Sorting Algorithms

Algorithm	**Strategy**
Bubble Sort	On each run through the array, switch each two unsorted adjacent elements. In each subsequent run, exclude the last element included in the previous run.
Selection Sort	Assume the first element of the array is the smallest. Compare this minimum to the other elements in the unsorted array to find the actual minimum of this array. Set this minimum at the beginning of the array and consider this element sorted. Repeat this process for the now smaller unsorted array.
Insertion Sort	Assume the first element of the array is sorted. Increment the right index, and each time, sort it through the elements to its left. Repeat this process until you reach the end of the array.
QuickSelect	Using the partition method, we set the index of some element in its sorted position as a variable, which we call “pivot.” If the pivot is not k-1 (since the indexing starts at 0), we recursively call QuickSelect with the elements in the array between index pivot and the start/end (exclusive of pivot) until pivot = k-1.
QuickSort	Using the partition method, we set the index of some element in its sorted position as a variable, which we call “pivot.” Then we recursively call QuickSort to sort the elements between the start/end and pivot (exclusive of pivot).

Merge Sort	We split an array into subarrays and sort them by calling merge() recursively. In merge(), we set two small subarrays as left and right, and run add the elements of one array into the other such that the resulting array is sorted.

Pros/Cons and Time Complexity

Algorithm	Advantages	Disadvantages	Time Complexity		
			Best Case	Avg. Case	Worst Case
Bubble Sort	- Easy to understand and implement - The order of the elements of the array affects the time complexity of the sorting	- Its average and worst case time complexity is $O(n^2)$, which makes the sorting process very slow for large datasets	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	- Easy to understand and implement	- quadratic time complexity - inefficient for large datasets	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	- Can begin sorting before complete data set is available - fewer comparisons than in Bubble Sort	- quadratic worse-time complexity - inefficient for large datasets	O(n)	$O(n^2)$	$O(n^2)$
QuickSelect	- smaller average time complexity than bubble sort, selection sort, and insertion sort	- quadratic worse-time complexity	O(n)	O(n)	$O(n^2)$
QuickSort	- sorts the array with a smaller average time complexity than bubble sort, selection sort, and insertion sort	- poor performance with arrays that contain many repeated elements - quadratic worse-time complexity	O(nlogn)	O(nlogn)	$O(n^2)$

Merge Sort	- sorts array with smaller average time complexity than bubble sort, selection sort, and insertion sort - O(nlogn) worst-time complexity - efficient for large datasets	- slower than other sorting algorithms for smaller datasets - requires additional space if temporary array is used	O(nlogn)	O(nlogn)	O(nlogn)