Andrew Smith and Dakota Donner

Random Array

The Bubble Sort on a random array had a large number of comparisons, with a large number of swaps. The Insertion Sort had roughly half the number of comparisons as the bubble sort, but an identical number of swaps. The Selection Sort had the largest number of comparisons (slightly higher than the bubble sort), but by far the least number of swaps. Despite running more comparisons, the selection sort seems to have performed the best of the three with a dramatically lower number of swaps, while the bubble sort performed the worst and the insertion sort was between the two in performance.

Sorted Array

Because this array was already sorted, no sorts performed any swaps. The Bubble Sort and Insertion Sort performed an identical number of comparisons, while the Selection Sort boasted a huge number of comparisons. Based on this, the bubble sort and insertion sort performed identically well, with the selection sort being horrendously behind the other two.

Nearly Sorted Array

The Bubble Sort performed a fair number of comparisons and a few swaps. The Insertion Sort had significantly less comparisons than the selection sort, with an identical number of swaps. The Selection Sort performed a dramatically higher number of comparisons but performed significantly less swaps than the other two sorts. The insertion sort seems to have performed the best with this array, with selection sort performing the worst and bubble sort between the two.

Reverse Array

In this scenario, Bubble Sort, Insertion Sort and Selection Sort performed an identical number of comparisons, with bubble sort and insertion sort also performing an identical number of swaps. Selection sort, however, performed significantly less swaps, making it the top performer with bubble sort and insertion sort bringing up the rear.

Duplicates Array

Bubble Sort and Selection Sort had an almost identical number of comparisons, while bubble sort and insertion sort had an identical number of swaps. Insertion sort had around half the number of comparisons as the other two, while selection sort sported a dramatically lower number of swaps. Based on this, selection sort likely performed the best with its lower total of swaps, with insertion sort in the middle and bubble sort coming up last.

-Selection sort has a dramatically lower number of swaps on average, only really struggling in scenarios where the array is already sorted or nearly already sorted. In these scenarios, an insertion sort might be preferable.

-As selection sort sports by far the least amount of swaps, this information suggests it is likely the best slow sorting algorithm overall.