Sorting Algorithms

Algorithm	Strategy
Bubble Sort	Uses two nested for loops to compare adjacent elements in an array. The algorithm starts by comparing the first and second elements, swapping them if the first element is greater than the second. It then moves to the second and third elements, comparing and swapping if necessary, and so on, until it reaches the end of the array.This process is repeated for arraySize -1 times or until the algorithm makes a pass through the entire array without swapping any elements.
Selection Sort	Uses 2 for loops nested inside each other. The algorithm works by iterating through the entire array to find the smallest element, then swapping it with the first element. It then moves to the second position and finds the next smallest element in the remaining unsorted portion of the array, swapping it with the element in the second position. This process is repeated for every element except the last one (for a total of arraySize -1 times).
Insertion Sort	Uses 2 for loops nested inside each other. It starts with the first element in the array, which is considered the sorted portion, and then iterates through the remaining unsorted portion of the array, one element at a time. At each iteration, the algorithm takes the current element from the unsorted portion and compares it to the elements in the sorted portion, moving larger elements to the right to make room for the current element. Once the correct position for the current element is found, it is inserted into the sorted portion of the array.The algorithm repeats this process for every element.
QuickSelect	Used to find the Nth smallest value in an unsorted array. It uses a combination of partitioning and recursion to efficiently locate the desired value. The partitioning method involves selecting a pivot point and comparing all other elements in the array to the pivot, then relocating the elements to the right or left of the pivot accordingly.
QuickSort	Works by partitioning an array around a pivot element. The algorithm selects a pivot element and then moves all elements smaller than the pivot to its left and all elements greater than the pivot to its right. This process is repeated recursively on each resulting sub-array until the entire array is sorted. At the end, all the sub-arrays are merged.
Merge Sort	Works by dividing an array into two halves, sorting each half separately, and then merging the two sorted halves back together. The process is repeated recursively on each resulting sub-array until the entire array is sorted.

Pros/Cons and Time Complexity

Algorithm	Advantages	Disadvantages	Time Complexity		
			Best Case	Avg. Case	Worst Case
Bubble Sort	Easy to implement	Time complexity with bigger data sets	Array sorting in descending order O(n)	Array is mildly sorted O(n^2)	Array is unsorted O(n^2)
Selection Sort	Works best on small list	Time complexity with bigger data sets	Array sorting in descending order O(n^2)	Array is mildly sorted O(n^2)	Array is unsorted O(n^2)
Insertion Sort	Less comparison made than bubble sort	Time complexity with bigger data sets	Array sorting in descending order O(n)	Array is mildly sorted O(n^2)	Array is unsorted O(n^2)
QuickSelect	Indicate the Nth smallest value in an unsorted array	Doesn't sort the array 🗿	The partition method starts with an ideal value as pivot O(n)	Partition starts with a random value as pivot O(n^2)	The Partition starts w the first or last value as pivot O(n^2)
QuickSort	Fast and time efficient	Bad when you get bad RNG 🗿	The partition method starts with the median as pivot O(nlogn)	Partition starts with a random value as pivot O(nlogn)	The Partition starts w the first or last value as pivot O(n^2)
Merge Sort	Significantly faster than bubble sort in larger sets of data Extremely efficient and stable	Takes up more memory	Array sorting in ascending order O(nlogn)	Array is mildly sorted O(nlogn)	Array is unsorted O(nlogn)