Programming Assignment 5

Due 21 OCT @ 11:59pm

Write a program that implements the MinPQ interface using a binary heap (a.k.a. MinHeap). A template is provided that will read in and execute a sequence of operations. The constructors use either the default capacity to start with or a number specified. The MinHeap should grow as necessary (do not worry about shrinking).

When you have your implementation complete, run a series of experiments comparing the running time of the binary heap versus an ordered array. The size of the input n should proceed as follows: 100, 1000, 10000, 100000, 1000000. The application PQExperiment will print out the time. Write up an analysis of the experiment and specifically comment on how the implementations compare for small values of n versus large values. The analysis must include the results of the timing experiments. The analysis should be short, one or two paragraphs.

The assignment has two ways to test your implementation, using the MinHeap.main method and using JUnit (which is more comprehensive). You will have to adjust compile and run as shown in lecture to use the JUnit approach.

Grading Notes

You must:

- Use the template provided for you
- Have a style (indentation, good variable names, etc.)
- Comment your code well (no need to over do it, just do it well)

You may not:

- Make your program part of a package.
- Use *code* from anywhere except your own brain.

Submission Instructions:

- Name a folder with your gmu username
- Put your java files in the folder (but not your .class)
- Zip the folder (not just the files) and name the zip “username-pa2.zip”
- Submit to blackboard

Grading Rubric

No Credit:

- Non-submitted assignments
- Late assignments
- Non-compiling assignments
- Non-independent work

1pt	Submission Format
1pt	Style and Comments
1pt	insert/checkGrow
2pts	sink
1pt	delMin
2pts	swim
1pt	min, iterator
1pt	Analysis

Example MinHeap.main Run

```
> java MinHeap operations.txt
insert=79
insert=45, 79
insert=-76, 79, 45
insert=-76, 73, 45, 79
insert=-76, 66, 45, 79, 73
insert=-76, 66, -40, 79, 73, 45
insert=-76, 66, -40, 79, 73, 45, 99
insert=-76, 19, -40, 66, 73, 45, 99, 79
insert=-76, 19, -40, 19, 73, 45, 99, 79, 66
insert=-76, 19, -40, 19, 27, 45, 99, 79, 66, 73
size=10
delMin=-76
delMin=-40
delMin=19
delMin=19
delMin=27
delMin=45
delMin=66
delMin=73
delMin=79
delMin=99
insert=94
insert=24, 94
insert=24, 94, 89
insert=-52, 24, 89, 94
insert=-52, 24, 89, 94, 28
insert=-52, 24, -2, 94, 28, 89
insert=-52, 24, -2, 94, 28, 89, 80
insert=-52, 24, -2, 70, 28, 89, 80, 94
insert=-52, 24, -2, 70, 28, 89, 80, 94, 99
insert=-52, 24, -2, 70, 28, 89, 80, 94, 99, 66
delMin=-52
delMin=-2
delMin=24
min=28
delMin=28
size=6
Final priority queue=66, 70, 80, 94, 99, 89
```

Example JUnit Compile and Run

```
> javac -cp .:./junit-4.11.jar TestMain.java
> java -cp .:./junit-4.11.jar TestMain
>
```

Empirical Runs

> java PQExperiment 10
BinaryHeap took = ???ms
OrderedArray took = ???ms

> java PQExperiment 100
BinaryHeap took = ???ms
OrderedArray took = ???ms

> java PQExperiment 1000
BinaryHeap took = ???ms
OrderedArray took = ???ms

> java PQExperiment 10000
BinaryHeap took = ???ms
OrderedArray took = ???ms

> java PQExperiment 100000
BinaryHeap took = ???ms
OrderedArray took = ???ms

> java PQExperiment 1000000
BinaryHeap took = ???ms
OrderedArray took = ???ms