Array size : 5

	Bubble sort runtime	Selection sort runtime	Insertion sort runtime
Ascending	0.0	0.0	0.0
Descending	0.0	0.0	0.0
Random	0.0	0.0	0.0
Nearly	0.0	0.0	0.0

Array size : 60

	Bubble sort runtime	Selection sort runtime	Insertion sort runtime
Ascending	46.0	42.0	1.0
Descending	89.67	42.0	78.0
Random	65.67	39.0	33.66
Nearly	49.0	41.0	6.33

Array size : 200

	Bubble sort runtime	Selection sort runtime	Insertion sort runtime
Ascending	412.67	454.67	5.0
Descending	705.33	405.67	749
Random	434.0	316.33	311.67
Nearly	433.67	368.0	67.0

Array size : 1000

	Bubble sort runtime	Selection sort runtime	Insertion sort runtime
Ascending	1020.66	924.66	25.0
Descending	1155.67	843.33	862.0
Random	508.0	665.66	1923.67
Nearly	1059.0	877.66	674.0

Bubble sort
When an array is already sorted, it gives the best case time complexity O(n). It only needs one pass to check if the array is sorted. Average and worst case time complexity is O(n^2). When the array is in reverse sorted order, it gives the worst case.In the average case it needs the same number of comparisons as the worst case but fewer swaps to sort the array. Bubble sort is simple to implement but generally poor in speed.

Selection sort
In selection sort, it searches for the smallest item and swaps it with the first element. It only needs (n-1) passes and (n-p) comparisons. Time complexity of the selection sort is O(n^2). Because, there is only one swap for each pass. So best case = average case = worst case.

Insertion sort
Best case time complexity is O(n) when each element already is at the insertion point. If each pass needs to go through all elements, it gives the worst case O(n^2). If it only needs to go through half of elements, it is the average case O(n^2). So the insertion sort is very fast with almost sorted data.

According to above runtime results, Insertion sort is the fastest sorting algorithm. Next the selection sort algorithm and then the bubble sort algorithm.