|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time taken for sorting algorithms to sort arrays of different sizes (in seconds) | | | | | | | | | | | | | | | | |
| Array to Sort | Descending Array | | | | Random Array | | | | Nearly Sorted Array | | | | Average | | | |
| Size: | 10 | 50 | 100 | 1000 | 10 | 50 | 100 | 1000 | 10 | 50 | 100 | 1000 | 10 | 50 | 100 | 1000 |
| Bubble | 0.000997 | 0.001828 | 0.006056 | 0.468792 | 0.001749 | 0.006848 | 0.014758 | 0.460297 | 0.000531 | 0.001267 | 0.002723 | 0.220623 | 0.00109241 | 0.003314 | 0.007846 | 0.383237 |
| Selection | 0.000787 | 0.001945 | 0.005017 | 0.234836 | 0.001901 | 0.00662 | 0.014004 | 0.286232 | 0.000752 | 0.00192 | 0.003587 | 0.217669 | 0.00114638 | 0.003495 | 0.007536 | 0.246246 |
| Insertion | 0.000508 | 0.001562 | 0.003232 | 0.290299 | 0.00141 | 0.006732 | 0.011603 | 0.301206 | 0.000439 | 0.001557 | 0.001315 | 0.038392 | 0.0007856 | 0.003284 | 0.005383 | 0.209965 |

As shown by the average time for each array size over the 3 array types (see table above) the insertion sort was the fastest sorting algorithm across all sizes. The selection sort was the worst algorithm for integer arrays of size 10 and 50, whilst the bubble sort was the slowest for an integer array of size 100 and 1000. This suggests that although the insertion sort remains the best algorithm as the size of arrays is scaled up, the selection sort seems to have a better scalability than the bubble sort, performing better in the tests with a larger array size.

The worst case scenario for the bubble sort seems to be a descending array whilst it’s best case is a nearly sorted array. The worst case for the selection sort is a nearly sorted array, with it’s best case being a random array. The best case for an insertion sort is a descending array however it performs the worst with a randomly assorted array.