Daniel Weitman
COSC 220
11/19/18

O(n^2) sorts

I) The insertion sort consistently sorted the array faster for all array sizes tested, whereas the bubble sort consistently sorted the slowest and the selection sort in between the them. Therefore, no algorithms were better at sorting smaller arrays then large arrays and vice versa, for times tested. For the smaller array sizes (< 10,000) the time discrepancy between all three algorithms was negligible, as they were all completed in under a second. However, as array sizes increased the discrepancy between the sorting times grew as well. All three algorithms displayed and exponential trend in their respective graph.

II)
Bubble sort: y = 0.1514248 – 0.000009415224*x + 3.462952 * 10^(-9)*x^2
Insertion sort: y = 0.179842 – 0.000003385988*x + 8.249646 * 10^(-10)*x^2
Selection sort: y = .07000672 + 0.000004268469*x + 1.475371 * 10^(-9)*x^2

O(nlog(n)) sorts

I) Similar to the O(n^2) sorts, the merge and quick sorts had no intersections between them for array sizes tested. Therefore, the merge sort performed faster for all tests. Yet, unlike the O(n^2) sorts the merge and quick sorts had a linear trend showed in their graph. This means that the time discrepancy was growing as array size increased but not to as high a degree. So both algorithms were comparable to each other in sort time for all times tested even though merge was slightly faster in all instances.

II)
Quick sort: y = y = 47.25613 + (-6.402776 - 47.25613)/(1 + (x/6500087)^0.7745467)
Merge sort: y = 142983.8 + (0.1135715 - 142983.8)/(1 + (x/361826800000)^1.064355)