A) The goal here was to benchmark the performance of a linked list implementation. I expanded the goal slightly to build a modular benchmarking system.
B) While there isn't much algorithm design in this project, I had a bit of flexibility in the form of my implementation, and I chose to optimize operations I thought would be common, like insertions at the front and back. This meant min and max were (in the improved version) constant time. I also altered my insertion algorithm to iterate manually, a large optimization. Also, I left in the old, worse "naive" versions of the algorithms to see their performance compared to the new ones. A doubly linked list could have made a few optimizations possible but, at least to my knowledge, I implemented any improvements I could think of. I wish I could have found a more efficient way to sort element-wise items like merge sort, but I have no clue how that would be implemented.
C) I ran this on a 1.4 GHz Quad-Core Intel Core i5 processor, with 8GB of memory and a 121GB SSD
   I repeated the experiment with the first input file several times, but as the longer one took so long to run, I was only able to run it once all the way through, but all partial experiments yielded similar results.
   I used javac 14.0.2 on a Macbook running macOS 10.15.5
D) The data of the bench is listed below. Min and max are the fastest, and almost the same, in each file, due to the constant time retrieval from the front or back of the sorted list. The median algorithm is roughly linear, so it is slower than both min and max. Insert, however, ended up being O(n^2), so it absolutely exploded, being 3 orders of magnitude larger than constant time operations on the first file and 9 orders of magnitude on the second.
   Input1.txt
   a. Values
      i. Min:1
      ii. Med: 2056
      iii. Max: 4000
   b. Fastest time
      i. Insert: 7976175ns (about 8ms)
      ii. Min: 9618ns (about 10µs)
      iii. Med: 15893ns (about 15µs)
      iv. Max: 9736ns (about 10µs)

   Input2.txt
   a. Values
      a. Min: 39
      b. Med: 4003661
      c. Max: 7999978
   b. Fastest time
      i. Insert: 1665358375838ns (about 30 mins)
      ii. Min: 8716ns (about 9µs)
      iii. Med: 3694648ns (about 3ms)
      iv. Max: 8432ns (about 8µs)