Hypothesis. Insertion sort is faster than quicksort for input arrays of size 50 or smaller.

Methods. The experiment is ran on Ubuntu 19.04. GCC version is 8.3 and python version is 3.7.3. Also, matplotlib is a dependency of plot.py

To run the experiment, first compile

g++ -fconcepts -std=c++2a -O3 code.cpp

Then execute

./a.out

To plot the results use

python3 plot.py

The code runs quicksort and insertion sort on arrays, of length n=1,...,400, containing uniform random integers from the range [0,2147483647]. For each input size n, the algorithms are tested on 10000 random input arrays of length n and the results are averaged. This produces n pairs of numbers. The ith pair of numbers, call it (a,b), represents a: the average time taken for quicksort on uniform random inputs of size i, and b: the average time taken for insertion sort on uniform random inputs of size i.

Results.

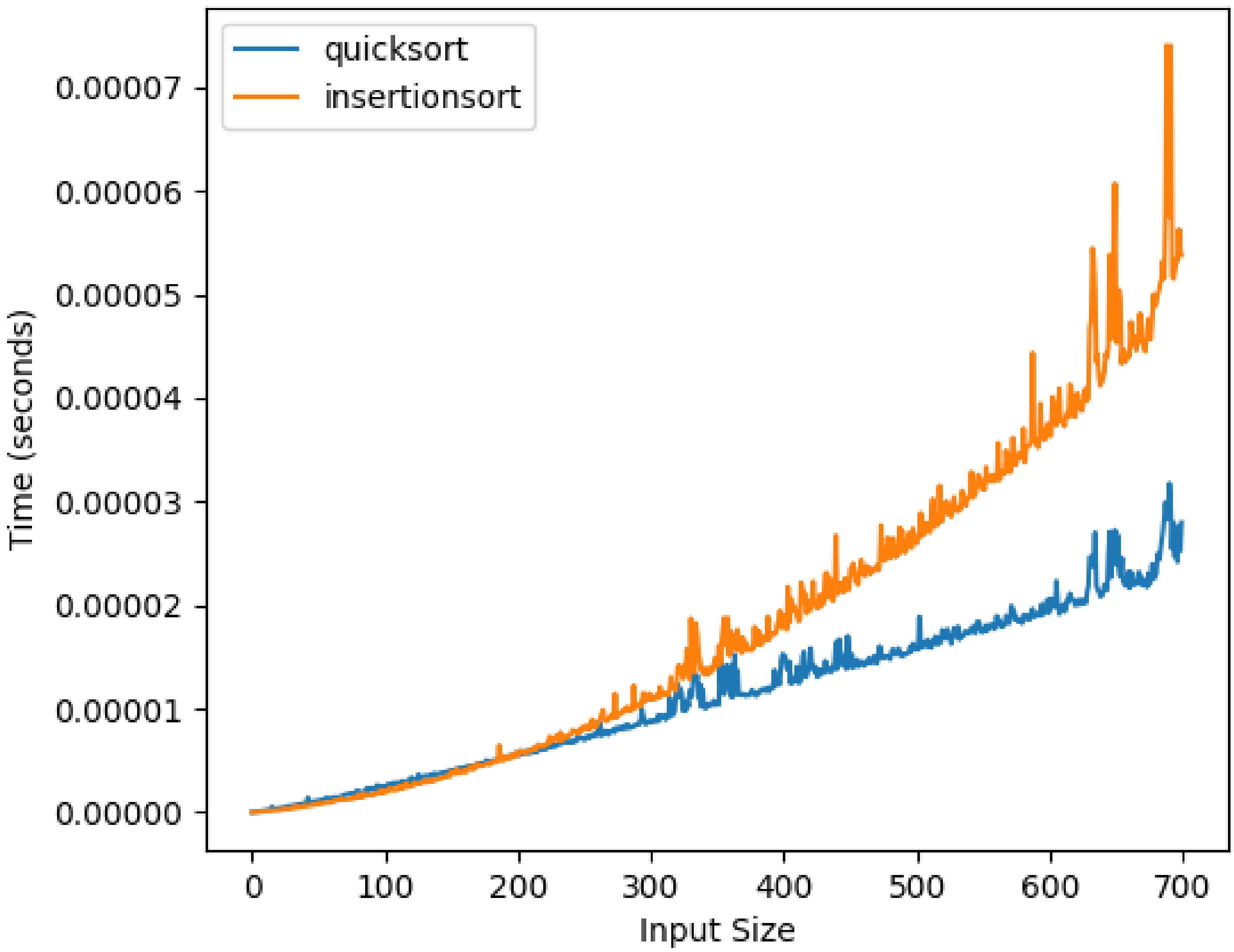

Figure 1

Figure 1 shows a plot of the n pairs of numbers described above.

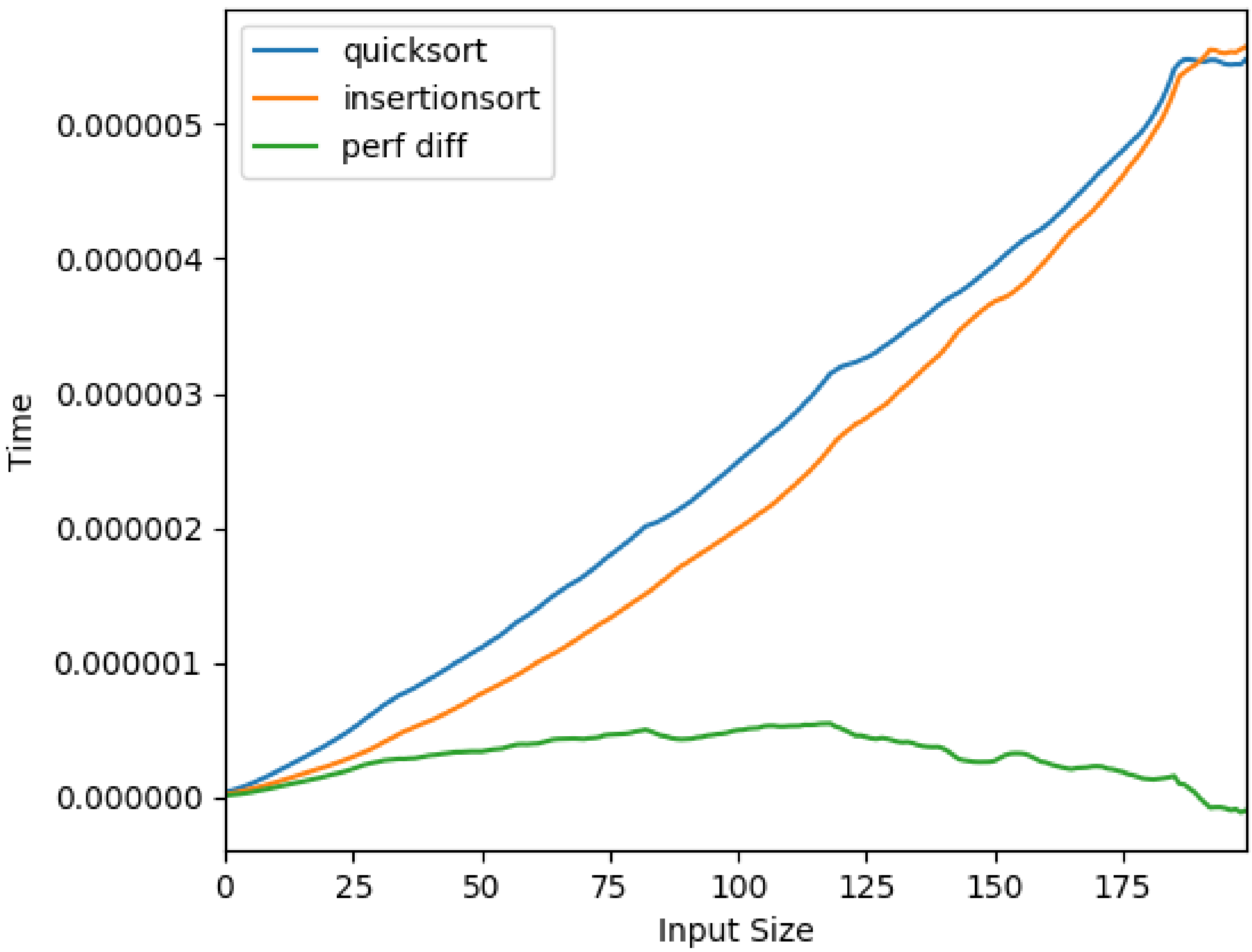

Figure 2

Figure 2 is a zoomed in plot of the data in Figure 1, but with an additional curve showing the difference in performance for quicksort and insertionsort.

Discussion. Figure 2 shows that quicksort takes the lead around input size 180. One thing to note is that the maximum performance difference between quicksort and inesrtionsort is somewhere between 75 and 115.

Conclusion. Quicksort is faster than insertion sort for n¿180.