anywhere in the priority queue.

The following are the methods of the SPQ ADT:

- **toggle():** transforms a min- to a max-priority queue or vice versa.
- **removeTop():** removes and returns the entry object (a key, value pair) with the smallest or biggest key depending on the current state of the priority queue (either Min or Max).
- **insert(k,v):** insert (k,v) which is a key(k), value(v) pair to the priority queue, and returns the corresponding entry object in the priority queue.

- **top():** returns the top entry (with the minimum or the maximum key depending on whether it is a Min- or Max-priority queue, without removing the entry.
- **remove(e):** Removes entry object e from the priority queue and returns the entry
- **replaceKey(e, k):** replace entry e's key to k and return the old key.
- **replaceValue(e, v):** replace entry e's value to v and return the old value.
- **state():** returns the current state (Min or Max) of the priority queue.
- **isEmpty():** returns true if the priority queue is empty.
- **size():** returns the current number of entries in the priority queue

You have to submit the following deliverables:

a) Pseudocode of your implementation of the SPQ ADT using a parameterized heap that is implemented using extendable array. Note that Java code will not be considered as pseudocode. Your pseudocode must be on a higher and more abstract level.
b) Tight Big-O time complexities of:

toggle (), remove (e), replaceKey (e, k), and replaceValue (e, v) operations,

together with your explanations. **These methods should be as efficient as possible**.

c) Well documented Java source code with at least 20 different but representative examples demonstrating the functionality of your implemented ADT. These examples should demonstrate all cases of your ADT functionality (e.g., all operations of your ADT, several cases requiring automatic array extension, sufficient number of down and up-heap operations, changing keys, and removing entries from anywhere of heap).

Submission

- **A group of 2 (maximum) is allowed. No additional marks are given for working alone.**
- **If working alone, you need to zip (see below) and submit your zipped file under the submission folder: Programming Assignment 3.**
- **If working in a group, only one submission is to be made by either of the two members (do not submit two copies). IF YOU SUBMIT TWICE, A ZERO MARK WILL BE GIVEN. You need to zip (see below) and submit your zipped file, under the submission folder: Programming Assignment 3.**

Submission format: All assignment-related submissions that have more than one document must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must also contain your name(s) and student ID(s). Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. If working in a group, the file name must include both IDs and last names.

IMPORTANT: A demo for about 5 to 10 minutes will take place with the marker. THERE WILL BE A DEADLINE TO BOOK YOUR DEMO SLOT, AND YOU MUST BOOK YOUR DEMO PRIOR TO THAT TIME. You (or **both** members if working in a group) **must** attend the demo and be able to explain their program to the marker. Different marks may be assigned to teammates based on this demo. The schedule of the demos will be determined and

announced by the markers, and students must reserve a time slot for the demo (only one time-slot per group; or zero mark is given).

Now, please read very carefully:

- **If you fail to demo, a zero mark is assigned regardless of your submission.**
- **If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.**
- **Failing to demo at the second appointment will result in zero marks and no more chances will be given under any conditions.**