CS206 Introduction to Data Structures
Homework 6
Priority Queues
Due: Oct 30, 2020 prior to 11:59 pm

There are two parts to this homework. In the first part you will adapt the heap-based priority implementation discussed in class. In the second part you will investigate the speed of several priority queue implementations.

Part 0
Create a new folder, presumably named Assignment6, and copy the following code into it.

BookReader.java
PriorityQInterface.java
AbstractPriorityQ.java
PriorityQueue.java
PriorityQueueSAL.java
PriorityQHeap.java
SAL.java

Also get:

descent.txt

(Yes, this is the full text of “The Descent of Man” by Charles Darwin. It has about 310,000 words.) Finally, for development it might be handy to get:

d1000.txt

which is about the first 1000 words of descent.txt.

All of the above is available on powerpuff in the directory /home/gtowell/Public/206/a6/. Use cp or scp as needed to copy this code to the directory containing the work for this assignment.

Part 1

PriorityQHeap.java implements a binary heap, which might also be called a 2-heap. That is, each node in the tree has children has exactly 2 children (yes there is at most one node that has exactly 1 child). Now consider a N-Heap. In this version of the heap, every node, possibly except one, has exactly N children. There may be one node that has one, two, three, ..N-1 children. (There may not be one node with exactly 1 child and a different node with exactly 2 children, etc.) Create a new class PQNHeap (or some other likely name) that implements a priority queue based upon a N-Heap. You should implement PQNHeap by extending PriorityQHeap. That is, the first line of code for PQNHeap should be:

```
public class PQNHeap<K extends Comparable<K>, V>
extends PriorityQHeap<K, V>
```

Further your code should include a constructor that that allows you to specify N. For instance,
public PQNHeap(int nnn, int capacity)
I would imagine that the only methods in PriorityQHeap that you need to override are: offer and removeTop, but tat is somewhat dependent on you.

It is considerably easier to build PQ3Heap which is a version of PQNHeap that handles only N=3. If you are unsure of how to write PQNHeap, write instead PH3Heap. PQ3Heap is worth 8 points less than PQNHeap. (I encourage everyone to start with PQ3Heap and only do PQNHeap when you are completely sure that PQ3Heap works perfectly.

As a place to start, draw pictures of exactly what you need to do with PQ3Heap. If you cannot get PQ3Heap working, submit pictures with detailed explanations. For instance draw a picture of the tree representation of a 3 heap. Show the array location of each node in the heap. Then, draw a picture of exactly what happens when you insert and when you remove. In you picture, show each decision that needs to be made at every step. Just drawing such a set of pictures is non-trivial (if you are worried about neatness) but it is pretty important to organizing you thoughts about the code that you would need to write.

Part 2
Timing Priority Queues

Collect the data necessary to fill in the following table

	Offer	**poll**	**total**
PriorityQueue			
PriorityQueueSAL			
PriorityQHeap			
PQNHeap with N=4			

for the file descent.txt. (If you wrote PQ3Heap, use that for the last line of the table). Note that you can collect data for the first 3 lines of this table without writing any new priority queue code.) Use the venerable BookReader class to read this file word by word, adding each word to the priorityQueue, where the word is both the key and the value. For example, the following code will add the first word in descent.txt to an instance of PriorityQHeap and then immediately remove that word from the priorityQueue:

```
 BookReader br = new BookReader("descent.txt");
PriorityQHeap<String, String> pqb =
             new PriorityQHeap<>(Ordering.MIN, 500000);
String w = br.nextWord();
pqb.offer(w,w);
pqb.poll();
```

Unlike this example you should add every word of descent to the priority queue before removing any. For PriorityQueue and PriorityQueueSAL, you can expect that this will take quite a while (say 10 minutes or more). If you feel like you are about to melt your laptop, adjust you code so that you only build priority queues on the first 100,000 words of descent.txt. If this is still too much, then build on only the first 50,000 words.

When creating instance of PriorityQHeap and PQNHeap give them space for at least 400,000 items.

Include this table in your README. Once you have completed this table, also in the Readme file explain the relationship among the times. For instance, if your data says that offer is slower than poll for PriorityQHeap, explain that. Asymptotic analysis (Big-O) would probably be useful. Do not limit your explanations to rows, explain the differences in columns as well.

Note that the timing data and discussion will be a significant part of the grade for this assignment and that most of this can done without a PQNHeap.

What to turn In

1. All code you wrote or edited, well commented.
2. README with the usual information. In addition, the readme should include the table and discussion from part 2.

Standard Boilerplate:
Your program will be graded based on how it runs on the
department's Linux server, not how it runs on your computer.

The following steps for submission assume you created a project named `AssignmentN` in the directory `/home/YOU/cs206/` on the CS department UNIX computers (e.g. powerpuff)

- For this assignment N=6
- Put the README file into the project directory
- Go to the directory /home/YOU/cs206
- Enter `submit -c 206 -p N -d AssignmentN`

For more on using the submit script http://systems.cs.brynmawr.edu/Submit_assignments