GITHUB REPO LINK

https://github.com/mepittma/bmi206-1

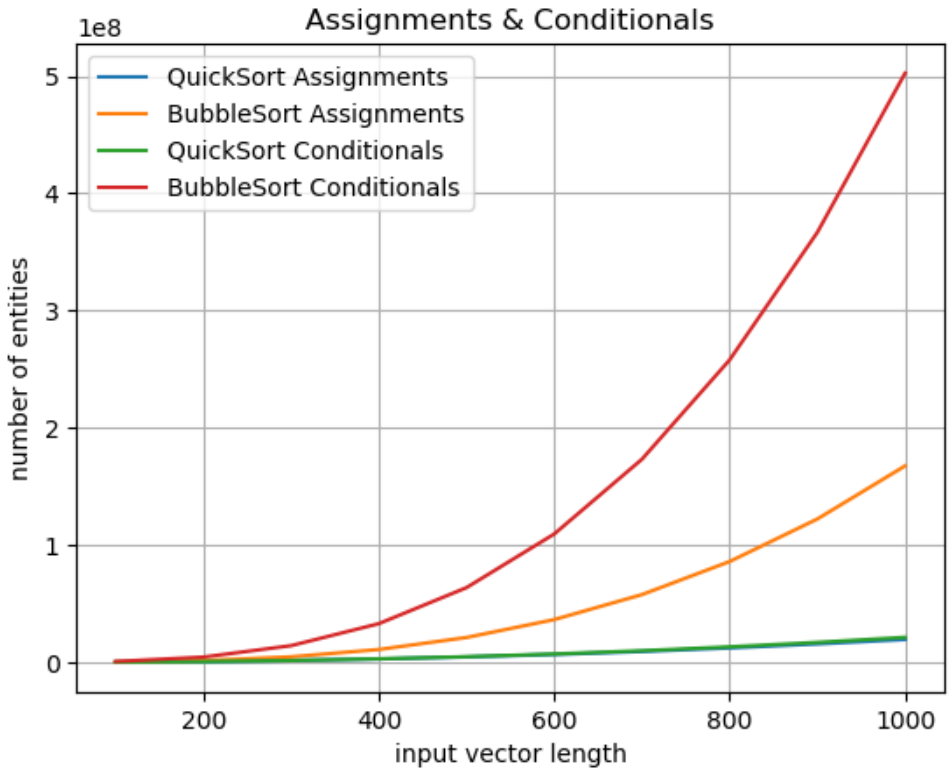
ASSIGNMENTS AND CONDITIONALS

For input vector sizes from 100 to 1000 (step = 100), QuickSort assignments and conditionals were on average (n = 100) roughly equivalent.

> QuickSort assignments: [166045.54, 692711.48, 1598331.15, 2912633.13, 4623432.37, 6720761.0, 9268777.47, 12278064.37, 15694206.22, 19486806.15]
>
> QuickSort conditionals: [157796.51, 674908.44, 1605036.68, 2981901.58, 4817619.82, 7115650.27, 9908561.79, 13216411.15, 17017329.07, 21299378.77]

The BubbleSort algorithm's assignments and conditionals grew at a much faster rate (Figure 1).

> B assignments: [248468.54, 1501066.98, 4746938.38, 10997633.18, 21235115.3, 36501241.52, 57762222.0, 86115097.8, 122447337.28, 167789653.28]
>
> B conditionals: [749925.0, 4499850.0, 14249775.0, 32999700.0, 63749625.0, 109499550.0, 173249475.0, 257999400.0, 366749325.0, 502499250.0]



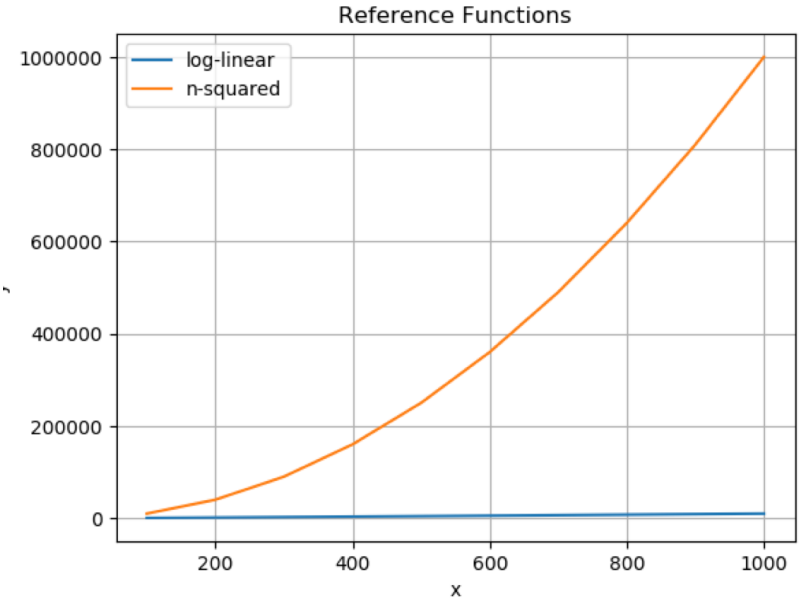
AVERAGE TIME COMPLEXITY

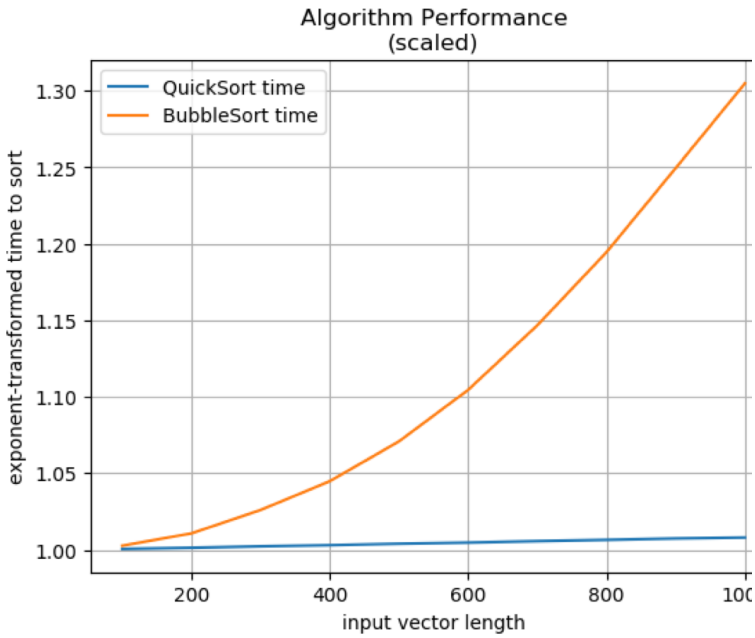
Theoretical complexities:

BubbleSort: n^2

QuickSort: nlog(n)

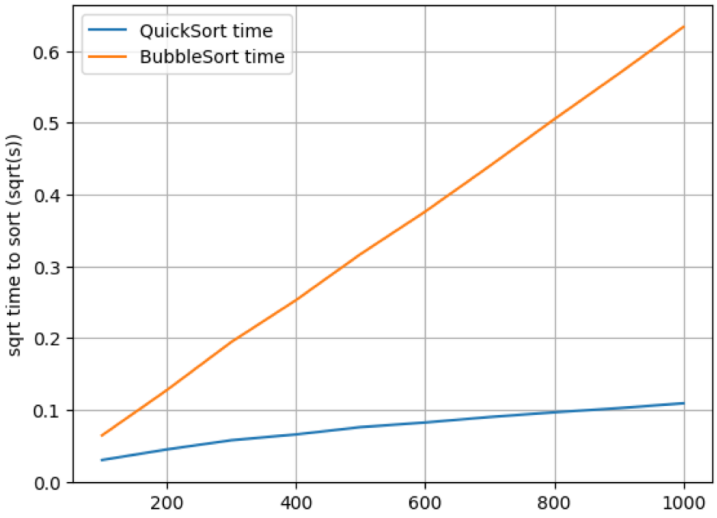
Comparing an untransformed graph of average runtimes (over 100 trials) for n = [100, 200, 300, …, 1000] to a reference graph showing n^2 and nlog(n) functions is consistent with the theoretical complexities.



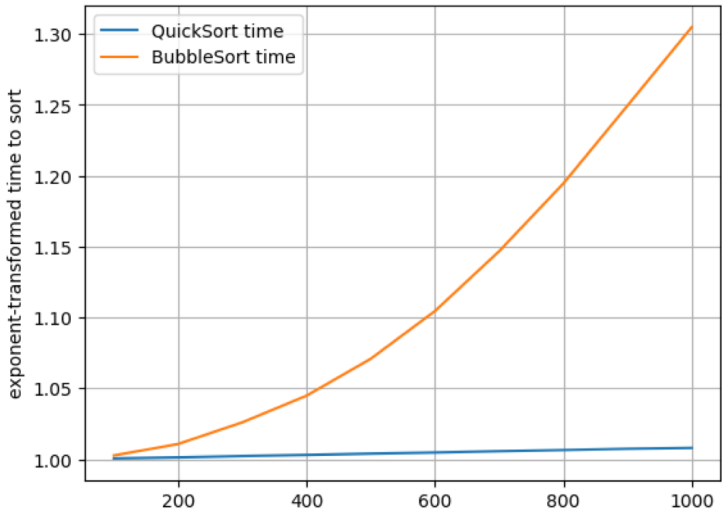


To confirm that my implementation matches these theoretical complexities, the sort-time data were transformed and graphed to show linearity. In theory, an algorithm with quadratic time complexity should graph linearly when runtime t is raised to the ½ power; an algorithm with nlog(n) time complexity should graph linearly when runtime t is transformed by 2^t. This was indeed the case with my implementation:

Time Transformation: sqrt(t)



Time Transformation: 2^t



Input Vector Size (n)