Lab 9 – Priority Queues

Due Sunday, October 18 at 23:59

Objective

Today's lab will help you get familiar with

- Use and efficiency of priority queues.

More about Priority Queues

In class you are covering priority queues and their implementation via heaps. This lab will focus on comparing your own array-based priority queue to the heap-based **PriorityQueue** class provided in **java.util.PriorityQueue**. The API page should be considered the primary resource for this assignment.

Assignment:

For this lab assignment, you will write both a **WeightedElement** class and an **ArrayPriorityQueue** class, and use them to compare performance of the naive array priority queue to the performance of the standard Java **PriorityQueue.**

1. Write a generic **WeightedElement<E,W>** class which stores an element of type **E** and a weight of type **W.** It should implement **Comparable** relying on **W**'s **compareTo()**. You should enforce that **W** itself is comparable.
2. Write a generic **ArrayPriorityQueue<T>** class. It should store an **ArrayList** of type **T**. You should enforce that **T** is comparable. This class will implement a priority queue via three methods:

a) **boolean add(T t)**: Insert **t** at the end of the **ArrayList**, returning true if successful
b) **T poll()**: Find the minimal element, remove it from the **ArrayList**, and return it. Use a simple linear scan.
c) **T peek()**: Find and return the minimal element, without removing it.

3. Unit test using the Java **PriorityQueue** to check against. Use **WeightedElement<Integer,Integer>** and **WeightedElement<String,String>**. Make sure to fully explore edge cases and generic cases.
4. Write an **ExperimentController** for evaluating runtimes. Use random ints to time creation of large priority queues of both kinds, timing average times over multiple runs. Also use multiple runs to average timing a series of **poll** and **peek** operations for comparison.

Submission:

For this lab, you need to submit the code and notes as usual.

Grading:

1. unit testing - 2
2. **WeightedElement** – 2
3. **ArrayPriorityQueue** – 3
4. lab notes - 2
5. Style/commenting - 1