Sort	Best Case	Average Case	Worst Case
Exchange	O(n^2)	O(n^2)	O(n^2)
Insertion	O(n)	O(n^2)	O(n^2)
Merge	O(n log(n))	O(n log(n))	O(n log(n))
Randomized QuickSort	O(n log(n))	O(n log(n))	O(n log(n)) / O(n^2)

Exchange Sort always compares one element with all of the elements on it's right no matter what i.e. it never checks for any conditions/patterns and goes on performing all the loops.
In the worst case, it'll check for all elements and put the smallest element in the first place.
Hence in the worst case it'll make O(n^2) comparisons.
Now even in the best case when the array is already sorted, it makes O(n^2) comparisons.
As it iterates through the algorithm without looking at the input order, average case will also be O(n^2).

Insertion Sort swaps elements only till it can find an element that is bigger than the key.
In worst case of input array in descending order. Every element will be swapped in every iteration Hence the number of comparisons and swaps will be of order of O(n^2).
In best case when the array is already sorted, the first comparison will result in a loop exit and hence the algorithm will iterate through the array once and exit. Hence best case complexity is O(n).
In average case, the algorithm will perform any amount of comparisons between the least n to the maximum of n(n-1)/2. Hence average case is O(n^2).

Merge Sort combines shorter arrays into the bigger one by scanning and comparing the shorter arrays. No matter what, it'll split the arrays and then build bottom up sorted arrays.
So it's design doesn't take care into consideration the pattern of the input.
Hence in every case, it's complexity will be O(n log(n)).

Randomized QuickSort places the pivot element at its correct position and then recursively does the same on the left and right subarrays. The selection of the pivot is element plays a crucial role in reducing the complexity of sorting from O(n^2) to O(n log(n)).
In best case and average case, it'll still keep partitioning elements and perform linear scan to partition. In best case, there would not be swaps and in average case, there would be a few swaps.
In worst case, complexity can be O(n^2) or O(n log(n)) depending upon randomization.

If the random pivot selection results in selecting pivots one after another index consecutively, it can result in a complexity of $O(n^2)$. The probability of this happening is almost zero. If we assume that there is no probability of this happening then even in the worst case this algorithm will be $O(n \log(n))$.