Calculating Large Factorials

Purpose:

To gain familiarity with the *use* of priority queues.

Due Date:

This assignment is due Monday, 2018-10-19 *at the beginning of lecture*.

Stopwatches

Use the `StopWatch` class that you created for Lab Assignment 1.

Re-read the "Instructions on Plotting and Timing" on Casperson's web-site.

Factorial using `BigIntegers`

The goal of this lab assignment is to compute $n! = n \cdot (n-1)! = n \times n - 1 \times \cdots \times 2 \times 1$ effiicently and accurately for large integers.

We use the `java.math.BigInteger` class for the actual arithmetic. As you learned in Lab Assignment One, multi-digit multiplications take non-trivial time.

Although multiplication is communative [$a \times b = \times a$] and associative [$(a \times b) \times c = a \times (b \times c)$], the order of the multiplications matters to the efficiency of the computation. For instance to compute $6!$, first computing $1 \times 2 \times 3 = 6$, then $4 \times 5 = 20$ then $6 \times 6 = 36$, then $20 \times 36 = 720$. is slightly more efficient than computing $((((1 \times 2) \times 3) \times 4) \times 5) \times 6$.

We generally want to multiply numbers of approximately the same size when we can arrange to do so.

Here is an algorithm that approximates this for computing $n!$.

(I) Put the numbers 1 to n in a priority queue Q.
(II) Extract the minumum number from Q into a.
(III) If the queue is now empty, the answer is a.
(IV) Otherwise extract into b, the smallest remaining number in Q.
(V) Re-insert the product $a \times b$ back into Q, and go back to Step II.

⇒ Write *two* methods to compute $n!$ using the `BigInteger` class. The first should compute $n!$ “na´ïvely by by multiplying 1 through n in order.

The second should use a priority queue as outlined above. Use the `java.util.PriorityQueue` class to implement the priority queue. (Useful `PriorityQueue<BigInteger>` methods include `.add(`*n*`)`, `.remove()`, and `.isEmpty()` .)

⇒ Using your `StopWatch` class, measure the time to compute $n!$ for n a multiple of 10 000 [10^4] less than or equal to 100 000 [10^5] for both methods.

⇒ Plot your data. (See the “Instructions on Plotting and Timing”.)