

# COMP 410 - Spring 2017

## Programming Assignment 3

Due back by 5:00 pm on March 3



This assignment builds off Assignment 2. For this assignment, you are to

- 1) Implement the *priority queue* ADT (specified in interface form below) as a **binary heap** – in Assignment 2, you had implemented it as a sorted array;
- 2) use your priority queue implementation to sort an array – this is unchanged from Assignment 2;
- 3) compare the efficiency of this sorting algorithm with another one implemented using the `java.util.PriorityQueue` implementation of priority queue;— this is unchanged from Assignment 2; and
- 4) Implement heap-sort as an in-place sorting algorithm for sorting doubles, and compare its efficiency with the other two – this is new to the current assignment.

Instructions from Assignment 2 are reproduced below; additions made for assignment 3 are written in **red**.

The **priority queue interface** that you are to implement is as follows:

```
public interface PQ<C extends Comparable<?super C>> {
    public boolean isFull();
    public boolean isEmpty();
    public void insert(C data); //Precondition: Not full
    public C min(); //Precondition: Not empty
    public C deleteMin(); //Precondition: Not empty
}
```

Below we provide a **skeleton for your implementation** of this interface; complete this skeleton by filling in the three methods `insert()`, `min()`, and `deleteMin()`. **You may also choose to declare some private methods here.**

```
public class PQasSortedArray<C extends Comparable<?super C>> implements PQ<C> {
    private C[] arr; // store the elements in the priority queue IN SORTED ORDER
    private int currentSize;
    public PQasSortedArray(int size){
        arr = (C[]) new Comparable[size];
        currentSize = 0;
    }
    public boolean isFull(){return currentSize == arr.length;}
    public boolean isEmpty(){return currentSize == 0;}
    public void insert(C data){ /*fill in the details here*/}
    public C min(){ /*fill in the details here*/}
    public C deleteMin(){ /*fill in the details here*/}
}
```

**Implement `heapSort` as a static method in your main program. It should look as follows**

```
public static void heapSort(double[] arr){ /*fill in the details here*/ }
```

**You may find it helpful to additionally define and use some private methods**

**Note.** You may *not* use Java's built-in `sort` methods within your method implementations — explicitly implement all sorting routines on your own.

**Using, and evaluating, your priority queue.** You are to use your priority queue implementation in a main class that

- 1) declares and fills in an array of a (specified) constant size  $N$  with random Doubles — read up on `java.util.Random` to learn how this is to be done.<sup>1</sup>
- 2) **Sorts** this array using your priority queue implementation. Let `arr1[]` denote an array to be sorted; the following code sorts it:

```
for (int i=0; i < arr1.length; i++)pq1.insert(arr1[i]);
for (int i=arr1.length-1; i >=0 ; i--)arr1[i] = pq1.deleteMin();
```

You should determine the amount of time your code takes to sort the array. You can time the performance of any piece of code in the following manner:

```
long startTime, endTime;
startTime = System.nanoTime();
//
//CODE TO BE TIMED GOES HERE
//
endTime = System.nanoTime();
System.out.println("Took " + ((endTime - startTime)/1000000) + " time units");
```

- 3) Repeat step 2 above (sort and time) for the same array, this time using the `java.util.PriorityQueue` implementation of priority queues. The sorting step would look like this:

```
for (int i=0; i < arr1.length; i++)pq2.offer(arr1[i]); // "offer()" is insert()
for (int i=arr1.length-1; i >=0 ; i--)arr1[i] = pq2.poll(); // "poll()" is deleteMin()
```

- 4) **Repeat step 2 (sort and time) for the same array, this time by making the call `heapSort(arr1)`**
- 5) Compute the running time of **all the** sorting algorithms for a range of values of  $N$ . Based on these computed values, estimate BigOh running times for **each approach**.

**Submission instructions.** You should upload the following three files in a .zipfile to Sakai

- 1) A file titled `PQasSortedArray.java`, containing your implementation of the priority queue interface
- 2) A file titled `Tester.java`, containing your “main” class.
- 3) A pdf file listing, in tabular form, running times of **all the** sorts for different values of  $N$ . Explain how these values were obtained (e.g., “I ran the code 12 times for each value of  $N$ , discarded the largest and smallest running times, and took the average of the remaining 10”).

1. As you did in Assignment 2, you should make multiple copies of this array so that each sorting algorithm gets the same input.