CPSC 222: Concurrent Convex Hull Implementation Assignment 2

University of Northern British Columbia

Department of Computer Science

Mason Legere
230128712

January 13, 2019
`https://github.com/MasonLegere/concurrentConvexHull`
Main Method in public class `ConvexHull`

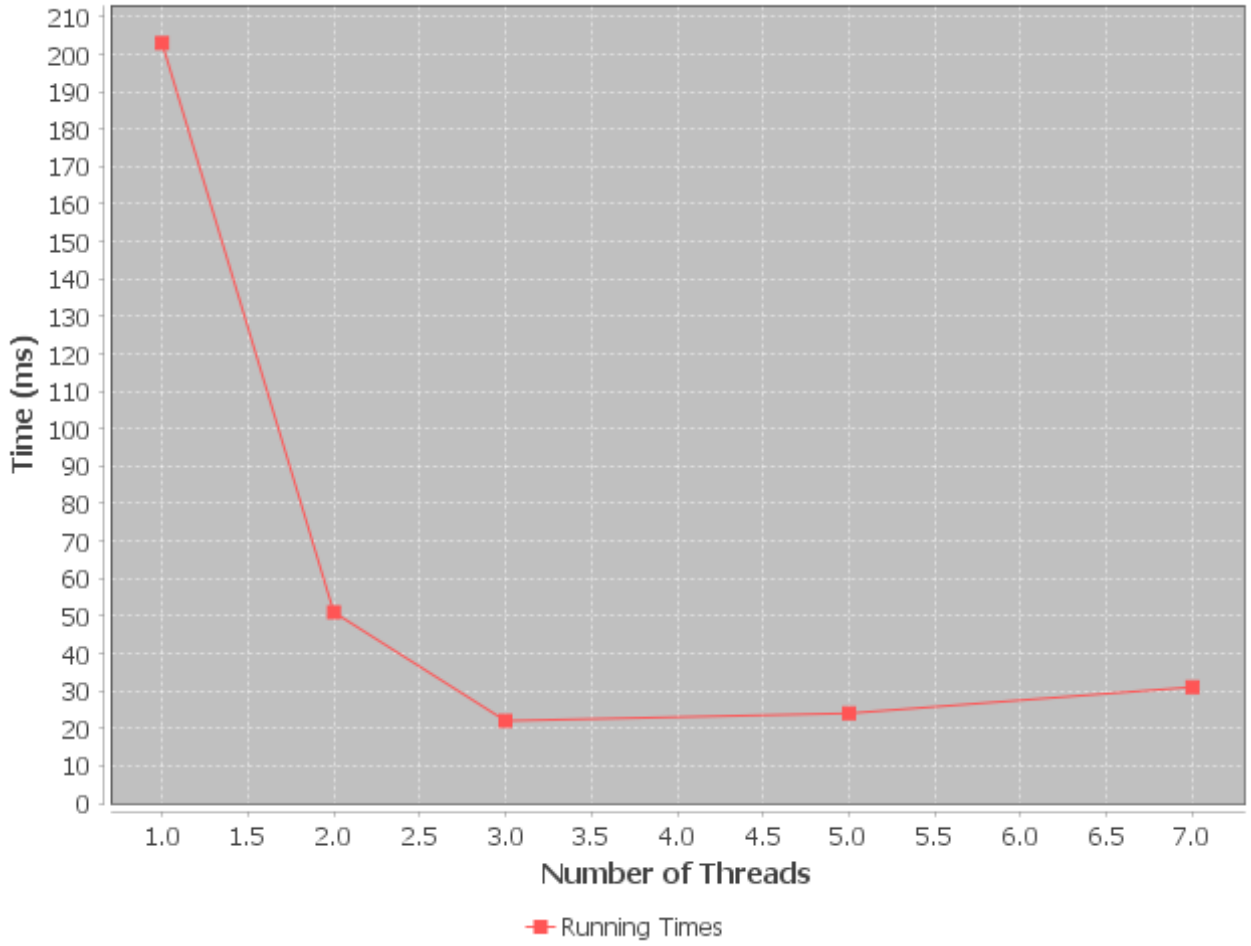

Figure 1: Line plot of running times as a function of the number of threads.

Results

Within the simulation we ran the program using 75000 random points within $\mathbb{N}^2_{\leq 75000}$. The points are generated by creating two arrays listing the positive integers from 1 to 7000, shuffling each array, and then combining each array component-wise to make the lattice points. In theory we would expect to see a super-linear decrease in the running times as a function of the number of threads . Further, we would expect to see this decrease up until the logical processors of my computer become saturated. In this case, as the results were found on my laptop, we expect to decreases in running time up to using three threads. A super-linear decrease rather than a linear amount is expected because by finding the convex hull of sub-lattices and merging them together afterwards we are reducing the overall number of operations needed.
From the plot we see what we would expect showing a decrease in running time as a function of the number of threads. Further we see that the minimum run time occurs when the program runs with three threads. This is in agreement with the computer that it is being ran on. We also see a very large value for the first simulation using one thread. I believe that this large value is due initial memory caching and initialization procedures.