CS2223: Algorithms | Assignment 2

General Instructions

• Code vs. Pseudocode: Each question will state whether the deliverable is pseudocode or an executable program and code that the TA will run to give you a grade.

• You should make clear in your report:
– How to compile your program
– How to execute it, along with any arguments

• Submissions: Submit your work as a .Zip file through Canvas. All programs plus your written questions (which should be PDF files to ensure we can see them) should be zipped into a single archive when submitted. Please organize folders by question, e.g. Q1, Q2, etc.

Q1: Sorting with Priority Queues, Min edition (40 pts)

In this question, you are to implement a version of HeapSort based on your book's implementation, with new features.

Tasks:

1) Implement the MaxPQ on p. 318 of Sedgewick.
2) Based on MaxPQ, make a new class, MinPQ
    a) MinPQ should implement the MaxPQ API (p. 309)
    b) **Except!** delMax should become delMin. That is, delMin will remove the minimum, while maintaining heap order.
    c) ^^ Your book discusses this at length, follow the clues!
3) Create your own "Data" class
    a) Create a class of your choosing modeled after the Transaction class described in your book.
    b) The essential property we're looking for is that you implement Comparable and can sort this more complex object according to some property it holds.
        i) Examples: a *Transaction* class that sorts on transaction amount (e.g. exercise 2.1.21). A *Document* class that sorts on creation date. A *Patient* class that sorts on some severity score.
4) In your main program, call functions from your MinPQ class to demonstrate the following functionality:
    a) Show that your implementation can be used to sort an array of randomly generated Integers (class code provides ways of making these)
    b) Manually or automatically create an array of objects based on the class you defined in Step 3, and demonstrate that you can sort these, as well

A note on sorting: In class code, we showed you how to use Priority Queues with Stacks for sorting. Basically, you insert all items, retrieve them repeatedly and put them on a Stack, then print the Stack.

Output:
The output of this should include Java code TAs can inspect and run.

Q2: HeapSort Class and Evaluation (40 pts)

In this question, your task is to take the HeapSort code discussed in class and in your book, and to construct a class from it that matches our previous sort classes: MergeSort, QuickSort, etc.

Tasks:

1) Begin with a working implementation of HeapSort (like what we provided in class, or from your book)
2) Refer to the implementations of MergeSort, QuickSort, etc. Note that these are in classes and invoked with the function *sort()*.
3) Construct a HeapSort class which enables this same functionality.
    a) Note: you will likely have to change method signatures, define new variables, etc., to make this happen.
4) In your Main class, demonstrate the HeapSort functionality by defining a random Integer array and sorting it with HeapSort.sort().
5) Using your Stopwatch class, conduct an empirical comparison between HeapSort and SelectionSort.

The learning goal of this question is to familiarize yourself with adapting algorithms presented in concise forms, and converting them to implementations that can be used in general applications.

Output

The output of this should include Java code TAs can inspect and run.

Q3: Heap by Hand (20 pts)

Demonstrate your Heap abilities by constructing a heap by hand:

(a) Show the step by step construction of the MaxHeap tree for the following sequence of inserted values:

22 38 5 40 20 38 25 47 5 19

Show the final heap tree after each insertion (you do not need to show the swaps).

(b) Build the Heap array (the indices) corresponding to the tree that you built in the previous step.

Output

The output of this should be a PDF, either typed out or handwritten / drawn.