CSCI 240 – PA 6

Priority Queues (PQ) and Heaps

Feel free to discuss and help each other out but does not imply that you can give away your code or your answers! Make sure to read all instructions before attempting this lab.

You can work with a lab partner and each one must submit the same PDF file (include both names in the submission file). Each person must include a brief statement about your contribution to this assignment.

You must use an appropriate provided template from Canvas and output "Author: Your Name(s)" for all your programs. If you are modifying an existing program, use "Modified by: Your Name(s)".

Exercise 1: Put together list priority queue (**ListPriorityQueue** for C++ and **SortedListPriorityQueue** for Java) and use a test driver to perform some operations to confirm it is working correctly. Use a PQ with integer as element. Create two PQ objects – one with the largest value having highest priority and one with lowest value having highest priority. **Be sure to use a comparator for the PQ**.

Perform the following operations: insert(5), insert(4), insert(7), insert(1), min(), removeMin(), insert(3), insert(6), min(), removeMin(), min(), removeMin(), insert(8), min(), removeMin(), insert(2), min(), removeMin(), min(), removeMin(). Output value for each min() operation so there should be 6 values for each set of output.

Exercise 2: Use your priority queue from exercise 1 to sort data in ascending order. Sort the data file ***small1k.txt***, containing a list of 1,000 integer values, and output the first 5 and last 5 values to the screen (5 values on one line and at least one space between the 2 values). Sort the data file ***large100k.txt***, containing a list of 100,000 integer values, and output the first 5 and last 5 values to the screen (5 values on one line and at least one space between the 2 values). For each set of data, collect actual run times in milliseconds and display to the screen as well. Confirm that the runtime is O(n^2).

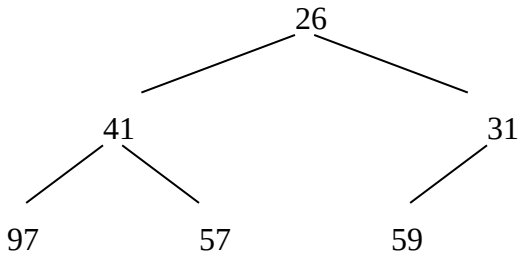
Exercise 3: Put together the heap priority queue (**HeapPriorityQueue** for C++ and Java) and use same test driver from exercise 1 to perform some operations to confirm it is working correctly. You can use a PQ with integer as element. Create two PQ objects – one with the largest value having highest priority and one with lowest value having highest priority. **Be sure to use a comparator for the PQ**.

Perform the following operations: insert(5), insert(4), insert(7), insert(1), min(), removeMin(), insert(3), insert(6), min(), removeMin(), min(), removeMin(), insert(8), min(), removeMin(), insert(2), min(), removeMin(), min(), removeMin(). Output value for each min() operation so there should be 6 values for each set of output.

Exercise 4: Use your priority queue from exercise ~~1~~ 3 to sort data in ascending order. Sort the data file ***small1k.txt***, containing a list of 1,000 integer values, and output the first 5 and last 5 values to the screen (5 values on one line and at least one space between the 2 values). Sort the data file ***large100k.txt***, containing a list of 100,000 integer values, and output the first 5 and last 5 values to the screen (5 values on one line and at least one space between the 2 values). For each set of data, collect actual run times in milliseconds and display to the screen as well. Confirm that the runtime is O(nlogn).

Question 1: What is a PQ? How is it different than a queue?

Question 2: Given the heap below, show the resulting heap after each of the following operations -- removeMin(), insert(30)



You can earn EC for only one of the options below:

Extra Credit Option 1: Add code to your exercise 4 to sort the data file ***small1k.txt*** in descending order and output data to an output file as well with 5 values per line (at least one space between the 2 values). You can submit one version here to include exercise 4. Make sure to provide the first screenshot of the output file as part of the submission.

Extra Credit Option 2: Provide pseudocode for insert() and removeMin() operations using a linked binary tree for the heap. *Hint: determine or keep track of the last node in the binary tree.*

Fill out and turn in the PA submission file for this assignment (save as PDF format).